TOWARDS CLASSIFYING AND SELECTING APPROPRIATE SECURITY VISUALIZATION TECHNIQUES

by
David Barrera

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
September 2009

Abstract

Visualization of network security events has become an important method for detecting, responding to, and resolving security incidents. While there are many security visualization tools and techniques available, each one may require a different run-time environment and data input, making it difficult for a network security analyst to try them all (or a significant subset) and select those that work best for a specific incident or purpose.

This thesis analyzes three common classes of network attacks that security analysts encounter. Relevant variables which help understand and resolve an incident are identified in each specific class of attack. We then survey a large set of network security visualization techniques and use a task-based methodology to assess the usability, insight gained and overall usefulness of visualization tools for specific classes of attacks. We also recommend the most appropriate techniques for visualizing each attack and suggest other features that could help provide more insight.

Acknowledgements

First and foremost I would like to thank my supervisor, Dr. Paul van Oorschot for helping me with my thesis and throughout my degree. This thesis would not have been possible without his valuable input and guidance. I would also like to thank my family for their continued and unconditional support. Finally a special thanks to my colleagues for their insightful discussion and comments.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Network security administrators and analysts are tasked with protecting the integrity and maintaining operations of today's computer networks. They achieve these tasks by constantly monitoring data flow in and out of the network and looking for signs of misuse or attacks.

A number of tools have been designed over the past 10-20 years which help alert the analyst in the event of an intrusion or attempt thereof. These tools often require a deep understanding of the network in order to accurately distinguish a real attack from a false positive. Both anomaly detection and signature-based intrusion detection systems generate massive amounts of audit data. Sifting through these logs, coupled with system logs, firewall logs and server logs makes security monitoring a difficult and time-consuming task for network administrators. The problem is becoming worse as the cost of storage is at an all-time low, and the amount of data being saved is higher than ever. Network administrators are constantly in search of ways to make sense of the overwhelming amounts of data coming in from a vast number of sources.

Information visualization combines the power of human perception and computer graphics to help understand and analyze large data sets. In recent years, there have been many proposed network security visualization tools and techniques ([34, 54, 63, 26, 40, 22, 69] to name a few), all of which claim to help the analyst gain insight into the current status of the network. While a visual approach to representing network data frequently does help analysts, the incorrect selection of a visualization technique might not reveal any interesting patterns at all, or worse, hide important information from the administrator. Thus, it is important for the analyst to appropriately select a technique that is tailored to the type of data and the answers being searched for.

1.2 Motivation

There is a tendency in the security visualization community to claim that a particular tool or technique is the best for visualizing a particular type of problem. The claim is justified by showing an example of the tool on a particular data set (which may or may not represent real-world data sets in general). Such claims are often based on the tool designer's own analysis on a specific data set and involve neither a user study nor independent expert evaluation. Furthermore, a user study of any of the currently available visualization systems is difficult to produce.[1] However, taking a step back to identify the core problem, there appear to be only a small number of core visualization techniques (with many small variations on each one), and a relatively small number of common classes of network security events for which these techniques are proposed or used. We believe it is possible to successfully pair each of the most common network security events with one or more corresponding visualization techniques, in order to provide the analyst with a deeper understanding of events. This thesis pursues this goal.

1.3 Problem

The problem we address consists of identifying security visualization techniques that are well-suited for particular classes of network attacks. A visualization technique is more likely to be useful if it allows the analyst to easily detect specific properties and reveals insight about the attack and attacker. In this thesis, we describe the different phases of network security monitoring and identify the main tasks required by the analyst to detect specific threats. We then list the currently available visualization tools and techniques and create a classification based on the type of graph used. Finally, we pair each of

[1] A meaningful user study would require experts in network security with knowledge of the network from which data is being analyzed.

three common classes of network security attacks (denial of service, network scanning, and brute-force password guessing) with the visualization technique that offers the most insight and allows the analyst to perform assessments as efficiently as possible.

The scope of this thesis is limited to two primary areas of network security: *network monitoring*, and *analysis and diagnosis* of events. Using visualization for the third phase of network security (i.e., incident response) is not discussed. The primary scope of network monitoring and analysis is limited to external-to-internal communication, with a heavy focus on perimeter security and capturing data at border routers. With this scope, we label remote devices as the *source*, and internal hosts as the *destination* of attacks. While we do reference some internal monitoring concepts in Chapter 3, the "internal threat"[2] is not analyzed. Some visualization techniques and data capturing tools can be extended to visualize internal-to-internal communication, but this is beyond the scope of this thesis.

1.4 Main Contributions

The thesis has five main contributions to the field of network security visualization, as follows.

1. **Classification of visualization systems.** To the best of our knowledge, there has been no prior work in classifying visualization techniques for network security. We present an up-to-date classification of network security visualization techniques grouped by the type of diagram used to display network data. This classification allows us to identify strengths and weaknesses of each technique, and then match them with three classes of network security attacks. This classification provides a state-of-the-art survey of network security visualization techniques.

2. **Filtering technique to reduce occlusion.** We briefly discuss a new data visualization filtering technique [17] (published jointly with Mansour Alsaleh and Paul van Oorschot) that helps reduce occlusion by eliminating data which is of limited

[2] The internal threat refers to malicious users or software located inside the network perimeter, usually running with higher levels of trust.

use to the analyst. Our results show visualizations which are less cluttered and easier to interpret, providing more insight than visualizations without filtering. The filtering technique is provided as background, and not described in detail here, however, some of the example visualizations in our paper are used in this thesis.

3. **Selection process and recommendation of security visualization techniques.** Using prior work in usability and network security, we identify what appear to be three major classes of network security attacks that occur frequently, and analyze them. The analysis helps identify the main stages of incidents of each class, and what variables are important to the analyst at each stage. Using these variables and analysis, we outline a methodology for selecting an appropriate visualization tool or technique to help understand each class of attack.

4. **Analysis of IPv6 support in visualization techniques.** All the visualization tools we analyze currently assume that the underlying protocol carrying the network data in question is IPv4. As the updated version, IPv6, becomes more widely deployed, visualization tools are failing to keep up, and still have no support for it. Attackers can therefore remain undetected if their tools use IPv6. We analyze the current state of IPv6 support in visualization tools, and identify the changes necessary for tools to support the new protocol.

5. **Proposal of new visualization techniques.** We propose a new visualization technique which we call *grid heatmap* to help detect denial of service attacks without occlusion. For the analysis of brute-force password guessing attacks, we recommend using *word clouds* along with *parallel coordinate plots.* In the case of IPv6 data sets, we offer two contributions [18]: a *white space filtering technique* that allows the visualization of the entire populated IPv6 address space, and the use of *treemaps* to visualize the hierarchy of IPv6 addresses.

1.5 Overview of Results

After looking at many freely available visualization tools and techniques, we have generated a classification which groups them into three main categories: (1) scatter plots; (2) link graphs; and (3) geographical and space-filling maps. We have identified the phases in the workflow of network security analysts and reviewed common classes of network attacks (denial of service, network scanning and brute-force password guessing). We have recommended visualization techniques for each class of attack and proposed new visualization techniques in cases where we were able to improve upon existing visualizations.

1.6 Organization of Thesis

The remainder of this thesis is structured as follows. Chapter 2 provides background concepts for both information visualization and computer networks, to help understand the network security attacks and visualization techniques. Chapter 3 provides a brief introduction to network security as well as some intuition as to how visualization can help analysts; reviews the work flow of a typical network security analyst and describes three major classes of security-relevant events that visualizations are used to identify, capture, and explore; and provides detailed description and analysis of phases of the analysts' work flow and example network security attacks, identifying key variables important for visualization. Chapter 4 reviews the most common security visualization techniques in the literature, including examples and a summary displaying features and drawbacks identified for each technique. Chapter 5 informally proposes a methodology for visualization technique selection, combining the classes of attacks from Chapter 3 with the techniques from Chapter 4. Chapter 5 also presents a new visualization technique and two new uses of existing visualization techniques for detecting and analyzing network attacks. Chapter 6 explores how visualization for network security data will be affected by the move to IPv6 and suggests two techniques for displaying the new protocol. Chapter 7 discusses limitations, future work and concludes.

Chapter 2

Background and Related Work

This chapter is divided into four main sections. The first introduces data visualization concepts. The second section reviews types of network security data, and where they can be obtained. The third section presents background on networking and network security. The last section covers some of the related literature which attempts to help organize visualization tools and techniques available for network security.

2.1 Data Visualization

Information visualization is defined as the representation of information through graphical means in order to convey a concept clearly [22]. Data visualization is a subset of information visualization and focuses on the visual representation of data rather than abstract concepts. Data visualization is used in a wide range of fields to illustrate, interpret, understand and analyze all types of data and leverages the power of human processing, along with our natural ability to quickly detect patterns, trends and changes in images. The primary objective of data visualization is to gain insight into a data set.

Visualization has many benefits. When large amounts of data can be compared at once (through a single graphical representation), information can be communicated more efficiently. There is no longer a need to manually interpret every data point, and rely on our short-term memory to compare values. Once the information has been communicated, questions about the data set can often be answered quickly, such as: “does this data representation reveal abnormal events?” or “are there any outliers?”. Visualization can also help support decisions, particularly in time-sensitive cases, where manual analysis or inspection is not possible.

Data visualization however cannot be seen as a replacement for text-based data analysis tools. Text-based tools will often provide extremely fine-grained detail about the data set, which cannot always be represented visually. This is due to certain inherent problems with visualization tools such as limited size displays, labeling, and occlusion (discussed in more detail in Section 2.1.3). Visualization could be described as a trade-off between a high level overview and a detailed perspective, requiring the user to actively switch between modes in order to navigate through the data. Ben Schneiderman noted this work flow and coined the *information visualization mantra* [73]: "Overview first, zoom and filter, details on demand". This mantra describes a 3-step technique for analyzing data, which usually turns into an iterative process.

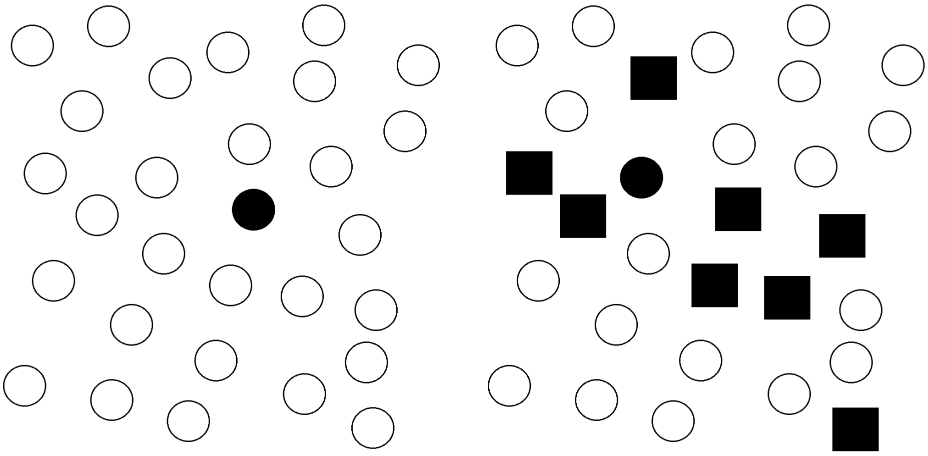
The graphical representation of data is created by using colour, shape, position, size or any other graphical property to encode information. The large number of combinations of these properties makes it possible to display thousands of data points at once, and still allow the brain to detect changes and patterns.

2.1.1 Preattentive Processing

Preattentive processing [42] is defined as visual processing that occurs prior to the act of selection, more specifically, to the humans' built-in ability to automatically (i.e., without searching) detect parts of images that "stand out" from the rest. Preattentive processing requires on the order of 250ms or less and requires virtually no effort other than the analyst looking at the image.

In Figure 2.1(a), we show an example image in which the user detects the filled-in circle using preattentive processing. Notice that it is extremely simple to pinpoint the circle that stands out from the rest. Figure 2.1(b) shows an example where the user needs to actively search for the unique part of the image, in this case a filled in circle.

As discussed in Chapter 4, many visualizations tools take advantage of preattentive processing to minimize strain on the analyst and maximize effectiveness of the tool. However, many tools are unable to exploit this feature due to occlusion or other technical limitations.



(a) Preattentive Processing (b) Non-preattentive Processing

Figure 2.1: Preattentive processing

2.1.2 Visualization Process

In the book *Visualizing Data* [35], Fry defines the visualization process (i.e., the steps required for converting raw data to the insightful visual representation of that data) in 7 steps.

1. **Acquire.** Data is captured or recorded.

2. **Parse.** Because the saved data can be in a number of formats, a parsing tool[1] reads in data and outputs it in a format that is suitable for the visualization tool.

3. **Filter.** Data is filtered to remove unrelated or uninteresting points in an attempt to reduce occlusion.

4. **Mine.** Similar to the filter step, data is reduced or modified through mathematical or statistical operations.

5. **Represent.** This step is the initial attempt at a visual representation.

[1] Parsers are generally written in scripting languages such as Perl or Python.

6. **Refine.** Once the data has been visualized for the first time, an iterative process of focusing on a specific part of the visualization, or further filtering, or even changing the visualization technique can be done.

7. **Interact.** This step involves exploring data by changing viewpoints or "pivots" (interesting data points used an anchors for further exploration).

The process begins with capturing data (1) either from log files or from a network monitoring device. This data is captured in its purest form, without any modifications. Parsing (2) the raw data is necessary to convert the file into a more flexible file format that can be imported into other applications for further processing. Some applications support raw data, but many more applications support comma separated values (CSV) or tabular entries. Filtering and mining (3, 4) involve the analyst interacting with the data set. It is common for security analysts to encounter data sets that contain *Internet Background Radiation*[2] (IBR) which needs to be filtered out in order to make clearer, less noisy visualizations. The filtering is often done by hand, relying on the analyst's expertise (which makes analyzing large data sets a time-consuming task). We discuss our own work on automatic data filtering in Section 3.3. The final steps of the visualization process are to generate an image (5) and look for sections that show interesting characteristics. Optional steps (6 and 7) can help the analyst detect previously unknown threats by interacting with the data set.

2.1.3 Problems with Visualization

While visualization provides many advantages over text-based tools, there are certain problems that can make visualization less useful. For example, occlusion (the visual obstruction of data) can make many data points overlap, preventing the analyst from viewing areas of the visualization where relevant data is contained. Labeling is another potential problem: too many labels on data points can hide actual data values, or overlap

[2] IBR is comprised of unsolicited, generally untargeted traffic coming from a large number of hosts (compromised by old worms and viruses) which were never disinfected and still look for new hosts to infect.

to the point of becoming illegible. Parallel coordinate plots (described in Section 4.3.3) often suffer from this problem due to their layout. 3D visualizations also require the analyst to actively interact with the tool to uncover data that is hidden “behind” another point. Reading precise values directly from graphs is also difficult, particularly when scale is non-linear, or too many labels are displayed. These limitations are mostly inherent to visualization in general, rather than specific to security visualization.

2.1.4 Visualization for Network Security

Network security visualization links the fields of data visualization, information visualization and network security to provide a graphical representation of network security events. These events (see section 2.2) can range from abstract data such as log entries or intrusion detection alerts, to measurable data such as average throughput or link saturation. One objective of data visualization in the field of network security is to help the analyst understand a particular event, and prioritize it among a sea of other events seen at the same time. Another important focus of visualization for network security is to help the analyst form a mental image of the current network status. This helps the analyst understand context [38] and generate better assessments in a timely fashion. More precisely the goal is not only to help the analyst detect attacks, but also understand the attacks and help prevent them in the future.

In the book *Applied Security Visualization* [62], Marty notes that there seems to be a *security visualization dichotomy*: good visualization tools are written by experts in visualization, who are rarely experts in security; and security experts are rarely experts in visualization. This dichotomy often leads to tools that generate *pretty pictures* but do not provide much insight, or tools that require high levels of expertise to use. Network security visualization aims to bridge this gap and provide visualizations which give analysts a better understanding of attackers and their attacks.

Complementary Visualizations

We define the concept of *complementary* (or secondary) *visualizations* as two or more visualization techniques or tools which don't provide much insight independently, but when displayed together significantly increase the analyst's understanding of an event. In security visualization, complementary visualizations are particularly useful, since viewing data in multiple ways can help the analyst discover abnormal behaviour, or provide deeper insight into a detected attack.

2.2 Data Types and Sources

Data visualization literature generally defines three types of data:

1. **Ordinal Data.** This is qualitative data in which there is an inherent order. However, no numerical comparison can be done between two data points. Examples of ordinal data include the priority (high, medium, low) of intrusion detection alerts.

2. **Categorical Data.** These are nominal values usually based on a type or label. There is no measurable quantity between points, so data is difficult, if not impossible to compare. Examples are types of log messages, names of applications or services and secure shell usernames.

3. **Interval Data (continuous measurement).** This type of data numerically describes an object, and possesses comparable and quantifiable values. Examples include packet size, time of arrival of a packet and sequence numbers.

Interval data is the easiest to display graphically, since the inherent numerical value can simply be plotted on a graph. Ordinal and categorical data must be processed and quantified in order to be visualized.

Network security administrators collect data from several sources. Servers, hosts, firewalls and other devices can generate data such as logs or network traces, which are all important to the network administrator. A typical network administrator will need to store the following types of data (while there are other sources, but this thesis focuses mainly on the following).

1. **Log Messages.** These are text records containing details of events. Logs are also known as event logs, event records, or audit records. When several logs are grouped together, the collection is known as an audit log or audit trail. Logs generated by the same application typically have the same format and are therefore simple to parse. Log messages are generated and then sent via email, saved to a storage device, or sent to a log aggregation facility. Some examples of log messages are:

   i) Intrusion Detection Systems (IDS) alerts. An IDS will generate many alerts or logs, especially if it is not configured correctly. Most IDS generate an overwhelming number of false positives. IDS logs are created when one of the rules (in the case of signature based IDS) was matched.

   ii) Firewall logs. Firewalls can record details about connections that were allowed, blocked or ignored. For very restrictive firewalls, logging all blocked packets will generate a substantial amount of information, so the configuration is usually dependent on the network architecture and configuration of the firewall.

   iii) Syslog logs. Syslog is a log aggregator on Linux systems that has become the *de facto* standard for storing and managing log files. Syslog aggregates logs from authentication services such as the Pluggable Authentication Module (PAM) or Kerberos, web servers, cron (a job scheduler) jobs and any application or service that supports logging to syslog.

   iv) Application logs. Applications can generate their own logs in their own format. Most enterprise-grade server applications support logging to syslog for log management and aggregation. However some applications simply output debug data and alerts to a file, which must then be parsed or read manually.

2. **Network Traces.** These are the full network data dumps off the wire obtained from passive network analysis. Network traces require large amounts of storage

space, but are very useful in forensic analysis of incidents, since they store every single packet (including header and payload) seen on the network in a separate data structure.

Netflow is a commonly used compact representation of a series of packets in a network conversation between hosts [24]. Each netflow record stores a 5-tuple (sometimes a 6 or 7-tuple depending on the protocol version used) in the form of:

```
<Timestamp, source IP address, destination IP address, source
port, destination port (, packet count, bytes transferred)>
```

This format greatly reduces the space required to track long lasting connections (e.g., a 650MB file download will only require a few bytes to describe, with the downside of losing the payload level information, and the exact packet arrival intervals).

2.3 Networking Concepts

This thesis focuses on the visualization of network level intrusions and attacks. Therefore it is important to have some background in networking to better understand the events being analyzed. This section reviews some general networking concepts and describes the tasks of a network security analyst.

2.3.1 The Network Stack

Operating systems provide a network stack that software services use to communicate over a network. The stack is divided into a specific number of layers (depending on the definition used) with each layer being responsible for a particular task. We will use the "Internet model" as defined by the Internet Engineering Task Force (IETF) in 1989 [48]. The layers are as follows.

1. **Link layer.** This layer is responsible for reading electrical impulses off the wire and translating them into a system-readable sequence of bytes. This layer also defines

the hardware addresses of systems, known as Media Access Control or MAC address; and is in charge of reading and verifying a checksum trailer on every network frame received, or appending a checksum to every frame sent. Data from the Internet layer (layer 2) is said to be *encapsulated* between the address header and the checksum trailer added at this layer. Address Resolution Protocols (ARP) are also described at this layer.

2. **Internet layer.** RFC 1122 [48] requires that all network stacks implement at least the Internet Protocol (IP) and the Internet Control Message Protocol (ICMP) at layer 2. The core functionality of the Internet layer is to verify that incoming link layer frames are correctly formatted, to process header options, to perform reassembly and to pass on the reassembled packet to the transport layer. Outgoing frames are fragmented if necessary, and the correct first hop is selected with help from the operating system's *routing table.*[3]

3. **Transport layer.** The transport layer defines transport protocols such as the User Datagram Protocol (UDP) and the Transport Control Protocol (TCP). The concept of port numbers is described in this layer, as well as session control for TCP and best-effort delivery for UDP. Some checksumming is also performed at this layer. TCP windowing and congestion control algorithms are all applied at this layer.

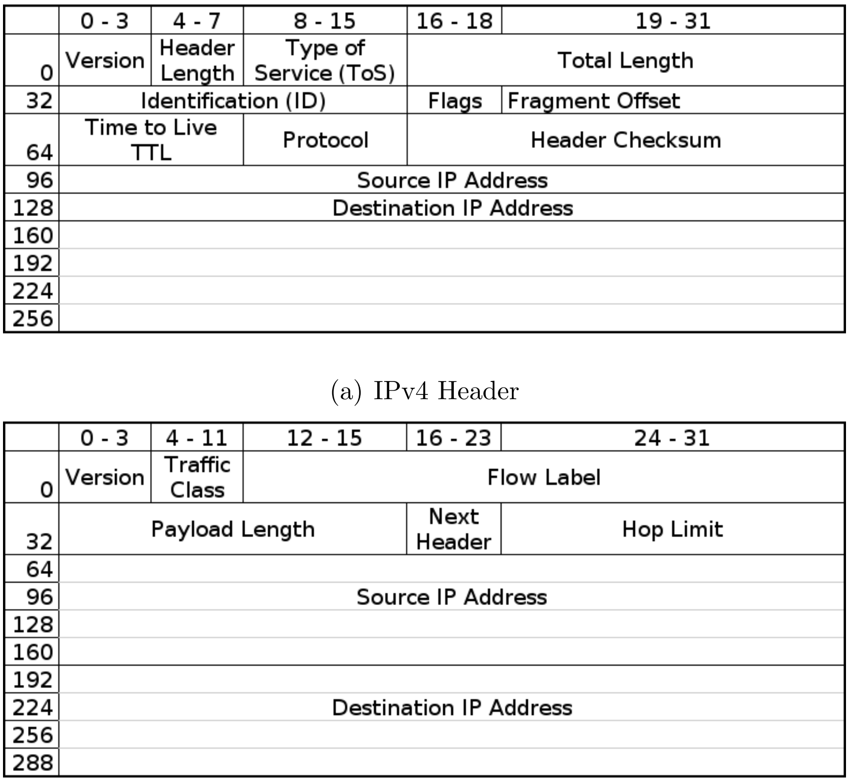
4. **Application layer.** The top layer of the network stack is usually the interface between the network and the user. These interfaces are usually in the form of software applications such as OpenSSH or an FTP client or a web browser, which will use the underlying transport protocols and in turn the lower level protocols.

The scope of this thesis is primarily focused on the Internet layer (layer 2) and Transport layer (layer3), since most of the data recorded will come from these two layers. Some application-specific details are important for forensic analysis (e.g., it might be useful to

[3] A routing table is a list of network address ranges and network interfaces through which those networks can be reached including the next network device. Routing tables can be built statically or learned dynamically through a routing protocol such as RIP, OSPF or BGP.

know what version of vulnerable web server software was exploited), but that data is mostly categorical and difficult to visualize. It may also be useful to know what hosts on a local network are communicating the most, but as explained in Section 1.3, the main focus of this thesis is external-to-internal communications. Since layers 2 and 3 use packets as their data units, we'll examine the structure of a packet.

Packet Headers



	0 - 3	4 - 7	8 - 15	16 - 18	19 - 31
0	Version	Header Length	Type of Service (ToS)	Total Length	
32	Identification (ID)			Flags	Fragment Offset
64	Time to Live TTL		Protocol	Header Checksum	
96	Source IP Address				
128	Destination IP Address				
160					
192					
224					
256					

(a) IPv4 Header

	0 - 3	4 - 11	12 - 15	16 - 23	24 - 31
0	Version	Traffic Class	Flow Label		
32	Payload Length			Next Header	Hop Limit
64	Source IP Address				
96					
128					
160					
192	Destination IP Address				
224					
256					
288					

(b) IPv6 Header

Figure 2.2: Comparison of IPv4 and IPv6 packet headers

Headers are located at the beginning of packets, preceding the payload. Headers contain information such as the source and destination IP addresses, source and destination ports, time-to-live values and any other critical information that the network infrastructure requires to get the packet to its final destination. A header format is usually static and defined by specific RFCs. In the case of IPv4, the RFC is 791 [49] and for IPv6, headers are defined in RFC 2460 [28]. Figure 2.2 shows the headers side-by-side. Notice that Figure 2.2(b) has a smaller number of fields, but the source and destination IP addresses are 128 bits long.

While most visualization tools assume the underlying protocol is IPv4, IPv6 is currently gaining adoption. When they deal with IPv6 data sets, many visualization tools will fail by crashing or ignoring the protocol entirely. In Chapter 6, we explore this problem and possible solutions.

Packet Payloads

Payloads contain the actual data being delivered from one host to another. This data can range from human-readable data such as unencrypted Hypertext Markup Language (HTML), to encrypted data or combinations thereof. Payloads contain data that could personally identify users. This often leads to a practice of capturing only header data, or removing payloads after capturing in order to preserve privacy. While many visualization tools only require header data, some tools allow visualization of the entire payload and provide more information to the analyst.

2.3.2 IP Addressing

IP addresses for network devices are like telephone numbers. They uniquely identify each device on a network, and remote hosts do not need to know the physical location of another host in order to send it data. The currently deployed Internet infrastructure runs mostly over IPv4 [41], so it is common to see visualization tools representing IPv4 addresses. IPv4 addresses are written in a *dotted-decimal* format that contains four 8-bit numbers. All IP addresses contain 2 parts: a host segment and a network segment. The following example illustrates how an IP address is split into two segments.

```
                       Decimal                     Binary

IP address    : 192.168.1.20   11000000.10101000.00000001.00010100

Subnet mask   : 255.255.255.0  11111111.11111111.11111111.00000000

Network addr. : 192.168.1.0    11000000.10101000.00000001.00000000

Host address  : 0.0.0.20       00000000.00000000.00000000.00010100
```

A logical *AND* operation between the IP address and the subnet mask reveals the network address, and the inverse operation reveals the host address. IP addresses belong to one of 4 classes: A (IP addresses that start with 0-127), B (128-191), C (192-223) or D (224-255). Each class has a fixed number of bits for subnet masks. This number can be varied to make smaller networks.

Subnets are important specifically for terminology, as many IP-based visualization tools display only a part of the address, like the network segment of a class A or class B address, and don't visualize the host level. Other visualization tools only display the host segment.

2.4 Related Work

To the best of our knowledge, there has been very limited work in attempting to match specific types of attacks with specific visualization tools or techniques. The two most complete references are security visualization books by Greg Conti [25] and Raffael Marty [62].

In *Security Data Visualization* [25], Conti presents an introduction to network security visualization. A number of security-related actions such as port scans, firewall logs management and vulnerability assessment are reviewed, with example visualizations. Other examples include the visualization of malware and binary files, which are not covered in this thesis. Conti highlights the importance of selecting the right visualization tool for the job, but the tools he chooses for specific events are his personal choice, with the rationale behind the selection rarely discussed.

In *Applied Security Visualization* [62], Marty focuses on visualizing data that helps secure organizations operationally. Rather than visualizing files, Mary shows examples of visualization of log files, perimeter network monitoring and insider attacks. The book has a heavy emphasis on the theory of information visualization and in understanding the different types of data available in network security. Marty also briefly discusses how a type of graph should be selected. His analysis is geared primarily towards understanding the number of variables and what type of information can be obtained from each visualization technique (comparison of absolute values, or displaying how variables relate). Our approach focuses on identifying a specific type of attack, and then selecting the most appropriate visualization technique.

There has also been some initial work in designing security visualization tools based on requirements of the user [39] which seems to have some positive results. A drawback of this study is that undergraduate students were representing the "expert" network administrators. However, user studies should not be discarded as they are a useful first step in designing visualization tools and, in addition to this thesis, could lead to more focused evaluations and user-centered designs.

Chapter 3

Three-Phase Model and Classes of Network Attacks

This chapter provides a brief introduction to network security and discusses how visualization can help analysts detect threats. We also review the work flow of a typical network security analyst, and describe three major classes of security-relevant events.

3.1 Introduction

Visualization is one of the many tools in the arsenal of a network security analyst. Graphically representing textual data can not only help detect new attacks as they unfold, but can also help analysts understand how attacks took place, and thereby how to prevent them in the future. Visualization for network security is a helpful way to interpret the vast amounts of data to be reviewed by administrators.

In this chapter, we review key network security concepts and define the phases in the work flow of a network security analyst. By clearly defining these phases, we are able to more accurately understand the requirements for each phase, which in turn will help facilitate selection of the most suitable visualization techniques. For each phase, examples of well-known attacks are discussed.

In Section 3.4, we analyze real-time network monitoring, and how administrators can benefit from working with real-time data. Section 3.5 reviews the analysis and diagnosis phase of network security. Finally in Section 3.6, we discuss denial of service attacks, network scanning, and brute-force password guessing attacks against services that require authentication.

3.2 Network Security

Network security analysts are tasked with defending resources and minimizing downtime created by attacks. A wide variety of freely available tools can help analysts perform their job quickly and effectively. Most of these tools (SiLK [11], Argus [1], tcpdump [13], Wireshark [16]) are text-based and not very user-friendly, requiring the analyst to leverage skill and prior knowledge in order to accurately detect attacks. There are also a variety of visualization tools that can help the analyst, but due to the lack of “real-world” testing, their value is questionable.

Network security is frequently associated with *intrusion detection*, and while there are other functions which are part of network security, intrusion detection is one of the most widely encompassing, covering all of the stages of an attack (reconnaissance, exploitation, post-exploit). Intrusion detection includes unauthorized attempts of accessing network resources (servers, end-user devices and other infrastructure).

Intrusion detection systems (IDS) work under one of two main approaches (or a combination thereof): signature-based detection and anomaly-based detection. The former relies on a database of patterns associated with packets or behaviour that should not be allowed on the network, while the latter uses heuristics and machine learning to detect packets or behaviour that are abnormal or *suspicious*.[1] When an IDS detects malicious or suspicious activity, it generates an alert (and can also be configured to drop that particular packet or block the activity).

Regardless of the type of network IDS being deployed, the motivation for its use is generally to discover and prevent attackers from disrupting or abusing the network infrastructure. Information learned from an intrusion detection system can also be correlated with data captured from different sources, such as those discussed in Section 2.2.

[1] In anomaly detection, there is a training phase in which normal network behaviour is modeled, and then activity that falls outside the bounds of normality is classified as suspicious.

3.2.1 The Three-phase Model for Network Security

To select a visualization tool, it is useful break down the problem we are trying to visualize. Recently work has gone into carrying out studies with the purpose of researching and understanding the steps in the work flow of a security analyst. By understanding the work flow, we can tailor visualizations to each phase rather than look for a tool that can visualize a broader range of events. In one of these user studies, Goodall et al. [39] determined that analysts routinely use a three-phase model for network intrusion detection. The phases were identified to be:

1. **Monitoring phase.** The network is observed, and the analyst looks for signs of anomalies and malicious behaviour. It is in the monitoring phase that the analyst builds a mental image of the current network, trying to include as many variables as possible for increased comprehension.

2. **Analysis and diagnosis phase.** In this phase, an event has triggered the analyst's attention, who proceeds to gather more specific details to determine if the event requires further investigation. If the event turns out to be an intrusion or attack, then more data from other sources can be analyzed, focusing specifically on the context and time period of the intrusion.

3. **Response phase.** This phase involves the analyst deploying countermeasures or safeguards against the attack, with the objective of preventing the attack from reoccurring.

We believe that this three-phase model is extremely useful since it provides a clear separation of the tasks involved in network security analysis. Furthermore, these phases appear to be widely accepted. Network security visualization is heavily task-oriented, with no single tool capable of solving all problems at once. By leveraging the three-phase model, we are able to more accurately identify the key requirements for each phase (as described in Sections 3.4 and 3.5), and find a well-tailored visualization tool that meets the requirements for that specific phase.

A different user study by Yurcik et al. [84] suggests that an overall situational awareness of the entire network is essential for network administrators. However, they also require the ability to drill-down and view specific details of the event being analyzed. Based on these requirements, the information visualization mantra, and the existing tools surveyed in Chapter 4, we believe that it is difficult for a visualization tool to provide both a high level overview *and* fine-grained detailed view at the same time. We also note that the requirements of visualization tools may vary from network to network, but in general requirements differ from the monitoring phase to the analysis phase. Therefore different visualization techniques might be appropriate at each phase.

Killcrece et al. [53] conducted a study of 29 Computer Security Incident Response Teams and found that a large percentage of the interviewed analysts' time is spent on analyzing artifacts (66%), monitoring IDS alerts (62%) and monitoring network and system logs (38%). These tasks were confirmed in a different user study by Botta et al. [20].

Goodall et al. [39] point out that most current visualization tools available are designed to graphically represent the monitoring phase. This is somewhat intuitive, since the monitoring phase requires the analyst to keep track of events occurring in real-time and also review data from a large number of sources. In the monitoring phase, a large number of independent variables are displayed (e.g., characteristics and trends of the data set in question), allowing visualization to be used to a greater extent. The response (third) phase is beyond the scope of this thesis (visualization helps detect attacks and analyze traffic, but the administrator or analyst's decision on how to proceed once they have obtained this knowledge is a further topic); we will focus on the monitoring phase as well as the analysis and diagnosis phase.

3.2.2 The Importance of Context and Time

Context

It is nearly impossible for a network administrator to accurately evaluate the severity, importance or impact of a particular event from simply examining a single log entry or alert. The importance of an alert will be heavily dependent on the context (i.e., other alerts, events, packets, etc. that were seen during the same time frame[2]) surrounding that particular alert. This means that the event is merely a starting point for exploration of data, with the end objective of classifying the event as an actual threat or as a false positive.

Once an event has gained the analyst's attention, he will generally proceed to supplement the event with data from other sources and proceed with further analysis depending on personal knowledge and experience. Aggregating this data provides the analyst with more information to reconstruct the timeline of the event and gain a better understanding of what actually happened.

For example a spike in network activity at 3 A.M. may mean that someone has gained unauthorized access to the network and is using resources. However, if this spike begins with a log message saying: "Nightly backup started at 3 A.M.", then the spike is most likely the backup taking place, and not a suspicious event.

Time

Most data in network security has a timestamp (a field containing the date and time at which the packet, log, or message was generated). If the data does not contain a built-in timestamp, the operating system or software can record the date and time at which that particular data was generated.

Time is one of the most frequently used variables in security visualization. Scatter plots, histograms, parallel coordinate plots and surface plots all use this variable to

[2] It is also possible to see attacks that occur in a different time frame, but typically there is a chain of events which can be followed up to the point of the intrusion or attack.

communicate the temporal context of data. Time is also useful to see what was happening at the same timestamp or near that timestamp to frame a better context.

Visualization with respect to time is a useful way of understanding the progression of an incident. Furthermore, time-based visualizations can help discover incidents that are related (occurring in the same time-frame) to the event that triggered the analyst's attention.

3.2.3 Network Security Analyst Work Flow

D'Amico et al. [27] offer insight into the work flow of a network security analyst. They identify the main steps taken by security analysts for moving from raw data to classifying a suspicious event as a security incident.

1. **Monitor raw data.** The analyst monitors various data sources with the objective of maintaining situational awareness (defined in Section 3.4). While many analysts look at alerts generated by IDS and firewalls to cue them when interesting events are detected, some administrators leverage their expertise (and perhaps visualization tools) to locate attacks.

2. **Search for interesting data.** At this step, some portion of the data set has focused the analyst's attention either by showing unexpected behaviour, or behaviour that is unexpected at that time of the day (e.g., heavy web browsing at 2 A.M.).

3. **Detect suspicious activity.** The analyst performs some basic anomaly detection by identifying unexpected behaviour in the previous step, and decides that the event merits further investigation to isolate the root cause and locate other instances of the same type of activity.

4. **Event classification.** If the suspicious activity has actually compromised hosts or disrupted business continuity, all data related to the suspicious activity is flagged as a security event that must be handled by a Computer Security Incident Response Team (CSIRT).

5. **Incident handling.** The response team locates infected systems and isolates them from the network. Countermeasures are put in place to prevent the same incident from occurring again.

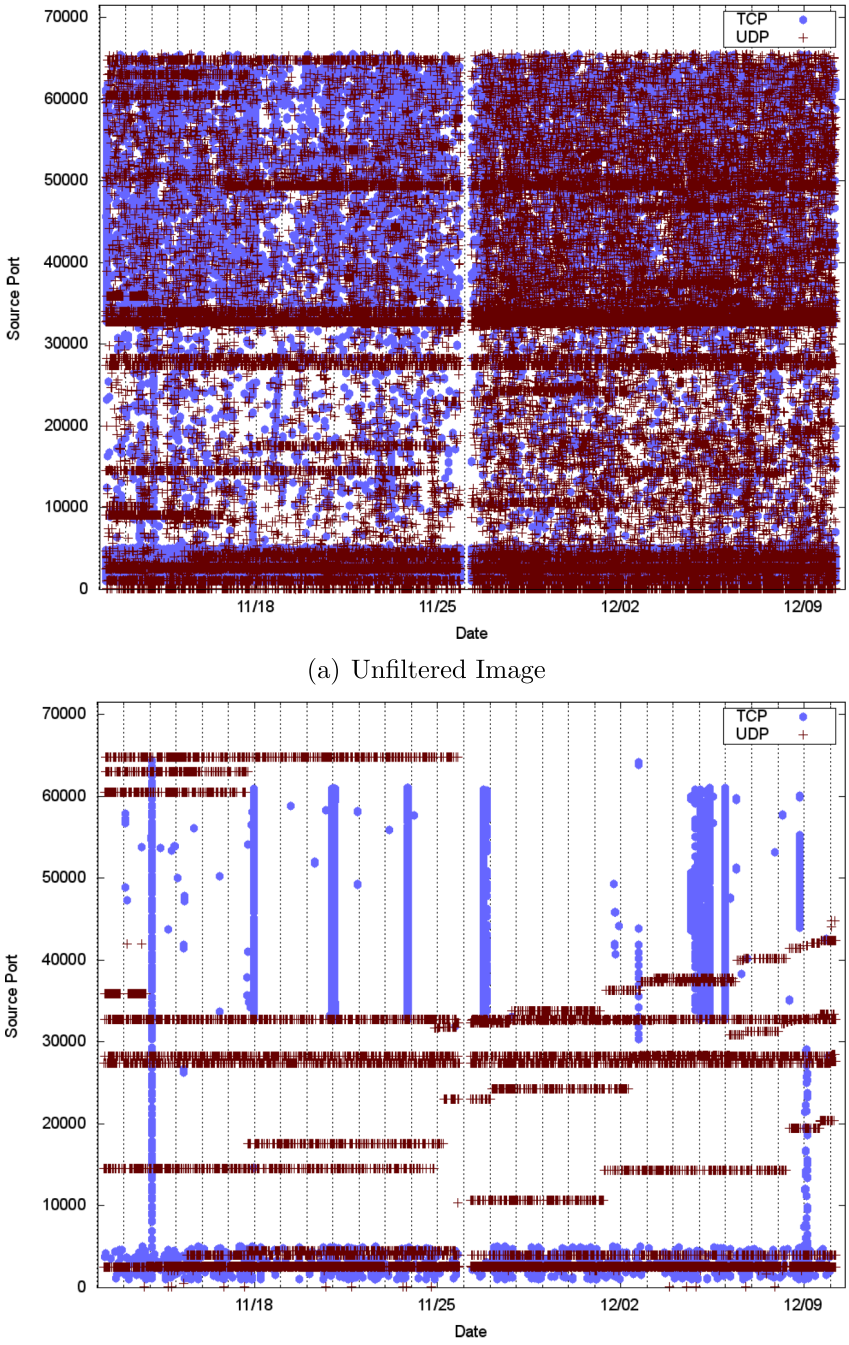
These steps might vary depending on the size of the security team at a particular organization. This thesis focuses on steps 1-3, where visualization can be used most effectively.

3.3 Data Filtering

Visualization tools can be useful because they are capable of displaying large amounts of data on a single image, and leverage human processing to locate complex patterns and trends. However, when attempting to visualize massive amounts of network data, analysts run into problems with processing this data, generating graphs, and interpreting them. Graphs with massive amounts of data tend to be very susceptible to occlusion, thereby counteracting the main motivation of visualization which is to simplify detection of malicious events.

Our work (published jointly with Mansour Alsaleh and Paul van Oorschot [17]) in the area of data filtering has produced results which are useful for visualization. The exposure maps technique of Whyte et al. [83, 82, 81] is used to identify remote hosts that are more likely to be malicious by using heuristics based on the success of previous connections. Once suspicious hosts are located, they are added to a pool of remote addresses for which flows will be visualized. Comparing the images before and after filtering reveals patterns that were not obvious in the original views. For example, Figure 3.1 (generated from a small university data set) reveals vertical lines in the filtered image. The data that is filtered out of the image corresponds to services that were offered by the destination network, and therefore is classified as expected or legitimate traffic.

This exposure map filtering technique is a good example of how current visualization tools can be improved without requiring access to source code or performing any other modifications to the tool. As noted in Section 2.1.2, the data filtering phase is usually



(a) Unfiltered Image

(b) Filtered Image

Figure 3.1: Scatter plot showing source ports used over time both before and after filtering with the exposure map technique [17].

done after parsing and before data mining. It is therefore critical to select the appropriate filtering technique with caution, to avoid losing important data.

3.4 Monitoring in Real-time

There are millions of Internet-enabled hosts online at any given time. Any single one (or many) could try to attack the network being defended. It is impossible to predict when one of these attacks, be it a massive worm outbreak or a targeted intrusion attempt, will take place. As an example, the Slammer worm (also known as Sapphire) infected 75,000 hosts in 10 minutes in July 2003 [65]. With this in mind, it is safe to say that reaction to an incident must often be quick in order to limit damage and ensure business continuity.

Visualizations are used to help the analyst build a mental image of the network and help make decisions quickly in the event of a breach or attack. The mental image may help the analyst in gaining *situational awareness* [61]. When a network administrator has this mental picture of the current network status, it is easier to decide if an IDS alert or a firewall log message is a false positive or an actual threat worth investigating.

Situational awareness is defined as the perception of environmental elements, the comprehension of their meaning and the projection of their status in the near future. This definition describes three phases: *perception, comprehension* and *projection* [30]. In the perception phase, data is gathered and monitored for interesting activity. The mental model of the network is built at this stage. During the comprehension stage, correlation, analysis and personal expertise are combined to determine if the interesting activity is dangerous enough to pose a threat. The analyst's mental image of the network is refined as well. Finally at the projection stage, threats are understood in depth, and it is possible for the analyst to forecast related types of incidents, and implement safeguards to prevent those incidents in the future.

An important use of real-time monitoring is *event triage*. The term triage is more often used in a medical context, referring to the priority with which patients are treated depending on their health status (i.e., when medical supplies are scarce, they will be given to patients with higher possibilities of surviving). Triage in network security is analogous to this concept, and refers to prioritizing alerts and log messages for more accurate response. Just like in the medical context, for analysts to perform accurate triage, they must have up-to-date information about the event and its context.

Later subsections 3.6.1 and 3.6.2 explain two network security attacks that benefit from real-time visualization for detection and quick response.

3.5 Analysis and Diagnosis

Analysis and diagnosis of network security data implies some form of investigation or research of a particular event. The objective of the analysis is to understand how an attack took (or is taking) place, what resources are affected, and how to prevent a similar attack in the future.

Analysis is important not only in cases where the intrusion was successful, but also when it fails. Understanding trends and changes in attacker behaviour can aid in the correct configuration of network security devices. A good example is firewall configuration: when firewalls were first deployed in the early 1990's, they were generally configured to *allow* all incoming traffic except specific types of known malicious connections. Today, most administrators will configure their firewalls to operate in a default-closed mode where incoming traffic is *denied* unless specified otherwise in a firewall rule.

While real-time monitoring uses dynamic, live data, the analysis phase will typically use somewhat more static data, such as network traces that have been captured and will no longer be changing (due to the analysis of a specific time-window), and logs no longer being added to the set; new information is being recorded elsewhere.

Later subsection 3.6.3 reviews an example attack where data must be analyzed to gain insight into intrusions and intrusion attempts.

3.6 Three Common Classes of Network Security Attacks

In this section we discuss the three common classes of network security attacks which are matched up with specific visualization techniques later in the thesis.

3.6.1 Denial of Service (Monitoring in Real-time)

i) High-level Summary

Denial of Service (DoS) and Distributed Denial of Service (DDoS) (expressed as DoS from here on) refer to the act of, through remotely originated actions, making a resource unavailable to its intended users. Resources generally refer to services running on a network enabled host (such as a web server, or a secure shell server). DoS is frequently accomplished by flooding the victim service with a high volume of requests. The server cannot tell the difference between the attacker's requests and legitimate requests, and with high probability will not be able to answer real requests. This type of attack may go on for a short period of time, or indefinitely, and may be associated with some kind of extortion [60] or other financial exploitation.

As an example, in 2003, the country of Estonia fell victim to several DoS attacks launched from across the globe [4]. The attack led to the shutdown of many government sites, as well as banks and other critical country infrastructure. Other examples of DoS attacks have targeted the root Domain Name Servers (DNS) [9] and large corporations [14].

ii) Technical Details

As the name implies, users can be denied access to many types of services. DoS attacks can been further classified according to the type of resource that is being manipulated. Some classes are as follows.

1. Consumption of computational resources, such as bandwidth, disk space, or processor time.

2. Disruption of configuration information, such as routing information.

3. Disruption of state information, such as unsolicited resetting of TCP sessions.

4. Disruption of physical network components.

Early DoS attacks [64] were based on simple Internet Message Control Protocol (ICMP, a.k.a. ping) flooding, and several other bandwidth consumption related attacks. The objective was to send more data to the server than their bandwidth could handle, saturating their connection and limiting capacity to answer legitimate requests. As Internet access became more widespread and access to bandwidth increased, ICMP DoS attacks became less common, largely due to their ineffectiveness.

Other more sophisticated attacks like SYN flooding [64] exploit the network stack of the operating system, by keeping a large number of half-open[3] connections active. Operating systems have a limit on how many half-open connections can be kept, and when the limit is reached, no new connections are accepted. Without necessarily flooding the bandwidth, the attack is effectively preventing new connections from being established, and therefore denying service to the users.

Bot networks (or *botnets*) are computer networks of zombie machines. These computers have been taken over by attackers and are being controlled remotely. As of 2009, there have been many denial of service attacks by these zombie networks, presumably controlled by a small number of people. These attacks are particularly difficult to stop because users of the end devices carrying out the attack are frequently unaware that their machine has been compromised.

Attacks by botnets can be considerably more damaging if *reflectors* [71] are used. A reflector is a host that responds with a new packet when a packet is sent to it. Web servers and DNS servers, for instance, fall under this category. When the zombie machines send spoofed packets to a victim machine, the victim attempts to complete a three-way handshake with a third party server. This amplifies the attack because the third party server will initiate a new connection back to the victim, using up another connection slot.

[3] A half-open connection is a TCP session that is awaiting the last step in the 3-way handshake.

iii) Defenses

While DoS attacks are largely regarded as unsolvable problems (i.e., the attacker will always succeed as long as his bandwidth is greater than the victims'), there are some mechanisms that help reduce the impact of the attack, or reduce its effectiveness. Ingress filtering [32] will drop any packets that come from non-routable networks such as 10.* or 192.168.*. However, if enough packets with spoofed addresses are sent, then the burden is moved to the inspection of each packet's source address, and network flooding might still occur. Working with ISPs can also help stop an attack, by dropping packets higher up in the network topology. In the Estonia DoS attack mentioned above, the only way city officials were able to mitigate the attack was by working with local Internet Service Providers (ISP), and temporarily blocking inbound connections originating from outside the country, which made it difficult for the rest of the world to learn what was happening within the borders. It is more difficult to DoS a backbone inter-ISP peering connection that has hundreds of megabits of bandwidth.

iv) Important Variables and Rationale

Any DoS attack that attempts to exhaust bandwidth or CPU resources is likely to happen within a short period of time (i.e., there is no such thing as a low-and- slow flooding attack). The connection flood must remain constant in order for the attack to be successful, otherwise a simple reboot would be enough to mitigate the attack. Thus, the analyst must have the capacity to detect changes in specific variables corresponding to relevant features of the data set that occur within small time increments. The changes the analyst is looking for to detect a DoS attack are related to networking metrics such as throughput, bandwidth, and port activity. Spikes in any of these measures are common, for example if a backup is started, but these events should be known to the administrator, and can be confirmed as normal when they occur. When several of these variables spike at once, and there is no other legitimate activity scheduled, a DoS may be taking place.

For the analyst to detect significant changes in any of the DoS-related variables, there must be a frame of reference as to how much traffic or activity is normally (as in *in conformance to the average*) seen. This means that simply displaying how a variable increases or decreases (without reference to the norm or previous values) will not provide the analyst with any insight, unless he can see if it is growing significantly beyond the average.

It is important to note that there may also be DoS attacks which, by sending a small number of packets to the victim, are able to cause the remote service to crash, forcing a manual restart to come back online. In some instances, only a single packet[4] or very few are needed to stop a service. In these cases, the analyst is not looking for spikes in activity, but rather the lack of activity on a particular server. In this case as well, the analyst is looking for a deviation from the norm.

Besides the frame of reference, it may also be useful to monitor port statistics. This will help the analyst identify ports that are seeing more traffic, and help decide on modifications to firewall rules. Port information will also reveal if a specific service is being attacked.

Finally one more useful variable is the number of source addresses, and if possible, the geolocation [67] of those addresses. One of the mitigation techniques during a DoS attack is to contact the relevant ISPs of the source hosts and have them limit their throughput or block their connections entirely. Having the capacity to see what countries are harbouring most infected hosts can help speed up response time.

3.6.2 Network Scanning (Monitoring in Real-time)

i) High-level Summary

Network scanning is usually the initial phase (reconnaissance) of an attack. If the attacker has no prior knowledge of the network being targeted, an initial enumeration of hosts and services is typically necessary. Scanning involves repeatedly (even if the frequency

[4] For example, the old Microsoft Windows 95 “land attack” exploit caused computers to crash upon receiving a network packet containing the same IP address as source and destination.

is very low) probing a remote system to discover what services are listening to incoming connections, and further service discovery will allow the attacker to determine if the discovered service is vulnerable to attack. While there is still debate within the security community as to whether or not scans are precursors to attacks [70], we believe this issue largely depends on the definition of "scan" being used. We also note that attempting to exploit an unknown system would be difficult.

Network scanning is easy to detect when a single host is rapidly probing all ports on a target machine. However, as of 2009, these types of scans are rarely seen, presumably due to the decline of scanning worms such as Slammer [65] and Code Red [66], as well as attackers moving to more stealthy techniques. Also, as we discovered [17], many attackers probe on average at most 4 ports each on a target machine to avoid detection with IDS. It is more common today to see low-and-slow scans (those that happen over an extended period of time), and distributed scans (where many different hosts scan a victim simultaneously).

ii) Technical Details

A simple network scan involves the attacker sending one or more packets to a victim machine and waiting for responses. If the target victim responds and is willing to establish a connection, then the attacker has successfully discovered an open port on the target system. Other responses such as "ICMP destination unreachable" and "TCP resets" might indicate the remote host is up, but not offering that particular service. The basic networking concept behind a TCP scan is the 3-way handshake. An attacker will send a TCP SYN packet to a particular destination port. If that destination port has a service running and accepting incoming connections, the system will reply with a TCP acknowledgement (ACK) packet. Once the victim responds, the attacker can optionally complete the three-way-handshake with a TCP ACK packet and establish the TCP session (this is known as a *connect* scan). Establishing the full TCP session is not always necessary for simple discovery since the victims' response is enough to detect if a service is listening or not; if the attacker does not complete the connection, this known as a *syn* scan or *half-open* scan. Scans that look for open ports on a single host are known as *vertical scans* or simply *port-scans*, and scans that try to find the same service/port on a range of machines are called *horizontal scans* or *port-sweeps.*

A number of difficulties arise when attempting to decide if an incoming connection is a scanning attempt, or simply a benign request. There have been a number of techniques developed to detect scans. Most of these techniques use the concept of detecting n events in t seconds. For example, Snort [12] and NSM [44] both check if a specific IP address has contacted more than n ports in a specific time period. This technique can help identify aggressive scanners, but will fail at detecting low-slow scanners (e.g., machines that attempt a single connection every day). More advanced algorithms such as Threshold Random Walk (TRW) [52] and exposure maps [83] are capable of detecting more sophisticated scanners, but suffer from high memory requirements and high false positives (under default configurations), respectively.

iii) Defenses

The main defense against network scanning is a properly configured firewall which drops unsolicited incoming connections, and can log connections from different sources. Firewalls will prevent access to specific hosts/ports, but if a publicly facing service is being offered, then inevitably it will be discovered. The objective of trying to detect scanning is to identify malicious remote hosts before they locate publicly facing services.

There are several intrusion detection systems that can detect specific types of scanning, but some preconfigured threshold must usually be set. For instance, Snort will classify a remote host as a scanner if it attempts 4 connections to different ports within a 60 second time window. Bro [3] will monitor the number of half-open connections and failed connection attempts. TRW will give remote hosts a score based on how many successful connections they make in comparison to failed connections. The score of each remote host is compared to pre-established thresholds that will classify that host as benign or as a scanner. Whyte et al. [83] classify a source as a scanner after a single failed connection, under the assumption that a properly configured host should not fail at all.

iv) Important Variables and Rationale

An obvious independent variable for port scan detection is the port number. More specifically, it is important to know which ports are being accessed and what protocol is being used. Simple (quick) scans tend to show up very obviously on graphical displays (long horizontal or vertical lines in the case of scatter plots), but low-and-slow scans will not be as clear. For this reason, having the ability to represent data over long time periods is useful. Coordinated and distributed scan detection is more difficult because it requires a wider view of the Internet (i.e., a highly distributed scan can be originating from multiple network blocks scattered over the IP address space).

3.6.3 Brute-force Password Guessing (Analysis and Diagnosis)

i) High-level Summary

In brute-force password attacks, services on the Internet that require some form of authentication to gain access are bombarded with a constant stream of login attempts. These attempts come from remotely controlled computers trying to guess authentication credentials with the objective of accessing and/or taking over the host. While Telnet and SSH (both used for remote shell logins) are frequently targeted, Telnet lacks some of the security features that SSH provides, so SSH is more widely deployed, and also more frequently attacked [8].

The SANS Institute maintains a list of the top 20 security vulnerabilities and issues. In their late 2008 report [10], SANS describes brute-force attacks as “the most common form of attack to compromise servers facing the Internet”. The impact of this is severe: fully up-to-date systems are just as likely to be compromised as older, unpatched systems if a weak username/password combination is being used.

ii) Technical Details

A brute-force attack is fairly simple to execute. The attack begins with the discovery of a remote service accepting username/password combinations. Once the host has been detected, an automated program will iterate through a *dictionary*, containing a list of candidate username/password pairs. Common usernames seen in dictionaries include: root, guest, admin, test and mysql. Passwords dictionaries frequently include simple strings like: letmein123, logmein, trustno1 or p4ssw0rd. If the target system rejects some specific credentials, new ones are tested until the attacker gives up, or access to the system is obtained.

Because each failed attempt might get logged, attackers run the risk of being easily detected and prevented from making further guesses. Distributing the attack is one way to overcome the limit, by having n different remote machines attempting unique combinations, but covering the targeted username/password pairs together.

iii) Defenses

According to the SANS institute [10], the best defense against brute-force password guessing is avoiding the use of simple, interactive username/password pairs for logging in to a system. It is advisable to use a stronger form of authentication such as key-based authentication to minimize the risk of a successful break-in. SSH keys are generally at least 1024-bits in length, making a brute-force attack against them very unfeasible. When key-based authentication is not possible, the best defense against brute-force password guessing is to use strong passwords which do not contain dictionary words. It is also helpful to limit the number of guesses a host can attempt in a defined period of time (e.g., 3 attempts every 10 minutes).

iv) Important Variables and Rationale

Depending on what type of analysis is required, and important independent variable for visualization graphs is the source IP address of the connection (services that have interactive logins are TCP based, so the source cannot be spoofed[5]). The source IP address in turn reveals information about the Internet Service Provider (ISP) and geographical area where the ISP is located.

Keeping track of the usernames guessed can help the system administrator identify accounts that are being attacked more frequently, and potentially enforce a stronger password policy for those accounts. While destination port numbers remain constant during an attack because the server generally stays on the same port, source ports can reveal insight into what operating system is being used, or what software is being used to launch the attack.

3.7 Further Discussion

This chapter has reviewed the phases in the work flow of network security analysts, and highlighted the differences between each phase. Three common classes of security-relevant activity which are the subject of network security monitoring, analysis and diagnosis were also discussed. These classes demonstrated the separation of requirements for displayed information during each phase, and that the independent variables to be graphed on charts, interactivity and level of detail required differ in each phase.

Real-time monitoring requires the analyst to maintain a high-level overview (for situational awareness) while attempting to minimize sensory overload as much as possible. Even though visualization allows for the interpretation of many data points at once, it is impossible to read through and interpret all data coming in from all sources simultaneously (especially in larger networks). Thus, it helps to eliminate some specific details from the real-time analysis to prevent occlusion [17]. On the other hand, the analysis phase

[5] The IP address of the last hop cannot be spoofed, but this does not guarantee that the attacker is at that same location. I.e., the attacker could be using a proxy.

Required Functionality	**Monitoring Phase**	**Analysis Phase**
Overview	High	Medium
Interaction	Low	High
Level of detail	Low	High
Num. of independent variables	Medium	High

Table 3.1: Requirements for monitoring vs. analysis

requires more detail in order to make a more informed judgement about the severity of an event.

The separation of tasks described in Section 3.2.1 does not imply that monitoring cannot be accomplished when working with analysis levels of detail, but that it is more complicated to do so due to potential occlusion. With this in mind, we have identified the required functionality of visualization systems and evaluated them depending on the phase of network analysis.

For example, if an administrator is monitoring a large network, keeping an overview of all hosts has priority over what port a particular host is accessing at any given time. The specific port information might be of interest, but the overall view provides more insight to the analyst, and it is also easier to interpret. At the analysis phase more detail is required to investigate an event, so the iterative process of zooming in and viewing details might be more appropriate than having a constant high-level view.

Table 3.1 summarizes the required functionality of visualization tools, and what level of importance (rated as low, medium and high) they have in each case. As mentioned previously, when analyzing a particular event, the administrator has already identified the time-frame, and no longer requires an *overview* of all data at all times. A high level of *interaction* is more important during the analysis phase, since one piece of data might lead to the discovery of other points of interest in either the same data set or in a different one. The monitoring phase requires less input from the analyst, except for when the visualization tool must be paused (in the case of animated[6] displays) or configured. The requirements for levels of detail are higher for the analysis and diagnosis phase, since

[6] In visualization, static images are usually preferred over animated displays to remove reliance on the analyst's short-term memory and continuous visual attention.

the analyst is capable of manually filtering through data during a longer time period. In the monitoring phase, keeping the number of independent variables underlying the visualizations high while not displaying fine-grained detail is key to maximize situational awareness while minimizing sensory overload.

Chapter 4

Classification of Security Visualization Techniques

4.1 Introduction

One of the goals of this thesis is to identify the most suitable visualization techniques (i.e., those that provide the analyst with the most insight) for specific classes of network attacks. In this chapter we survey visualization tools and techniques, and present a classification of techniques based on the graph types.

Lenger et al. [59] define a visualization method as "a systematic, rule-based, external, permanent, and graphic representation that depicts information in a way that is conducive to acquiring insights, developing an elaborate understanding, or communicating experiences". We believe this definition is accurate, but we refine it to be more specific to security visualization. We define a security visualization technique as a technique producing a graphical representation of security data (from logs, alerts, network traces, etc.) which helps network security analysts perform their job more effectively.

Most of the techniques outlined below were obtained from proceedings of the Annual Visualization for Cyber Security Workshop (VizSec) dating back to 2004. Other techniques were obtained from books and Human Computer Interaction (HCI) workshops. We have also implemented our own versions of some techniques when there was no publicly available software. We note that not every single visualization tool publicly available was included in our survey, since many tools offer little or no benefit over other techniques which are discussed in this chapter.

To the best of our knowledge there is no prior work in classification of visualization techniques for network security. There have been other attempts of classifying general-purpose visualization techniques. One example is the periodic table of visualization [59].

This is a tabular representation (much like the periodic table of elements in chemistry) of a number of techniques, organized by the type of visualization (data, information, concept, strategy, metaphor and compound). Due to the type of data being visualized in network security, visualization techniques in this area will generally fall under the data visualization and the information visualization categories except for instances where more abstract data such as policy or access control is visualized (these are beyond our scope). The periodic table of visualization also labels the techniques based on the overview/detail metric, as well as the type of critical thinking (convergent or divergent thinking) required to interpret the visualization (see Section 2.1.2).

Note that the focus of this chapter is on the visualization *techniques* not the tools that demonstrate or implement such techniques. In security visualization literature, new proposed techniques are accompanied frequently by a software tool. While the tool might demonstrate how the technique works on a real data set, it is an engineering accomplishment that could have been implemented in different ways.

After reviewing many publicly available visualization tools and papers, we have classified the visualization systems they describe into three main categories: (1) scatter plots; (2) link graphs; and (3) geographical and space-filling maps. All of the surveyed papers and tools are added into this classification as examples of each category.

4.1.1 Visualizing Processed Data

A recent trend in security visualization is to do more pre-processing before visualizing in order to maximize the effectiveness of the visualization. Early visualizations simply extracted a particular characteristic of a packet or a netflow and plotted it. While undoubtedly there is value to be gained by this approach (i.e., a picture is worth a thousand words), advanced pre-processing can help remove unwanted (or irrelevant) data, as well as help the analyst focus on specific parts of the data set.

One specific technique for pre-processing network traffic is discussed in detail by Alsaleh et al. [17]. This work presents illustrative visualizations which plot both raw data, and processed data. The raw data refers to a port number, IP address or sequence number (i.e., data that can be obtained directly from the packet or log without any further computation). Processed data refers to the idea of calculating a value from the data set, and then displaying the derived value rather than the original data. Visualizing processed data is more time-consuming than directly plotting (due to the time necessary for processing), but can frequently lead to interesting insight. For example, Figure 4.1 shows an *impulse graph* generated from a small university data set in 2007. The graph shows the port number on the x-axis, date on the y-axis and number of unique source IP addresses on the z-axis. The graph shows a sudden change in the number of unique sources seen contacting port 53 starting about halfway through the capture period. The information revealed in this graph is not directly accessible from any one packet, and must be computed after the data capturing period is complete.

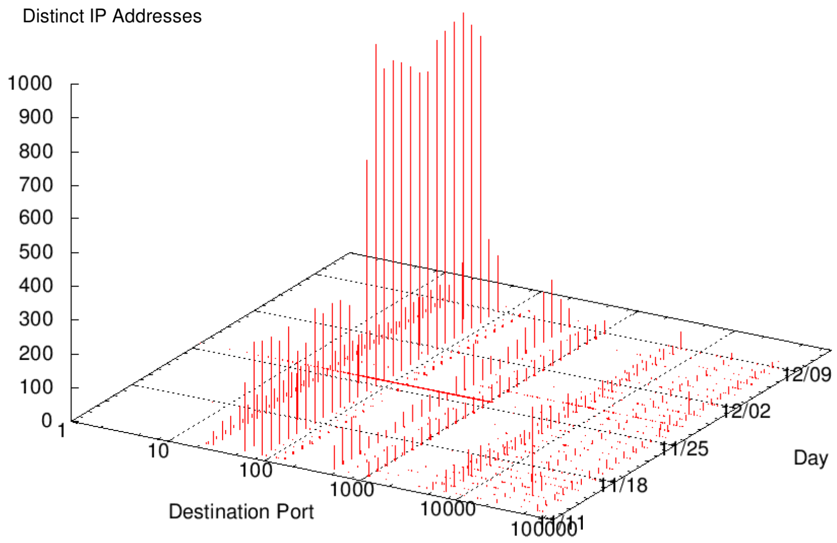


Figure 4.1: Impulse graph displaying the number of unique hosts seen probing a specific port (on any machine in the subnet) each day during the capture period [17].

4.2 Scatter plots

Scatter plots are one of the most frequently used and best understood graphs in security visualization. A scatter plot is a representation of a multivariate data set where the value of one variable is displayed on the horizontal axis, and the value of another variable is shown on the vertical axis. A symbol is drawn where the variable from a data set takes on these values. Scatter plots allow the user to quickly identify clusters and trends in data sets, and because there are many tools[1] that can easily generate these plots, they are often seen in security visualization.

Although scatter plots are frequently used to display two variables, it is possible to display more by using a 3D space. 3D scatter plots are useful for displaying how 3 variables interact, but due to 2D nature of monitors and paper, they often introduce usability problems, and require the user to actively interact (as opposed to passive analysis or analysis of printed or static images) with the visualization to uncover occluded data. Another way of displaying more than 2 variables in a scatter plot is by using colour or size encoding. Instead of displaying a simple dot or cross at the appropriate location, a different symbol (differing in size, colour, shape) can be used to display magnitude, frequency, etc.

Many possible combinations of data can be displayed using scatter plots in network security. As explained in Section 2.2, any number of data inputs can be used. Among the most popular combinations are displaying IP address (source or destination), port number (source or destination), time, and any pairing of these variables.

What follows is a list of example scatter plots that have been presented in security visualization literature

4.2.1 Source and Destination IP address and Destination Port

Scatter plots that display source and destination IP address and port numbers are seen very frequently in security visualization literature. These scatter plots show the aforementioned variables in pairs and may reveal interesting trends in activity. When time is added to the list of variables, time-series can be constructed to display the behaviour of a particular host or port over a given time-period. The most frequent application of these scatter plots is for scan detection, where the destination ports contacted are analyzed to

[1] Tools such as Gnuplot, Mathlab, R and spreadsheet programs can read CSV files and generate scatter plots.

find remote hosts attempting to discover services. Another use is for network profiling (determining what services a network offers). In Figure 4.2 we plot the destination port (TCP and UDP) of connections destined to a small university data set over a one-month period. The source IPv4 address of that connection is displayed on the x-axis, allowing the analyst to detect groups of hosts exhibiting similar behaviour. The resulting scatter plot displays a vertical line in the lower right quadrant which we later attributed to a vertical scan against the university network.

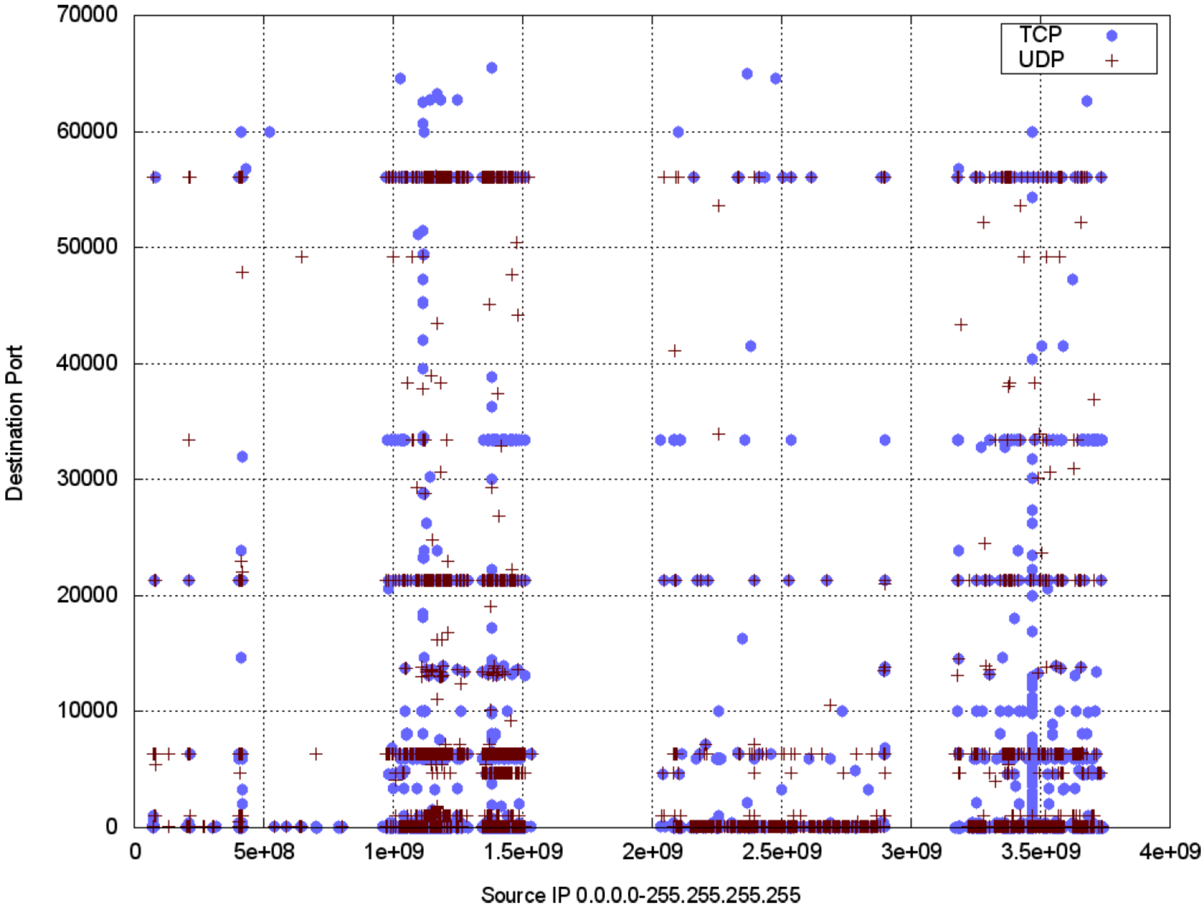


Figure 4.2: Scatter plot showing source IP and destination port [17].

Existence plots were recently proposed as a low-resolution time-series for port behaviour analysis [51]. Existence plots are 2D scatter plots that display time on the x axis and port number on the y axis. The plot graphs the port behaviour (the port number being used) of a single host over a fixed time window. Colour encoding is used to display the first, second and third quartiles of destination port activity. The box-and-whisker diagrams next to each existence plot displays this colour key (based on the number of bytes in each packet). These visualizations, while simple, provide a high-level overview of all the port activity of one particular host. Existence plots for many hosts (perhaps on the same subnet) can be aggregated (by plotting all ports contacted on the same existence plot) to display port information of an entire subnet. This information can be used to discover hidden services, unusual port activity, and scans. In Figure 4.3 the author of the existence plot tool shows 2 example scatter plots from a host that is performing scans on the network. The unique visual fingerprint (an image that uniquely identifies an event) of *port-cycling* (exhausting all ports in the ephemeral range[2] as a consequence of initiating many connections during a scan) is clearly shown in the upper area of both existence plots.

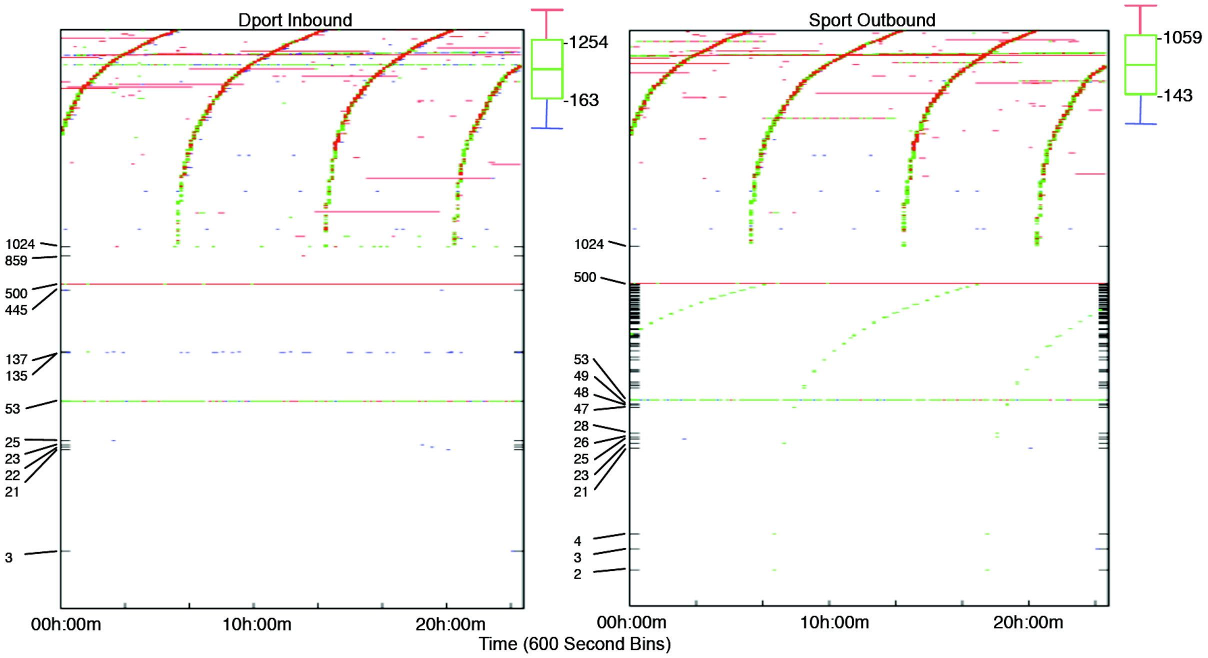


Figure 4.3: Existence plot of a host performing network scans [51]. The plot graphs source and destination port numbers vs. time. The y-axis extends to 65535 (though the highest label shown is 1024).

[2] The ephemeral port range refers to port numbers 1025-65535, which are used for outbound communications, and are generally unreserved by the operating system for establishing network connections.

4.2.2 IP Matrix

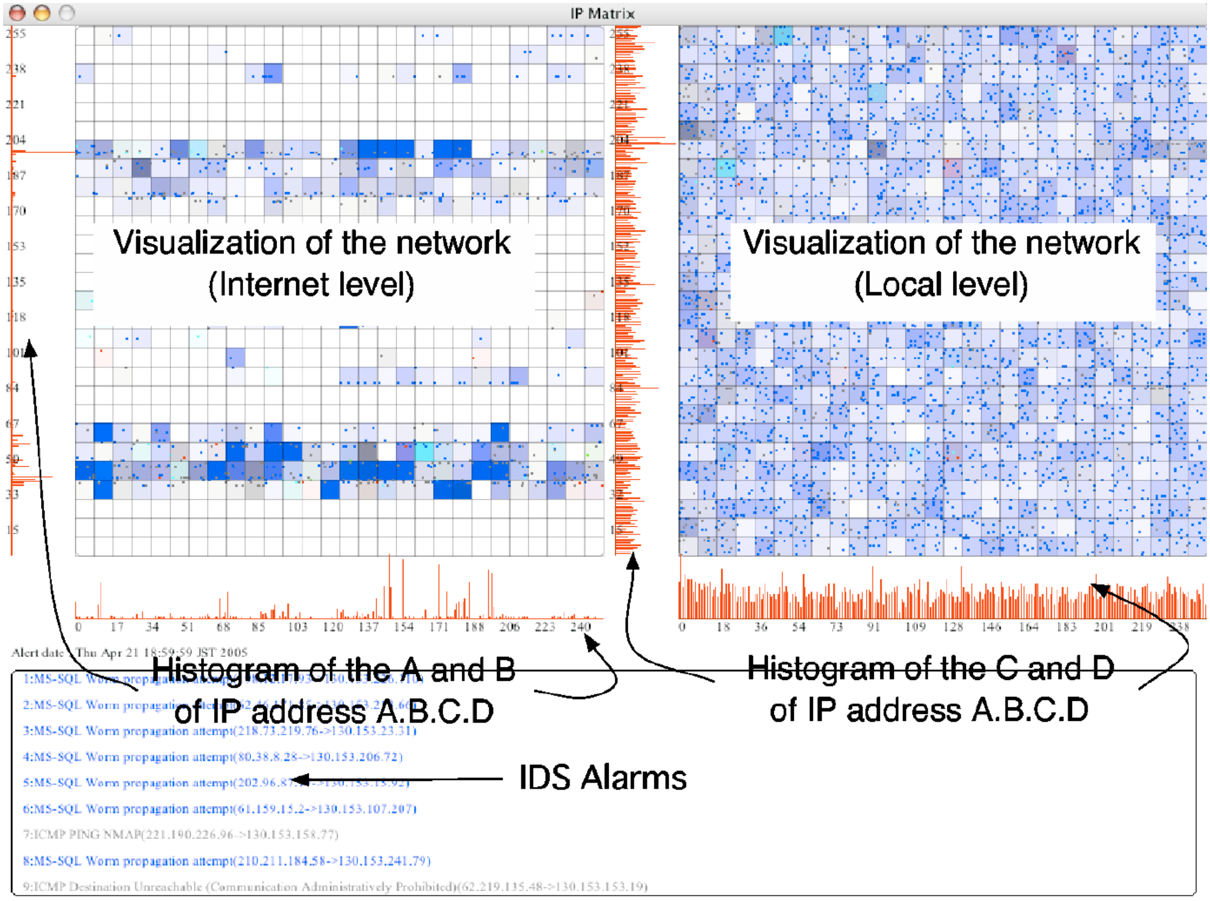
IP Matrices were proposed by Koike et al. [54]. They are constructed using two adjacent two-dimensional scatter plots that show netflow [24] data. One scatter plot represents the most significant 16 bits of the *source* IP address (**208.67**.29.84 for example), and the other scatter plot displays the least significant 16 bits of the destination IP address (192.168.**20.30** for example). One octet is displayed per axis. A point is shown at the intersection of both axes if a source at that address generated traffic. The IP Matrix tool allows for simultaneous displays of IDS alerts and as shown in Figure 4.4, which is a screen capture of the user interface. Histograms of host activity along the x and y axes, make it easier for the analyst to identify points of high activity. For example in Figure 4.4, there is high activity around source IP address 140.200.x.x (located by finding the value 140 on the x-axis and 200 on the y-axis of the left scatter plot), and also at destination IP address x.x.90.204.

There is no line connecting both scatter plots because the objective of the dual scatter plot display is to provide a compact representation of IP address space. This technique helps analysts gain situational awareness by allowing them to see sources at the Internet level, and also target machines at the host-level.

The IP Matrix technique provides good insight into specific host (either remote or local) behaviour, but screen real-estate can be inefficiently used if the IP addressing scheme is clustered in a small range. The technique might also not scale well on large networks (class A), since the host level scatter plot might be more dense than the Internet level.

4.2.3 NVisionIP

Lakkaraju et al. [55] present NVisionIP [55], a tool that uses scatter plots and simple bar charts to display host behaviour. The main window of the NVisionIP tool displays a 2D scatter plot showing network activity on a class B network (up to 65535 hosts). Similarly to the local level scatter plot of the IP Matrix tool, the x-axis displays bits 17


IP Matrix
Visualization of the network (Internet level)
Visualization of the network (Local level)
Histogram of the A and B of IP address A.B.C.D
Histogram of the C and D of IP address A.B.C.D
IDS Alarms
255
238
221
204
187
170
153
135
118
101
84
67
50
33
15
0
17
34
51
68
85
103
120
137
154
171
188
206
223
240
0
18
36
54
73
91
109
128
146
164
183
201
219
238

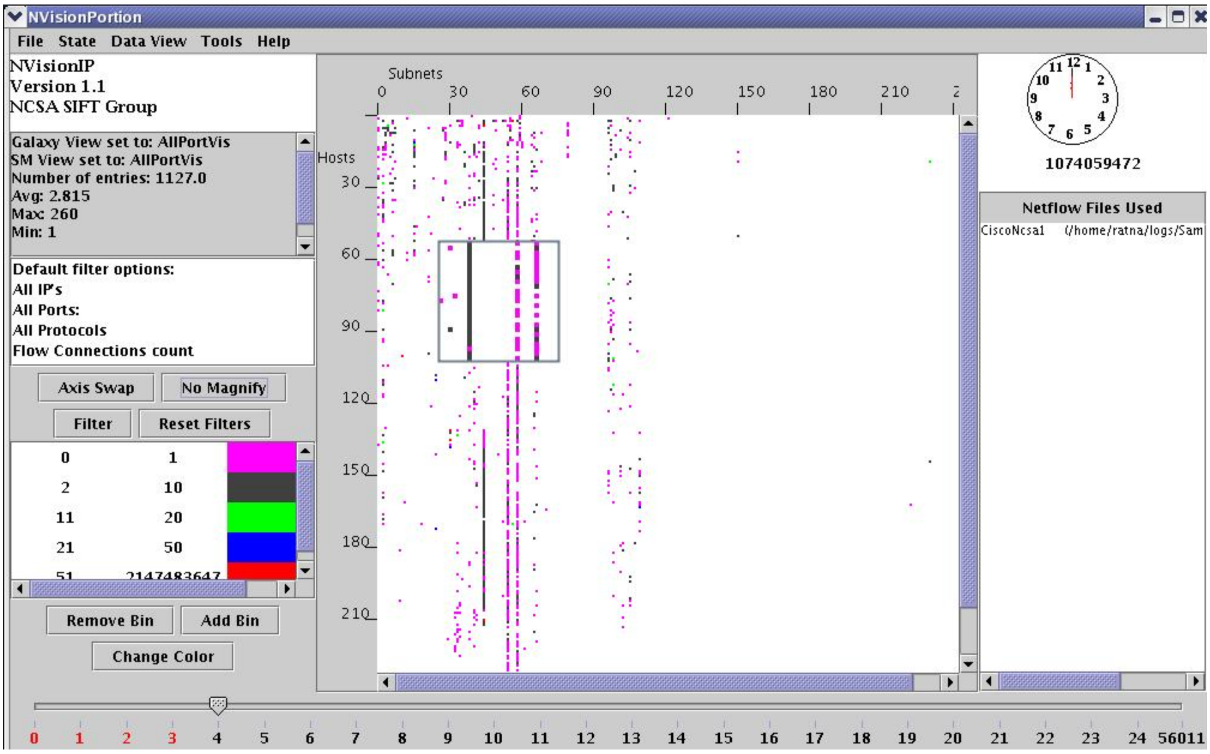
Alert date Thu Apr 21 18:59:59 JST 2005

1:MS-SQL Worm propagation attempt([illegible].17.93->130.153.2[illegible])
2:MS-SQL Worm propagation attempt(62.46.[illegible]->130.153.2[illegible].60)
3:MS-SQL Worm propagation attempt(218.73.219.76->130.153.23.31)
4:MS-SQL Worm propagation attempt(80.38.8.28->130.153.206.72)
5:MS-SQL Worm propagation attempt(202.96.87.[illegible]->130.153.15.92)
6:MS-SQL Worm propagation attempt(61.159.15.2->130.153.107.207)
7:ICMP PING NMAP(221.190.226.96->130.153.158.77)
8:MS-SQL Worm propagation attempt(210.211.184.58->130.153.241.79)
9:ICMP Destination Unreachable (Communication Administratively Prohibited)(62.219.135.48->130.153.153.19)

Figure 4.4: Screenshot of the IP Matrix visualization tool [54]. The most significant 8 bits of the IP address segment are shown on the x-axis in each scatter plot.

to 24 of the IP address (labeled as “subnet” in Figure 4.5), and the y-axis displays the least significant 8 bits of the IP address (bits 25 to 32, labeled as “host” in Figure 4.5). The scatter plot encodes port information by using user-configurable colours, and a basic method magnification to zoom into an area when the mouse is hovered over the scatter plot. The distortion (depicted in Figure 4.5) allows the analyst to dynamically zoom in, but raises some usability concerns due to the loss of perspective in relation to the rest of the plot.

One of the more interesting sides of NVisionIP is the use of small multiples view (SMV). Small multiples is a technique developed by Edward Tufte [78] in which a small number of similar visualizations are displayed next to each other in a 2x2 or 3x3 grid. This method helps the analyst compare activity from one host to another. In NVisionIP,



NVisionIP
Version 1.1
NCSA SIFT Group

Galaxy View set to: AllPortVis
SM View set to: AllPortVis
Number of entries: 1127.0
Avg: 2.815
Max: 260
Min: 1

Default filter options:
All IP's
All Ports:
All Protocols
Flow Connections count

0	1	🟪
2	10	⬛
11	20	🟩
21	50	🟦
51	2147483647	🟥

1074059472

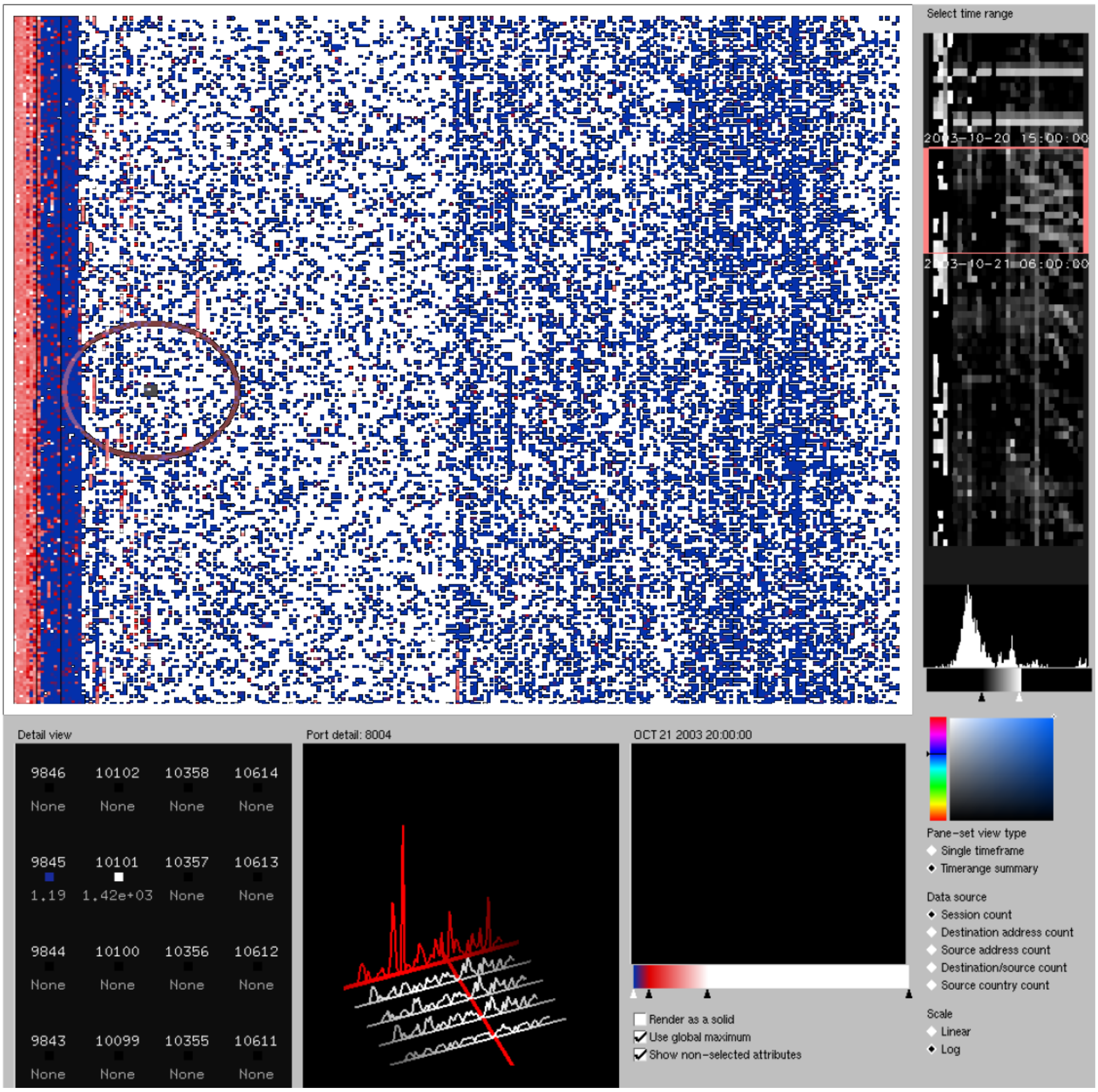
Netflow Files Used
CiscoNcsa1 (/home/ratna/logs/Sam

Figure 4.5: Screenshot of the NVisionIP tool displaying the mouse-hovering zoom feature [55].

the SMV displays bar graphs with port activity for each of the hosts selected from the main view (note that SMV is not shown in Figure 4.5). The ability to compare hosts provides analysts with context and allows them to make better judgements as to whether or not the behaviour is normal.

4.2.4 Portvis

Portvis [63] is a tool for analyzing netflow data with an emphasis on port behaviour visualization. The main window displays a large 2D scatter plot that graphs port numbers on the x-axis and IP addresses on the y-axis. The destination port of connections (attempts and completed) that took place over the capture period is graphed. Clicking on the scatter plot allows the operator to view the selected port activity in the tool's sidebars. Portvis is designed to work with netflow data (i.e., relying purely on IP/port information and not requiring payload information). The Portvis tool, while providing the analyst with a simple user interface, also follows the visualization mantra. Instead of moving between high-level and detailed views, the tool always displays the high-level overview scatter plot plus a detailed view on the sidebars. Figure 4.6 shows a screenshot of the tool, displaying the main scatter plot view and the surrounding detailed statistics. The bottom of the image shows detailed port *hit counts* (how many times a port has been probed or accessed), along with a time-series of port activity for a selected port (port 8004 in the example).



Select time range

2003-10-20 15:00:00

2003-10-21 06:00:00

Detail view

9846	10102	10358	10614
None	None	None	None
9845	10101	10357	10613
1.19	1.42e+03	None	None
9844	10100	10356	10612
None	None	None	None
9843	10099	10355	10611
None	None	None	None

Port detail: 8004

OCT 21 2003 20:00:00

☐ Render as a solid
☑ Use global maximum
☑ Show non-selected attributes

Pane-set view type
○ Single timeframe
● Timerange summary

Data source
● Session count
○ Destination address count
○ Source address count
○ Destination/source count
○ Source country count

Scale
○ Linear
● Log

Figure 4.6: Screenshot of the Portvis tool [63].

4.2.5 3D Scatter Plots

3D scatter plots provide an extra dimension which can help display a third variable such as time. While it seems like this would always help (i.e., the analyst would get to see more data at once), this isn't always the case. Usually we interact with visualizations on a 2D display (screen, paper); thus in order to view a 3D visualization, it must be projected down to its 2D equivalent. This creates distortion, disorientation, and occlusion. If a tool or display is easy to navigate through, then there is indeed value in representing more dimensions on the same display, however, empirical evidence suggests that displaying a third variable is more helpful by using other encodings such as colour or size rather than a third dimension.

Spinning Cube of Potential Doom

One of the most popular 3D scatter plots is the Spinning Cube of Potential Doom [56]. Figure 4.7 displays a 3D visualization similar to the spinning cube of potential doom that shows the source IP on the x axis, the target *host IP* on the y axis, and the destination port on the z axis. Points are drawn at the intersection of axes for a given connection attempt (i.e., what remote address contacted what local host on what port). In the example we see a large number of diagonal solid lines, indicating that a large number of single source IP addresses performed port scans on all destination hosts in the subnet. Figure 4.7 was generated from a one-month university data set on a class C network. Notice that it is difficult to identify connections, especially on low-numbered ports, due to occlusion. To overcome occlusion, the analyst must interact with visualization (i.e., *spin* the cube). To remove remove additional work from the analyst, 2D static projections of 3D graphs are preferred.

The Contact Surface

The *contact surface* is a technique by Gates et al. [36] is a 3D scatter plot that attempts to answer the question: how many unique source IP addresses x contact unique destination

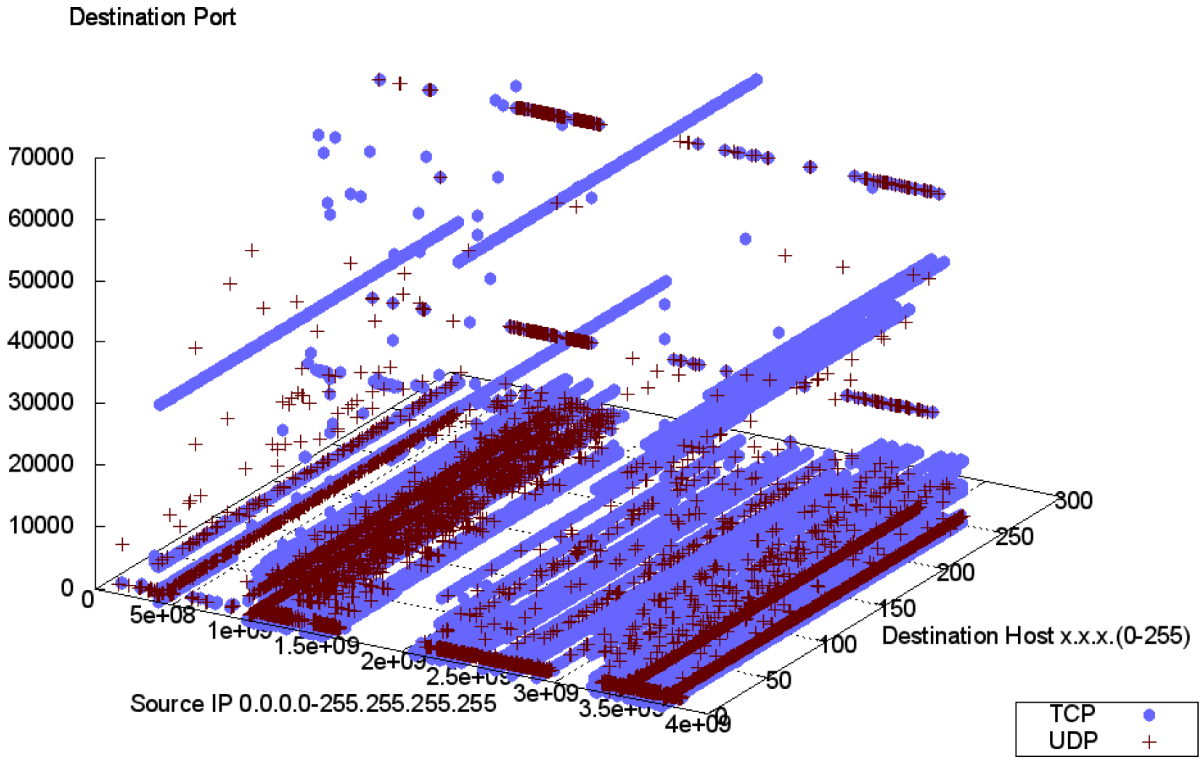


Figure 4.7: 3D visualization similar to the spinning cube of potential doom [17].

services or hosts y. This type of graph is intended to help the analyst visualize behaviour rather than raw data (see Section 4.1.1), and may take a long time to compute particularly on large data sets. The resulting graph (generated from a one-year capture on multiple class C networks), shown in Figure 4.8 shows an interesting powerlaw-like distribution where a large number of sources (consistently around 100,000) connect to a small number of services. This is expected behaviour, as end-user devices generally connect to a small set of servers. On the other side of the graph, we see a small number of sources connecting to many services. These hosts are generally infected with scanning worms, or they are intentionally malicious. The contact surface can be useful for discovering Internet-wide attacks if netflow data is captured on a link that has enough traffic.

4.2.6 Sparklines

Sparklines [2] were invented by Edward Tufte and are used to encode large amounts of information on limited resolution displays. The idea is that absolute numeric values are

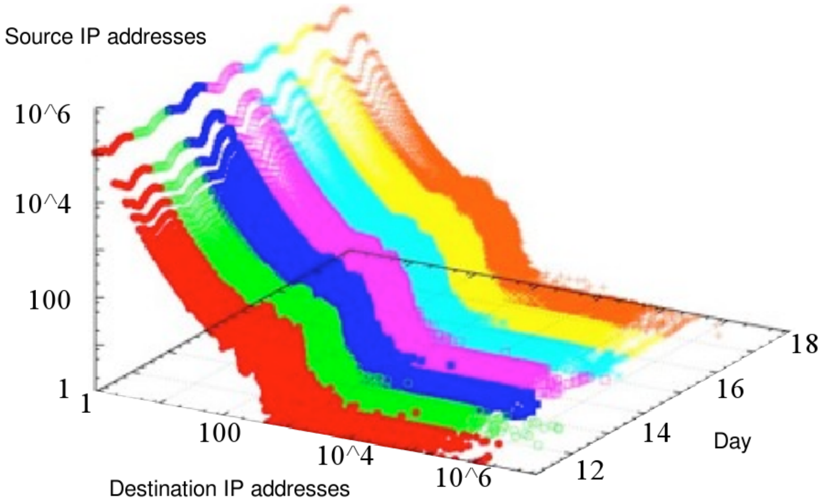


Figure 4.8: The Contact Surface graph [36].

not important, so the focus is on analyzing trends in a particular data set. A sparkline is a 2D line chart that displays the behaviour of a variable over time. The mean and standard deviation can easily be encoded, giving the analyst a sense of whether or not the current values are expected. One classic use of sparklines is a stock market ticker. Due to their compact representation, sparklines can easily be added to other techniques to provide historical port behaviour [62]. In Figure 4.9, we show 4 sparklines displaying the number of connections to specific ports. The average number of connections is shown in gray.

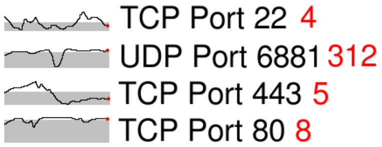


Figure 4.9: Example Sparklines.

4.3 Link Graphs

Link graph visualization techniques attempt to show how 2 or more hosts or events are related by drawing lines to connect them. We emphasize the use of the term graph in the mathematical context (i.e., the structure that shows pairwise relations between vertices or nodes by connecting them with edges), rather than the abbreviation for the term graphic. Computer network traffic flows between hosts, so using the graph metaphor seems appropriate to visually represent how data is transferred between hosts.

Some of the visualizations presented in this section are not explicitly link graphs, but use the same principle of connecting points to display how variables are connected. It's important to notice the difference between the connections used in graphs compared to lines drawn in a line chart. A line chart will generally show only how a single variable is changing over time (trends) rather than showing how two variables are related.

4.3.1 Classic Link Graphs

Classic link graphs are constructed from nodes and edges that connect nodes. A *directed graph* is a type of graph that shows the direction of the connection (through the use of arrows on the edges) as opposed to an undirected graph, where direction information is omitted. Link graphs are useful for displaying relationship among individual elements and are sometimes referred to as *semantic graphs, relationship graphs* or *link maps.*

In the context of network security, link graphs generally display at least 2 variables: an event or resource, and the connection between them. More dimensions can be added into a link graph by using colour, shape and edge thickness. If there are too many nodes to display, then graph layout algorithms [29] become important to prevent occlusion and overlap in resulting graphs. Some graph display tools allow the user to selectively group nodes to maximize screen real-estate. It is common to see this aggregation of hosts based on subnet or type of device (server, client, phone, etc.).

Becker et al. [19] introduced a basic graph representation for network-enabled hosts using a directed graph where the edges are labeled according to the number of packets sent over that link. They also show an alternate representation where sent and received packets can be displayed on one edge by dividing it into 2 segments (transmit and receive).

Erbacher et al. [31] use link graphs slightly differently to visualize IDS logs. They use several classes of edges (dotted lines, multiple arrows, etc.) to convey the type of traffic that is flowing between hosts. Nodes use special glyphs to encode the number of active connections at that particular host at a given time. Finally, colour helps the analyst identify which connection attempts generated alerts and which ones were allowed. An example of Erbacher's visualization technique is shown in Figure 4.10, where each host is represented by an angular spoke.

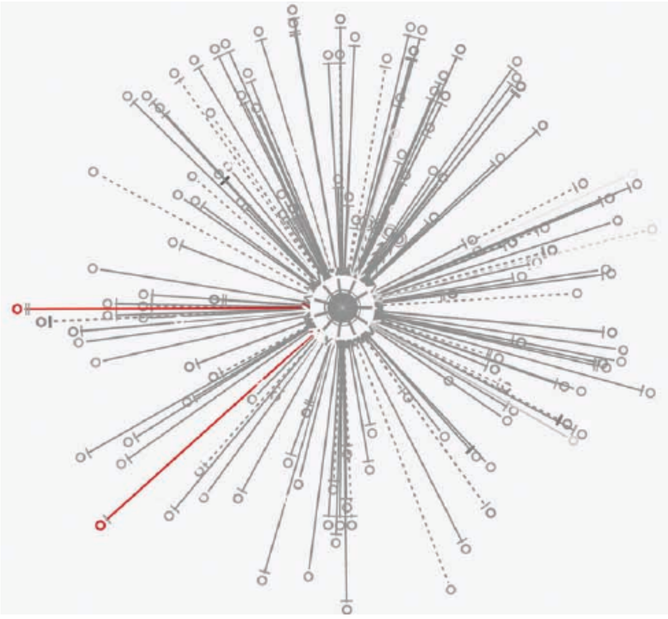


Figure 4.10: IDS log visualization using glyphs [31].

4.3.2 3D Link Graphs

The same way that scatter plots can be created in 3D, it is possible to generate three-dimensional link graphs. The advantages of displaying a link graph in 3D are less obvious than doing the same for a scatter plot, since the third dimension does not provide an extra variable. The actual advantage is in the layout of nodes, since there is a new plane to move nodes on. However, occlusion once again becomes an issue, and the user is forced to interact with the visualization to display areas that are hidden by other points.

The Cooperative Association for Internet Data Analysis (CAIDA, an organization devoted to measuring important characteristics of the Internet) released a Java tool called Walrus [15] which is capable of rendering, in 3D, link graphs with thousands of nodes, and has been used to visualize the spread of the CodeRed [66] worm. Figure 4.11 visualizes the 24.0.0.0/8 subnet, which is primarily made up of home broadband customers, and a small number of business customers. Using visualizations like these, CAIDA was able to spot subnets that were heavily infected. While the 3D link graph is impressive visually, the literature contains very limited discussion as to the benefits of displaying data this way.

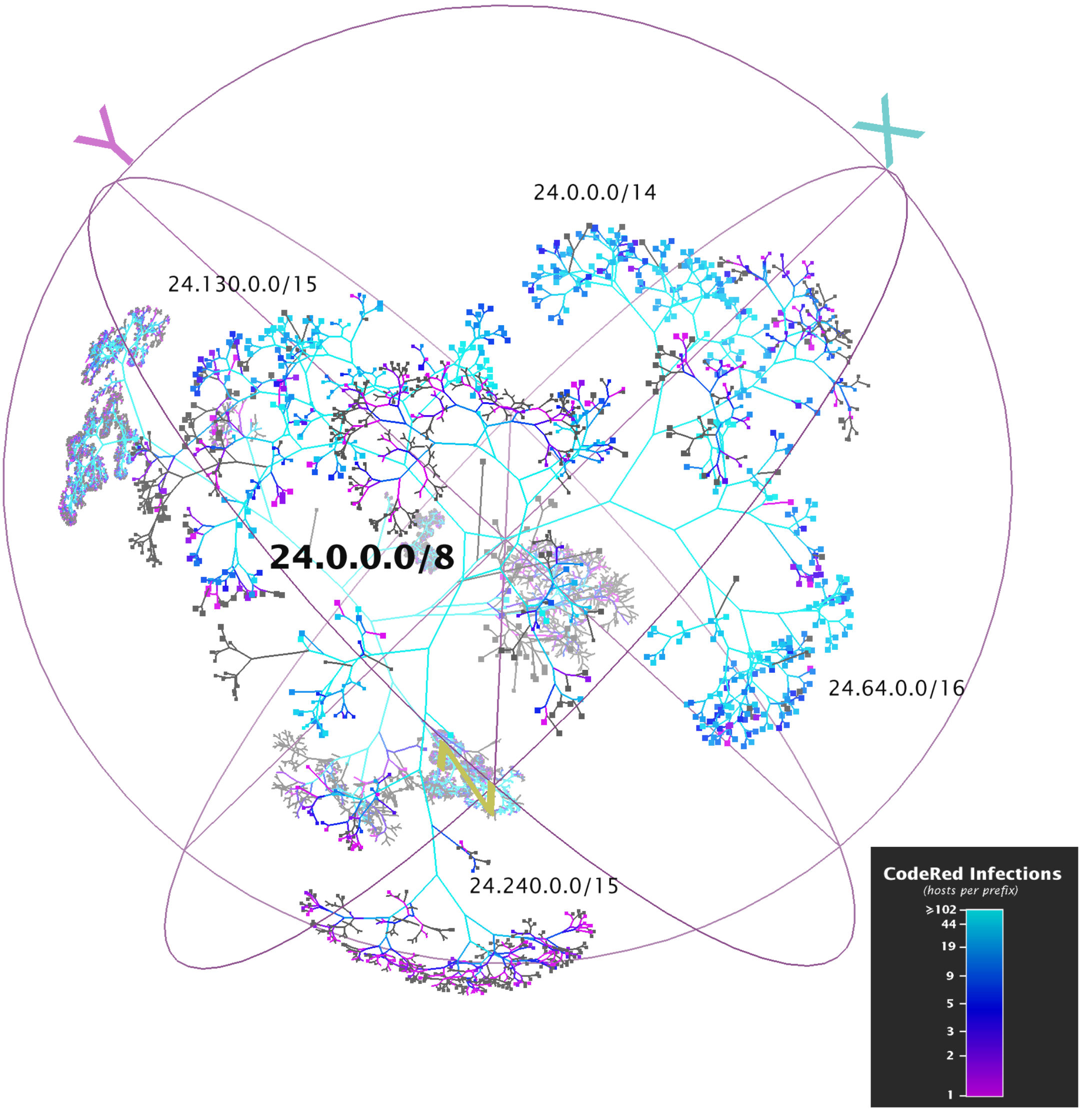


Figure 4.11: 3D Link graph showing the spread of the CodeRed worm [15].

Oberheide et al. [69] developed the Flamingo tool for visualizing Border Gateway Protocol (BGP) traffic between ISPs. The visualization uses the familiar 3D cube display, where internal faces are divided using quadtree coding (see Section 4.4.2) to represent the full 32-bit IPv4 address space. In addition, Flamingo can visualize in real-time network traffic to specific ports, where servers hosting services are represented as dots (or nodes) inside the 3D cube (shown in Figure 4.12). Lines connect servers and clients, creating a 3D link graph. The authors mention that having the ability to navigate through the cube is crucial in order to minimize information loss due to overlap and occlusion.

There has also been some work done in combining 2D and 3D visualizations by Le Malecot et al. [58] who use an interactive 3D link graph visualization on the left side of their tool to give the analyst an overview of network traffic, then use a 2D scatter plot on the right side to visualize selected (by clicking on an area of interest) activity. The tool can also work in reverse, by selecting an area of interest in the 2D display which will have an effect on the 3D visualization.

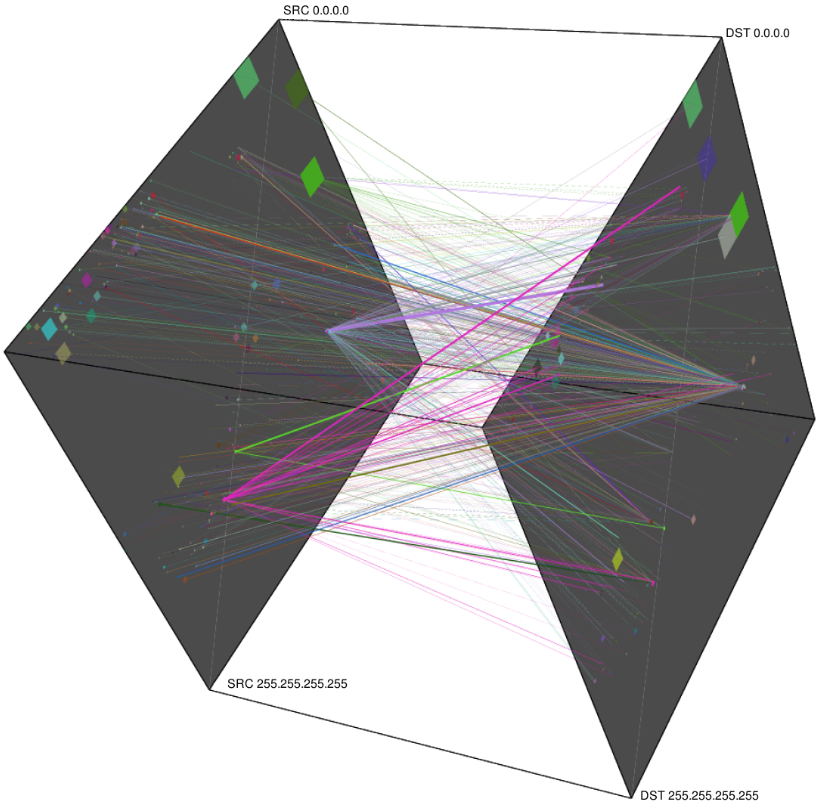


Figure 4.12: Screenshot of the Flamingo tool for visualization of BGP traffic [69].

4.3.3 Parallel Coordinate Plots

Parallel coordinate plots [79] are 2D visualizations that share some of the properties of link graphs. They are used for representing highly dimensional or multivariate data. Parallel axes are drawn on the display, each representing one variable (minimum value at the top, maximum at the bottom or vice-versa). Visualizing a new variable only requires the insertion of another parallel axis. The key element of parallel coordinate plots is that lines are drawn connecting each one of the axes for that particular datapoint. For instance, if an entry in a data set contained values 3, 5, 3, 7 and 9, then the parallel coordinate plot would have 5 axes, and a line connecting point 3 on the first axis, to 5 on the second, etc.

Parallel coordinate plots are used frequently in security visualization because the data being analyzed generally has multiple variables (a netflow record has at least 5 variables, a packet or log entry has multiple fields). Adding a new axis also has the advantage of allowing for the display of a new variable and this can be done essentially infinitely (provided the screen is wide enough, or has scrolling capability).

It is important to note that parallel coordinate plots display how variables are related, so the organization of axes is important. A potential insight might be lost if unrelated variables are placed next to each other, and overlapping lines between axes make it difficult to spot trends.

We have implemented a basic parallel coordinate plotting tool for displaying the source and destination ports and IP addresses of network packets as recorded off the wire (see Figure 4.13). Our implementation of parallel coordinate plots improves upon alternatives such as Picviz[3] and Rumint[4] by displaying the first 1024 ports (the reserved port range) in the top half of the port axis. Ports 1025-65535 are shown in the lower half, allowing the analyst to visualize common activity on the lower range ports without having to zoom in due to scaling problems. The example shows 4 axes representing source IP address,

[3] http://www.wallinfire.net/picviz

[4] http://rumint.org

source port, destination port and destination IP address in that order. We selected this particular order to display how source and destination ports interact. We can learn from the image that the network being analyzed has mostly bi-directional traffic (TCP) by looking at the symmetry between source and destination ports. We also note that there is some occlusion in Figure 4.13 due to the use of a primitive labeling algorithm.

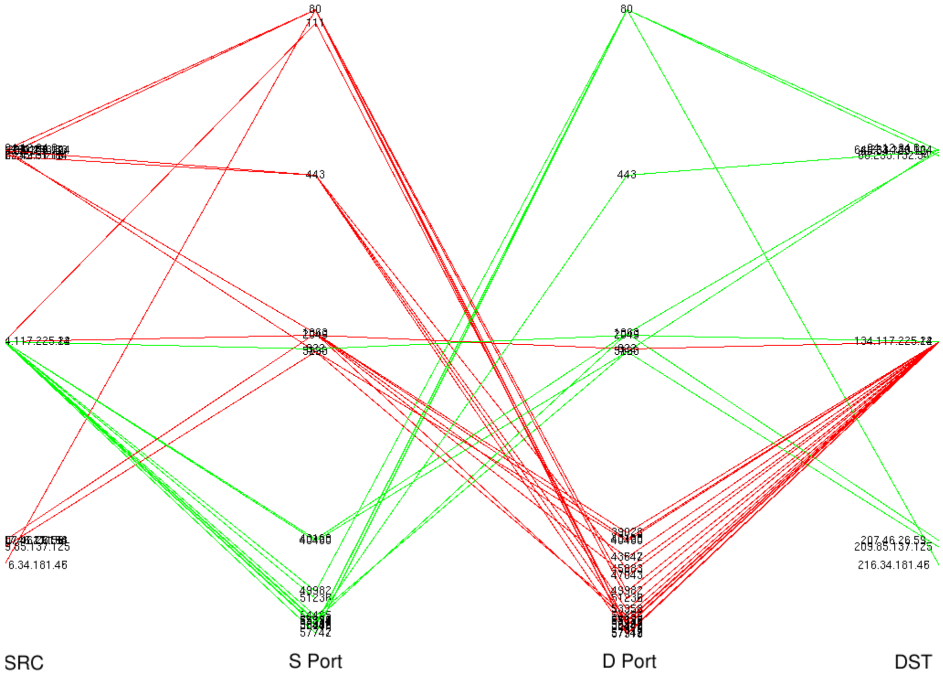


Figure 4.13: Screenshot of our implementation of parallel coordinate plots. The order of axes is: source IP address, source port, destination port, and destination IP address. Best viewed in colour.

4.3.4 Starplots

Starplots [23] are another type of link graph used to display multivariate data similarly to parallel coordinate plots, with the difference being the placement of the axes. A starplot (as shown in Figure 4.14) consists of a group of angular spokes, where each spoke represents one variable, and may be scale-independent from the other spokes (i.e., one spoke may represent a 16-bit destination port, and another may represent a 32-bit IP address). Lines are drawn connecting each spoke, generating a star-like appearance. Starplots and parallel coordinate plots are similar in that they display variables of different types on each axis and also how the variables are related (by drawing a connecting line). The difference (besides the obvious placement of the axes) is the number of points per axis. Starplots show a single data point per axis, whereas parallel coordinate plots can show more.

Because starplots show only minimal information for a single event or packet, it is difficult to obtain relevant information from the starplot exclusively. Therefore, the small multiples view (discussed in Section 4.2.3) is used in conjunction with starplots. In small multiples, a 3x3 or 4x4 grid is displayed, which each cell showing a starplot. The idea is that a human can then detect similarities and differences between items on the grid. Starplots are used primarily for comparing a set of events and identifying similarities among them, which is why numeric labels are omitted.

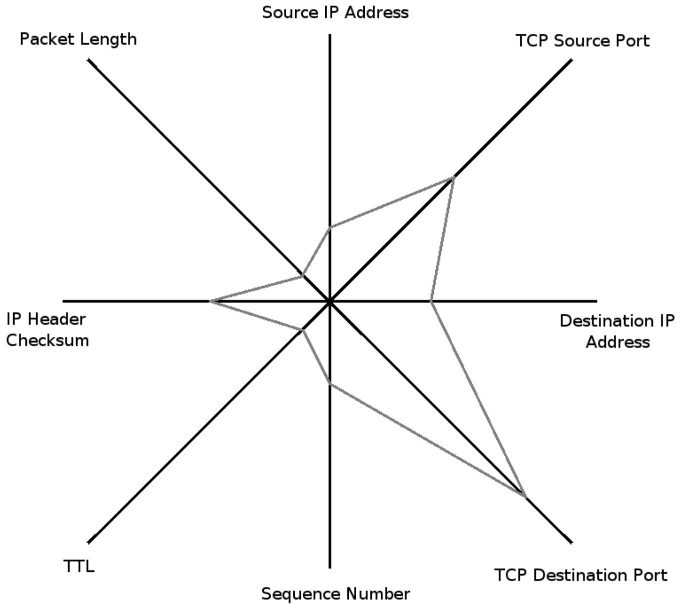
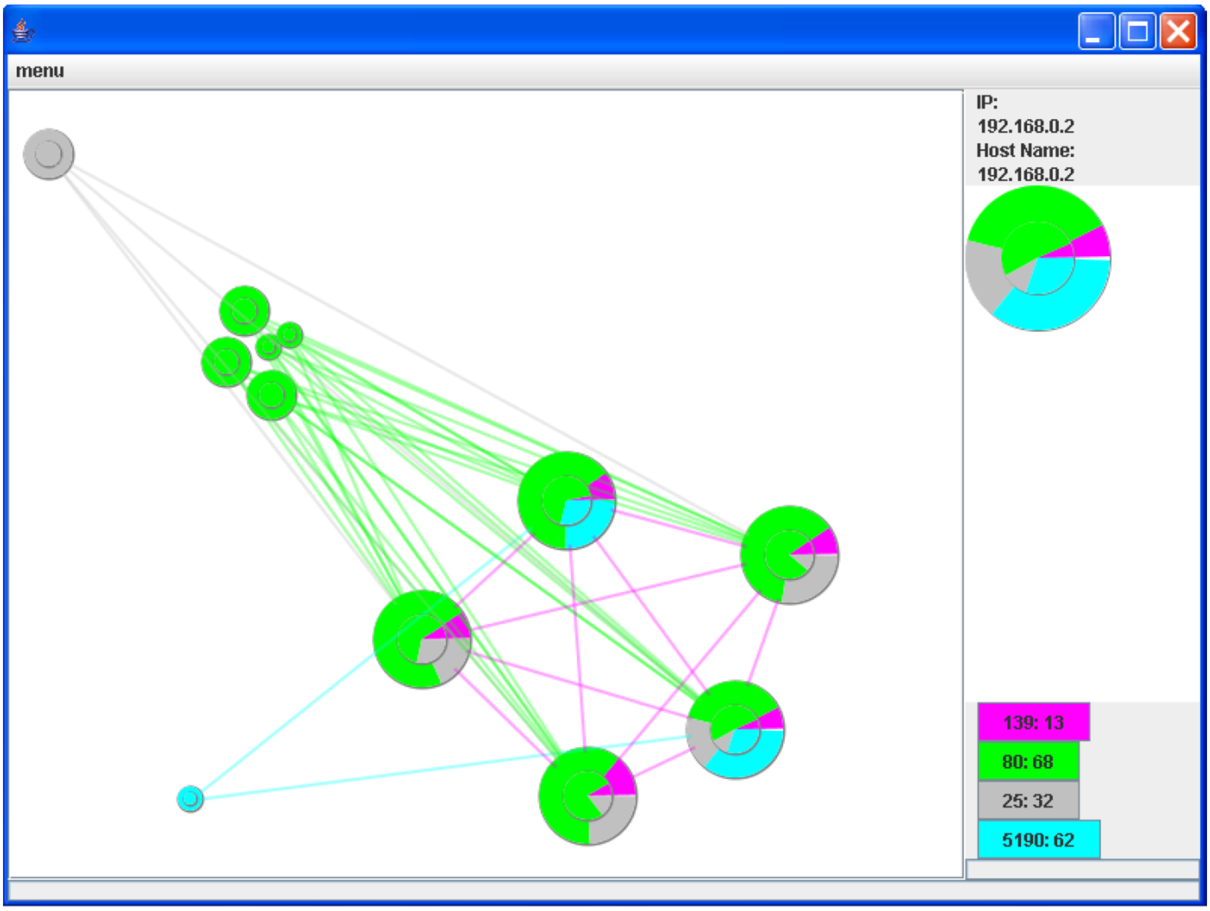


Figure 4.14: A starplot displaying packet header values

4.3.5 Compound Glyphs

Pearlman et al. [72] use link graphs to visualize network activity, but develop a glyph display consisting of a circle divided into 4 sub-circles. Each node in the network is assigned one of these circles which will reflect the amount and type of activity being transmitted by that particular node. Colour and size encode volume and type of activity. Temporal information is added to the link graphs by varying the opacity of nodes as connections become outdated or stale. An administrator can thus identify recent connections easily. Figure 4.15 shows a screen capture of the glyph based IDS visualization tool being run on a real network. In the example, the authors point out that nodes are clustered together based on IP address, and the tool is capable of fading out nodes selectively to help focus the analysts attention to specific parts of the graph.



IP:
192.168.0.2
Host Name:
192.168.0.2

Figure 4.15: Screenshot displaying a glyph-based link graph from a real network [72].

4.3.6 FloVis

FloVis [76] is a flow visualization system developed by Taylor et al. to visualize netflow data in different ways, with the hope that multiple representations of the same data set will provide more insight to the analyst. The system is also modular so that once the

Flovis tool is publicly released, the plan is that users will be able to contribute their own visualizations, and they can be plugged in to the application. Since Flovis is a complex visualization suite for network security administrators, it is difficult to classify it into just one of our categories. However, two of its three currently available visualization modes are similar to scatter plots mentioned in Section 4.2. The remaining visualization known as the *bundle diagram* (see Figure 4.16) is a link graph.

One of the FloVis visualization modes is the *Activity plot*, essentially a scatter plot displaying IP addresses and time, using colour encoding to show what type of host was seen transmitting data. If the host was seen only *responding* to connections, it is classified as a server. If it seen only *initiating* connections, then it is classified as a client. If it is both initiating and responding then it is labeled as both client and server.

Another visualization mode currently implemented in Flovis is the *NetBytes viewer*, which displays port activity over time. This display mode uses a 3D impulse graph showing individual ports and how many connections to that port were seen on a given day. In contrast to the sparklines method, the NetBytes viewer displays labeled values at each date.

An important contribution of Flovis is the application of *edge bundling* [46]. Edge bundling algorithms group edges as they traverse a large portion of an image, and then ungroup them as they reach the endpoints. FloVis uses a circular display where the lower semi-circle represents the 32-bit source IP address, and the upper semi-circle the destination IP address. Values are increased as points move across the arc of the semi-circle. Lines are drawn between the 2 points (one source and one destination, each on different semi-circles) representing the hosts that were seen communicating. Concentric circles help identify "branch-off" points which tend to be subnets where many hosts display the same type of activity. Figure 4.16 shows an example where a single host (in the 128.0.0.0 network) is seen scanning the entire destination subnet. Notice how even though there are a large number of connections, the center of the display is not cluttered since multiple edges were bundled.

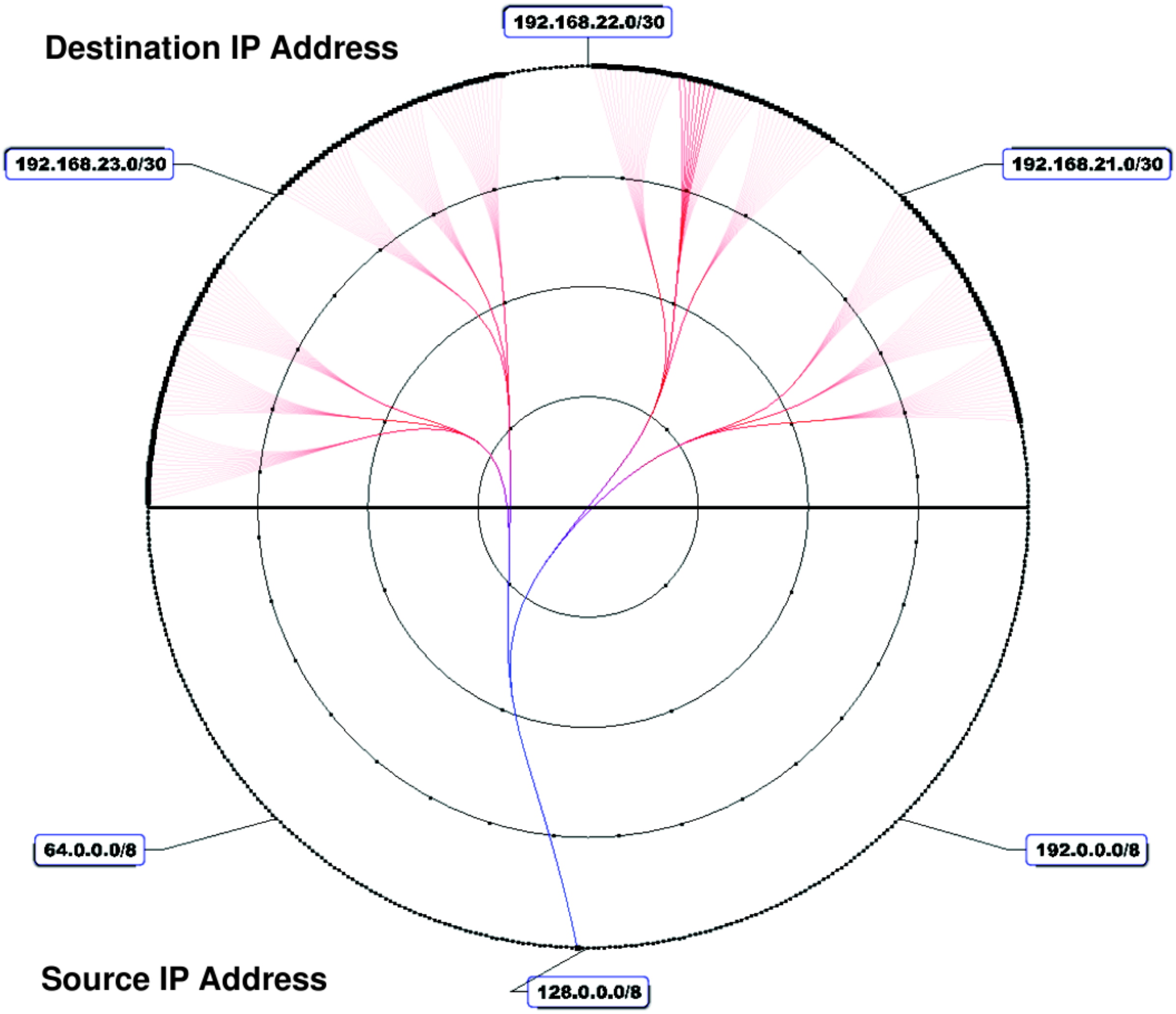


Figure 4.16: Screenshot of FloVis edge bundling [76].

4.4 Geographical and Space-filling Maps

A *map* is a visual representation of an area. The word “map” is commonly used in the context of cartography, and while there are visualizations that utilize the geographical definition of maps, there are also non-geographical maps which can provide great insight for network security data sets. In this section we survey both geographical and space-filling maps to represent network security data.

4.4.1 Treemaps

Treemaps [74] are a 2D space-filling technique that use nested rectangles to display the hierarchical structure of a data set combined with volume information. Treemaps are useful for visualizing hierarchical data, and due to the space-filling algorithm, they tend to work well on displays of all sizes. Treemap implementations frequently have dynamic zoom so that the user can select a node and redraw the treemap using it as the root. They also allow for the display of large data sets, and help identify patterns between different segments of a hierarchical data set.

Treemaps require a *tiling algorithm* that will handle the way in which rectangles are drawn and placed on the screen. There are 5 popular tiling techniques (binary tree, ordered, slice and dice, squarified and strip) each of which will produce similar results (i.e., a nested set of 2D rectangles) but with slightly different aspect ratios for the rectangles.[5] It is common to toggle between the different tiling techniques to discover which one generates the "best" treemap for a particular data set.

Treemaps can also use colour to encode additional information. In the Linux tool Kdirstat,[6] the filesystem is represented as a treemap where the hierarchy is the directory structure. Each file shows up as a rectangle that is proportional in size to the amount of space it consumes in the filesystem. The colour of each rectangle (or node) will depend on the file type so the user can easily spot common file types such as images or videos. In network security, treemaps have been used to visualize firewall logs [5] and to visualize the structure of IPv6 addresses [18] (see further discussion in Chapter 6). Fischer et al. [34] also use treemaps to visualize large-scale network attacks, but extend the treemap technique by overlaying a link graph and using edge bundling [46] to connect nodes to the treemap.

Figure 4.17 displays a treemap of firewall logs obtained from the online security visualization discussion group SecViz.org. The treemap shows a three level hierarchy, where the

[5] Some tiling algorithms apply more preference towards the relative size of a rectangle compared to other rectangles, and other algorithms help visualize hierarchy.

[6] http://kdirstat.sourceforge.net

top level represents hosts which are being monitored. The second level shows the sources of the connections, and the third level displays the port number, with colour-encoding used to distinguish successful and dropped packets.

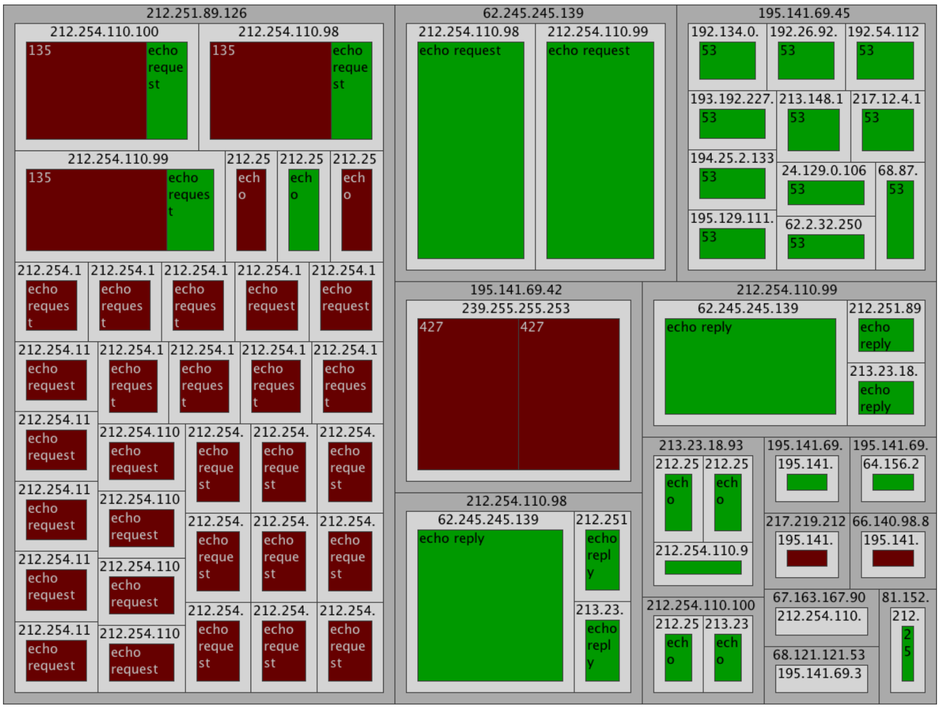


Figure 4.17: Treemap of firewall logs obtained from the online security visualization discussion group SecViz.org [5].

4.4.2 Quadtree Maps

Quadtree maps [33] are another form of 2D space-filling visualizations in which each node has up to 4 children. The main area, generally a large square, is divided into 4 quadrants (00, 01, 10 and 11), and then each of those subsections can be further divided into 4 more quadrants (see Figure 4.18). This visualization gives priority to the high-order bits in a binary structure, such as an IP address, because the high-order bits will be placed in larger squares than low order bits. For example, an IPv4 address is 32-bits long, so using a quadtree that has 16 iterations of splitting (4 quadrants = 2 bits per iteration) is enough to display it.

Teoh et al. [77] use quadtrees for visualizing the source and destination IPv4 addresses in BGP updates. In their implementation (see Figure 4.18), 4 lines surrounding the quadtree represent autonomous system (AS) numbers (unique numbers assigned to large enterprises or Internet service providers). The quadtree itself represents IPv4 addresses, and diagonal (coloured) lines connecting pairs of AS-IP addresses on the quadtree map show which IP addresses belong to which AS. This visualization is intended to allow the analyst to perform basic visual anomaly detection by looking for uncommon connections between AS numbers and IP addresses.

Another example of using quadtrees in security visualization is described in Section 5.4.1, where Le Malecot et al. [57] use quadtrees to display the source IP addresses associated with an SSH brute-force attack.

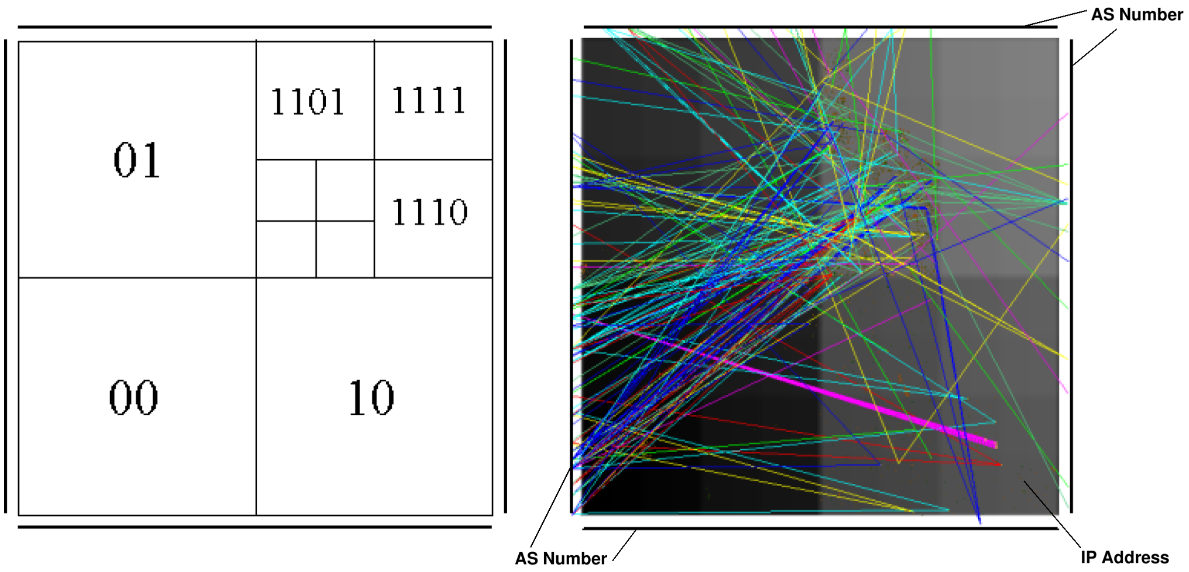


Figure 4.18: Quadtree layout and example visualization showing connectivity between autonomous systems and IPv4 addresses [77].

4.4.3 Geographical Maps

Because (wired) network devices usually have a static physical location, geographical maps are a useful way of visually conveying this information to the user. The displayed information on a map can range from representing the device's continent, country, city, building, floor, or even rack. The most common type is a world map display where a particular event is shown geographically.

There are many IP to geolocation [7] services available both commercially and publicly available (free) that will provide country and city information based on an IP address. The accuracy of these services is generally high in detecting the last hop of a connection, but this does not necessarily imply that the last hop is the true originator of the connection [67].

For geographical maps, information can be encoded as a point or circle over a particular city, or as a *heatmap* where a colour gradient is overlayed on each country or city. The intensity of the colour in an area reflects the number of events observed in that particular point.

Geographical maps, by definition, will only convey geographical information and therefore their application is limited. Becker et al. [19] were among the first to publish maps of Internet backbone connectivity within the United States.

4.5 Summary

This chapter has reviewed a wide range of publicly available visualization tools and techniques. Commercial tools exist as well, but these were not analyzed or included in our survey. Other more simple techniques such as *pie charts* and *bar graphs* did not have enough applicability to security visualization to be included in the survey, but they do have potential use in some specific areas.

Table 4.1 presents a summary of visualization techniques grouped by the types presented in our classification, and lists features of each technique. This table will be used for pairing visualization techniques with the classes of network attacks in Chapter 5.

Class of Graph	**Technique or tool**	**Number of independent variables displayed**	**Potential occlusion issues**	**Level of user interaction needed**	**Handles large data sets**
Scatter Plots	Src. IP addr., Dst. IP addr., SPort, DPort [51, 17]	1-4	Normal	Low	✓
	IP Matrix [54]	1-4	Normal	Low	✓
	NvisionIP [55]	1-4	Normal	Medium	✓
	Portvis [63]	1-4	Normal	Low	✓
	3D Scatter Plots [56, 36]	1-4	Normal	Low	✓
	Sparklines [2]	1	-	-	-
Link Graphs	Classic link graphs [19, 31]	2-4	Normal*	-	✓
	3D link graphs [15, 69, 58]	2-4	High	High	✓
	Parallel coordinates [79]	> 10	Normal	Low	✓
	Starplots [23]	> 8	-	-	-
	Compound Glyphs [72]	1-8	Normal*	Low	✓
	Flow Bundle [76]	1-4	Normal*	Low	✓
Geo. and space-filling maps	Treemaps [74]	4-6	Low	Low	✓
	Quadtree maps [33]	1-4	Normal	-	✓
	Geo. maps [19]	1-4	Low	-	✓

Table 4.1: Summary table of visualization techniques and features. (*) denotes high occlusion on large data sets.

Chapter 5

Visualization Technique Selection

5.1 Introduction

In this chapter we use a task-based evaluation methodology to apply visualization techniques from Chapter 4 to the classes of network attacks described in Chapter 3. We also apply specific techniques to example attacks as a case study and aim to provide an objective claim regarding the insight gained by using a technique for the visualization of a network attack. Finally we suggest a novel visualization technique to visualize DoS attacks which we call the *grid heatmap* (Section 5.3.1), and propose a new use of two existing visualization techniques for visualizing brute-force password guessing attacks (Section 5.4.1).

5.1.1 What Are We Looking For?

When using visualization to understand data sets the analyst poses the question: "What are we looking for?" If the analyst knew what he was looking for, then typically he would not need visualization to find it; a set of direct queries on the raw data would suffice. As discussed in Section 2.1, one of the most important motivations for using visual techniques is to discover trends or patterns in data that were not obvious in a text-only environment. This reasoning may lead to the incorrect conclusion that the most insight can be gained when a data set is visualized in *all* possible ways, giving the analyst a better chance to discover something interesting. While it is in fact useful to visualize data in different ways (for example using complementary visualizations as described in Section 2.1.4), there are certain visualization techniques that by definition will not provide insight in specific cases, e.g., using a geographical map to visualize TCP sequence numbers. Geographical information cannot be obtained from sequence numbers, therefore using a map visualization to represent them will not help the analyst arrive at any useful conclusion about the data set.

That said, it is possible for the experienced analyst to select visualization techniques based on his own intuition regarding the situation or the attack. While the analyst may not know exactly what he is looking for, he may have an idea on how to increase his odds of finding interesting data. This is part of the reason why the experience of the analyst plays a critical role in successful security visualization.

We have already identified some of the key variables for the phases of network monitoring in Chapter 3. We now select visualization techniques which are most capable of displaying those variables clearly and intended to help guide the analyst in discovering previously unknown threats.

5.1.2 Factors to Consider When Selecting a Visualization Technique

At a high level, there are certain factors to take into account for selecting the best visualization technique for a particular task. Some types of graphs help display how variables are related, while other techniques show the absolute value of variables or trends in data. Still other techniques might be better for detecting clusters or to identify outliers in the data set. There are also visualization techniques that can display many dimensions at once, while some can display at most 2 or 3 independent variables. These factors are summarized in Table 4.1 Section 4.5.

5.1.3 Challenges With Security Visualization User Studies

While there have been many papers published in the field of security visualization, user studies are still fairly uncommon. This is likely because the *typical user* of these visualization systems needs knowledge about the network (IP space, number of hosts, etc.), data (logging setup, capture period, etc.) and context being analyzed in order to provide an accurate judgement about a security event. All these requirements make a proper user study more difficult. Furthermore, many user studies in the literature frequently involve asking a small number of undergraduate students in Computer Science: “What do you think is happening in this visualization?” and making claims based on how close their answer is to what is actually occurring. The benefits of such experiments remain questionable. While our methodology does not carry out a proper user study either, it uses expert evaluation to help identify appropriate visualization techniques for representing network attacks.

5.2 Methodology

This subsection outlines the steps needed to select an appropriate visualization tool for a given class of network security attack. Our methodology begins with an informal task-based analysis of each attack, followed by the identification of new or existing visualization tools that allow for the accurate completion of such tasks. Our approach shares some similarities with the *cognitive walkthrough* [80] method used in the field of usability inspection, but we do not perform usability analysis on any visualization tools.

5.2.1 Task-based Analysis

We begin with the definition of the tasks that analysts must perform to correctly identify and understand security threats. These tasks are defined by an expert (i.e., the author) in network security using the phases (monitoring and analysis) described in Chapter 3. More specifically, we use the available empirical literature on network attacks, along with an analysis of the work flow of a typical network security analyst (see Section 3.2.3) to select the most appropriate visualization techniques based on which tasks they allow the analyst to accomplish.

The *monitoring* phase requires the analyst to create a mental model of the current network status. This is achieved through a high-level overview of multiple data sources over an extended period of time. Specifically, the tasks that must be performed at the monitoring phase are:

1. Maintain an overview of the network;
2. Selectively zoom in to areas of interest; and
3. Interact with raw data as little as possible.

The *analysis and diagnosis* phase requires the in-depth investigation of events that have attracted the analyst's attention while monitoring the network. The analysis phase by definition needs much more detail than the monitoring phase, since the analyst is already focused on a smaller time period. This phase involves the following tasks:

1. Obtain details about a subset of data (zooming); and
2. Display details either dynamically or statically.

The identified tasks serve as a general starting point for selecting appropriate visualization techniques. Future work could include refining task definition for other classes of attacks.

Other Factors

We argue that the quality of the conclusions drawn from visual analysis depends not only on the visualization used, but also on the expertise of the user interpreting the data, as well as the original data set characteristics. Information visualization follows the "garbage in, garbage out" motto in that poor quality data fed into even the most powerful visualization tool will result in an image not as good as if higher quality data were used. Capturing more data is generally preferred because it leads to fewer gaps or lower potential loss of information. At the same time, storing the unmodified data (e.g., without anonymizing), will help provide results that are applicable to real-world attacks.

There are other factors that may help determine if a visualization technique is useful for exploring a class of attack. Some of the surveyed tools in Chapter 4 can eventually provide good insight, but have steep learning curves, requiring a longer time investment to get the data "just right" for analysis. In his book [62], Marty points out that minimizing *non-data ink* (pixels on the graph that do not reveal any information about the data set) is important as well, since we don't want to give up precious screen real-estate to bulky labels.

5.2.2 Method for Selecting a Visualization Technique

Our first step in selecting a visualization technique for a particular attack is to define what tasks analysts need to perform. Once the tasks are defined, we look at the important variables in each phase of network monitoring, and discard visualizations that do not allow the accurate representation of those variables. This is a coarse but effective way of discarding techniques which will not help the analyst.

As a complimentary step, we look at previous approaches in the literature for the visualization of network attacks, and verify if those approaches match the task and variables criteria. If our recommended visualizations differ, we identify the improvements our suggestions offer over previously-suggested approaches.

5.3 Monitoring in Real-time

As mentioned in Section 3.4, real-time visualization systems should aim to provide a constant overview of the data being monitored. Ideally, the system would allow input from as many sources as possible (firewall logs, IDS logs, network packet captures, etc.) and display the data set related variables corresponding to those sources visually without occlusion.

5.3.1 Denial of Service

Previous Approaches

Pearlman et al. [72] use their glyph-based visualization to display a DoS attack. Their link graph visualization (as shown in Figure 5.1) represents one host per node with an edge drawn between hosts if they are seen communicating during an arbitrary time-window. As expected during a DoS attack, a single node (the victim) has many edges going toward it (in this case coming from over 100 attackers), which represents a distributed DoS. While the glyph approach is not capable of encoding time (i.e., the glyph inside each node displays only the type and direction of traffic being sent to or from it), fading is used to show stale connections. Nodes or edges that are more transparent imply older connections, while fully opaque ones are brand new. One problem with this type of time-encoding is that it is not easy to measure the time between an old connection and a very old connection, for example. To counter this problem, the authors use "time slices" rather than animating[1] the display. With time slices each visualization is a graphical snapshot of what happened during a defined time period, and the time period can be either shifted or slided to generate newer images.

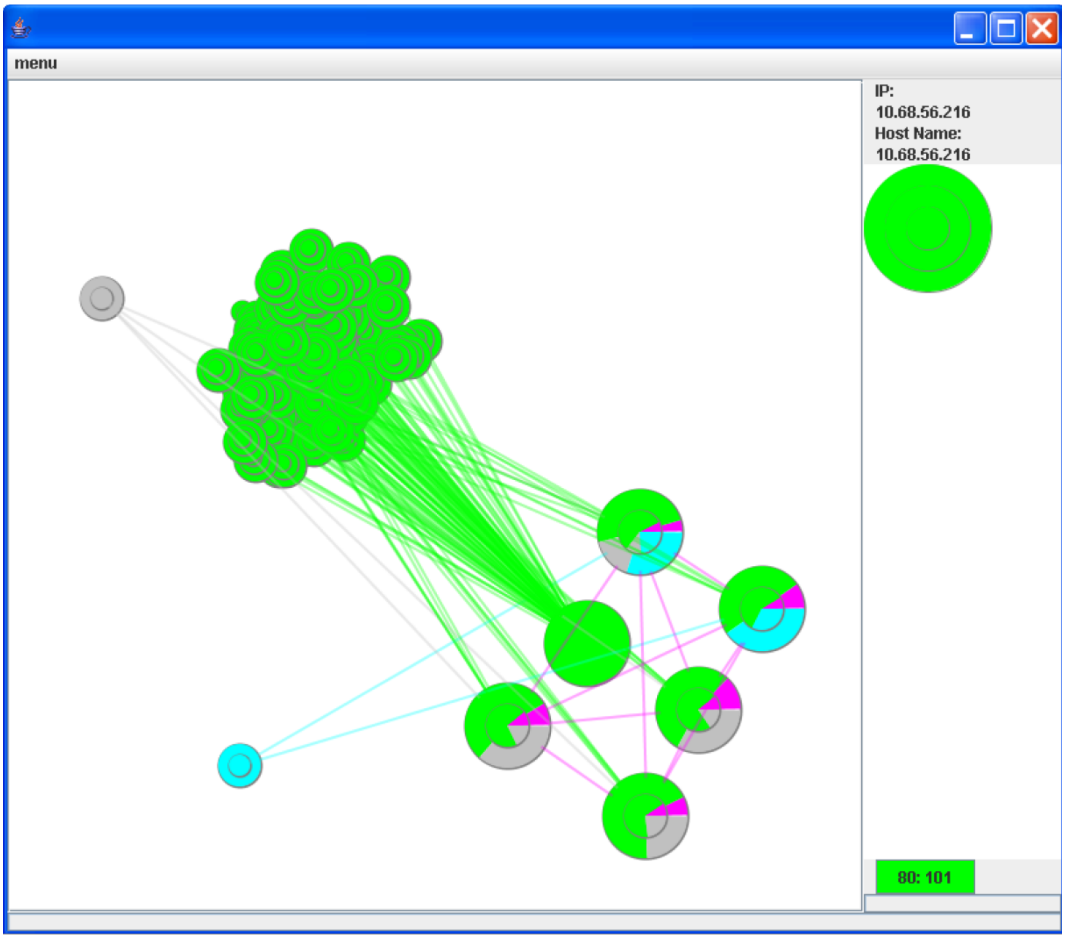


Figure 5.1: Screenshot of the glyph-based link graph display for visualizing DoS attacks [72].

[1] Animation is often discouraged in visualization due to the heavy reliance on the analyst's short-term memory. Static images, even with a playback feature, usually offer more benefits.

Tasks

Using the description of a DoS attack from Section 3.6.1, we define the following tasks specific to this class of attack. Tasks for this section also include those defined for the overall monitoring phase of network security.

1. Identify connectivity between hosts or devices;
2. Identify sudden changes in variables; and
3. Identify changes in behaviour (such as a services going from active to inactive).

Analysis

We argue that using link graphs to display DoS attacks convey only some of the information needed to detect an attack. Link graphs are capable of accurately displaying the *connectivity* between hosts, and when combined with techniques like glyph nodes, the type of traffic being transferred can be displayed as well. However these graphs do not generally scale well when the attack is targeting more than one host. Link-based visualizations tend to become cluttered when nodes have many edges, and even moreso when many nodes show this behaviour on the same graph. A 3D link-graph might help place nodes more intelligently while preventing clutter and bypassing the need for a good 2D layout algorithm, but requires more interactivity and navigation from the user.

One of the important variables identified for detecting DoS attacks in Section 3.6.1 is bandwidth. The link graphs by Pearlman et al. do not encode bandwidth information in the edges (i.e., all edges are the same colour and width). Attempting to display bandwidth by using edge thickness might work, but would lead to occlusion with even a small number of hosts. The precise value of the bandwidth variable in the DoS attack is not critical, but having a sense of whether a host (either a source or a target) is using high bandwidth can be of use to the analyst.

Another important independent variable to consider is the number of remote hosts involved in the event. On a link graph, there is no intuitive way of determining if the number of hosts connecting to a server is high or low unless the analyst has a high level of understanding of the network. For example, seeing 10,000 connections per minute to `http://cnn.com` might be expected as the site is a global news outlet. Seeing 10,000 connections per minute going to a site that generally sees 100 connections per minute should cause alarm. Finally, since one of the ways to end a DoS attack is to engage in talks with the attacker's ISP, obtaining geographical information can also be useful.

Recommended Visualizations for Detecting DoS Attacks

We recommend using link graphs to display connectivity changes. While link graphs are prone to occlusion, in this case they remain useful in identifying a large number of connections towards a small set of targets. Bandwidth should be encoded in the link graph as well, either through glyphs at the nodes or colour or transparency at the edge level.

DoS attacks are one example where complementary visualizations (see Section 2.1.4) can be used. Since link graphs do not show trends in bandwidth, or the number of sources, a histogram can be used to improve the analyst's overview of the attack. Sparklines could help keep track of averages in traffic on individual ports.

We propose a new visualization technique which we call *grid heatmap* for displaying connectivity between hosts without using link graphs. The techniques mentioned above can be used along with grid heatmaps to provide more insight. The advantage of our technique is that we do not show connectivity between hosts using edges, but rather by using a grid. This helps prevent occlusion, and scales better to more hosts.

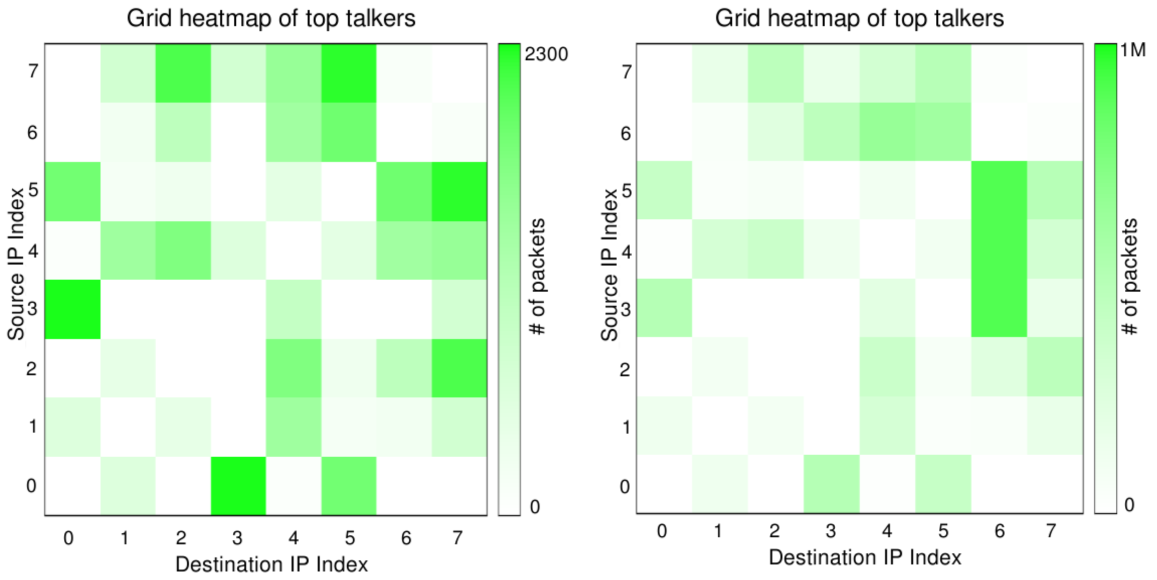
The grid heatmap visualization represents the number of packets sent and received by the *top talkers* of a data set. Wireshark [16] and the SiLK tools [11] have user-friendly options to display the hosts that sent out or received the most packets. The grid heatmap works with packet data (as opposed to netflows), therefore when a host participates in bi-directional communication, there should be roughly the same number of packets sent and received (due to acknowledgements of each packet). We use this observation to identify hosts that are being flooded with network packets, and can't respond to all of them (this is essentially the definition of denial of service).

Once the top talkers have been identified, they are designated as both sources and destinations of connections in the grid. For example, the intersection of row 4 and column 5 will represent the number of packets that host 4 sent to host 5. Under normal network conditions, there should be some symmetry to the grid heatmap, as shown in Figure 5.2(a). This heatmap was created from a publicly available one-hour data set[2] containing 95% TCP traffic. Due to the nature of TCP traffic, every packet that is received must be acknowledged back to the source. This leads to a diagonally symmetric grid heatmap when packet count (as opposed to byte count) is being displayed. Lack of symmetry can indicate that the receiving host is unable to keep up with the sending host (a typical sign of DoS), or that the communication is unidirectional (UDP).

We were unable to test our visualization technique under a real DoS attack due to the apparent unavailability of data sets containing DoS attacks. To test our technique, we wrote a simple Python script to inject fake packets into the data set of Figure 5.2(a) and simulate a DoS. In our simulation, hosts 3, 4 and 5 are cooperating to denial of service host 6. Each host sends roughly one million randomly-generated TCP packets to host 6. The resulting grid heatmap is displayed in Figure 5.2(b). Notice how most of the grid has faded, and the column belonging to host 6 has more traffic coming specifically from hosts 3, 4, and 5. We also notice that the row belonging to host 6 (i.e., packets host 6 sends in response to incoming requests) is lighter than the corresponding column, indicating that there are more packets flowing *to* host 6 than *from*.

We envision the grid heatmap visualization being used on-demand by security analysts when there is suspicion of a DoS attack. An alert (either automated or reported by a user) would cue the analyst to bring up the grid heatmap of a recent time period. The grid heatmap could also update in real-time allowing the analyst to keep track of the current state of connections, as well as compare the heatmaps from 2 or more points in

[2] The data set contained about one million IPv4 packets, and contained mainly web traffic.



(a) Normal traffic

(b) Simulated DoS

Figure 5.2: Grid Heatmap under (a) normal conditions and (b) a simulated DoS attack. IP index is a pointer to a list of IP addresses that generated the most traffic.

time. Alternatively the heatmap could be displayed constantly, and used as a tool to improve situational awareness.

5.3.2 Network Scan Detection

Previous Approaches

Scan detection has been an active area of research for many years and a large number of techniques (both visual and non-visual) for detecting network scans have been developed. In fact, scan detection is still so popular that most security visualization authors will often demonstrate how their techniques and tools function under a scan attempt, even when the tool is not explicitly designed for scan detection.

Portvis [63] facilitates the detection of scans by visually representing port activity in several different ways. The main display shows colour encoded ports where the pixels corresponding to a port remain black if there hasn't been much activity targeting that port, and white otherwise. Intermediate colours such as dark blue and red indicate low and high levels of activity, respectively. Port banding (i.e., entire regions of ports displaying the same colour patterns) can indicate that a scan took place during the highlighted time window. Another way Portvis facilitates visual detection of scans is by creating images containing strongly highlighted horizontal or vertical lines. The main display shows port numbers on the x-axis and timestamp on the y-axis; thus a remote attacker connecting to the same port repeatedly over an extended period of time will induce a vertical line at that port on the display. Similarly, scans that happen over a short period of time and sweep a large number of ports create a (mostly[3]) horizontal line on the display. Portvis is good at helping the analyst detect scans that happen over a short period of time. Low-and-slow scans on the other hand will go undetected in Portvis because they won't significantly change the port activity, or create distinct vertical or horizontal lines.

A visualization highlighting low-and-slow scans was achieved by Grizzard et al. [40]. Their tool involves a simple 2D scatter plot (which they call the *TCP histogram*) similar to that of Portvis with two key differences: first, port numbers on the x-axis range from 1 to 1024 rather than 1 to 65535; second, timestamps on the y-axis display extended time periods (about 2.5 years in their examples) of network data. While in general, the larger the data set, the more occlusion there will be in the visualization (due to the larger number of data points), this seems not to be the case with the TCP histogram because the resulting visualizations show highly contrasting white lines on a black background corresponding to slow scans. The authors identify up to 8 scans in their visualization tool, 4 of which were done over an extended time period and might have been highly distributed, demonstrating the usefulness of aggregating and visualizing extremely large amounts of data.

Irwin et al. [50] evaluate Snort [12] and Bro [3] scan detection accuracy by visualizing network traffic in a cube-like display with the Inetvis tool. As usual the axes represent source IP address, destination host, and port number. The authors compare reports by the text-based scan detection tools to the visual fingerprint of the scans to validate the effectiveness of both Snort and Bro. Inetvis is unable to detect low-and-slow scans

[3] The line would actually be more of a diagonal if the timestamp is very accurate.

although Irwin et al. argue that text-based tools also fail at this task.

Tasks

We identify the following tasks for detecting scans:

1. Monitor port behaviour for single hosts and groups of hosts; and

2. Display extended time periods as well as short time periods.

Analysis

Ideally we seek a visualization technique capable of highlighting all types of scans, ranging from easily detectable and obvious random worm scanning to targeted low-and-slow scanning. However, we believe that there might be more value in focusing on slow scanning since the currently available text-based techniques [52, 12, 83] are still lacking in the detection of these types of scans. Any of the aforementioned scan detection tools will easily classify a host as a scanner if it is rapidly probing a wide range of ports.

Because the visualization of scan detection requires displaying port information and IP addresses (as identified in Section 3.6.2), a visualization technique that is capable of displaying only a small number of variables suffices. Scatter plots have been established as the *de facto* standard for scan detection due to the visual fingerprint left by scans (recall the naming of vertical vs. horizontal scans). We have seen however that many authors omit displaying time as one of the variables; for example, the spinning cube of doom [56] and Inetvis [50] do not display time as one of the axes.

We believe it is important to display time directly (as opposed to encoding it in some other way). Having the capacity to visualize the time-frame of a scan is useful because while some scans might happen over the course of a few seconds or minutes, slow scans will take longer. Thus it is important to display extended time periods, while also allowing the analyst to zoom in on shorter timespans of interest.

Aggregation (grouping port activity from all monitored hosts into one) is useful as well, since it can help detect distributed scans (many hosts attempting a small number of connections to a small number of target machines). The Portvis approach plots time and port number, but does not specifically display IP addresses. The resulting visualizations help visualize scans, regardless of how many sources are participating. The disadvantage of this technique is that the analyst must still search for the source IP addresses in the raw data once a scan has been identified.

Recommended Visualizations for Detecting Network Scanning

The TCP histogram approach of Grizzard et al. [40] was designed specifically for detecting stealthy scanning, and currently seems best at detecting these types of scans. It does require large amounts of data as well as large amounts of processing power, so it may not be an easily applicable technique for all networks. However we strongly believe that the notion of aggregating large amounts of data, and visualizing connections over an extended period time is what reveals the broadest variety of scans.

Due to the tried-and-true use of scatter plots for scan detection, we recommend using this visualization technique for scan detection with emphasis on the following features:

- *time as one of the axes.* Helps the analyst obtain contextual information without requiring interaction with the tool.

- *extended time windows (days or months) with zoom capabilities and real-time updating.* Visualizing extended periods of time facilitates the detection of low-and-slow scans while simultaneously allowing the detection of short-lived scans. Dynamic zoom capabilities help in detecting short-lived scans.

- *remote host aggregation.* Displaying only port information rather than IP address and port combinations allows the analyst to detect coordinated scan activity, regardless of the number of attackers. Obtaining the IP addresses of attackers must be done manually (by looking up identified port numbers and timestamps in the raw data set, or automating the tool to do this upon clicking). An alternative is to encode the most significant bits of the IP address with particular colours to give the analyst a sense of what IP range to focus on.

5.4 Analysis and Diagnosis

In the analysis and diagnosis phase, analysts have already been alerted of suspicious activity and are attempting to classify a detected event as either a threat or a false positive. To do this, analysts need as much information as possible. Therefore, visualization tools and techniques that are capable of displaying more features from the data set tend to provide greater insight than those that display fewer during the analysis and diagnosis phase.

5.4.1 Brute-force Password Guessing Attacks

Previous Approaches

As brute-force password guessing attacks have become more popular in recent years [10], we are beginning to see more research in the area of visualization of this threat. Brute-force attacks have been explored in at least two different ways in security visualization literature. The first approach by Fischer et al. [34] uses treemaps (which display internal IP addresses and type of service offered through colour encoding) with overlayed edges connecting points on the treemap to nodes (representing remote attackers) outside the treemap. When there are many sources of an attack being displayed, the edges displaying connectivity can saturate the visualization. In an attempt to prevent occlusion, Fischer et al. use the edge bundling technique [46] described in Section 4.3.6 and shown in Figure 5.3. We notice that even with the edge bundling technique, large amounts of overlap remain, completely hiding the bottom-left quadrant of the treemap. The saturation also makes the image somewhat daunting to analyze since it is representing a lot of information. Realistically, an analyst looking at this visualization will only see that large numbers of hosts are connecting specific targets; all other information must be looked up manually in the original data set.

Another attempt of visualization of brute-force password guessing attacks, by Le

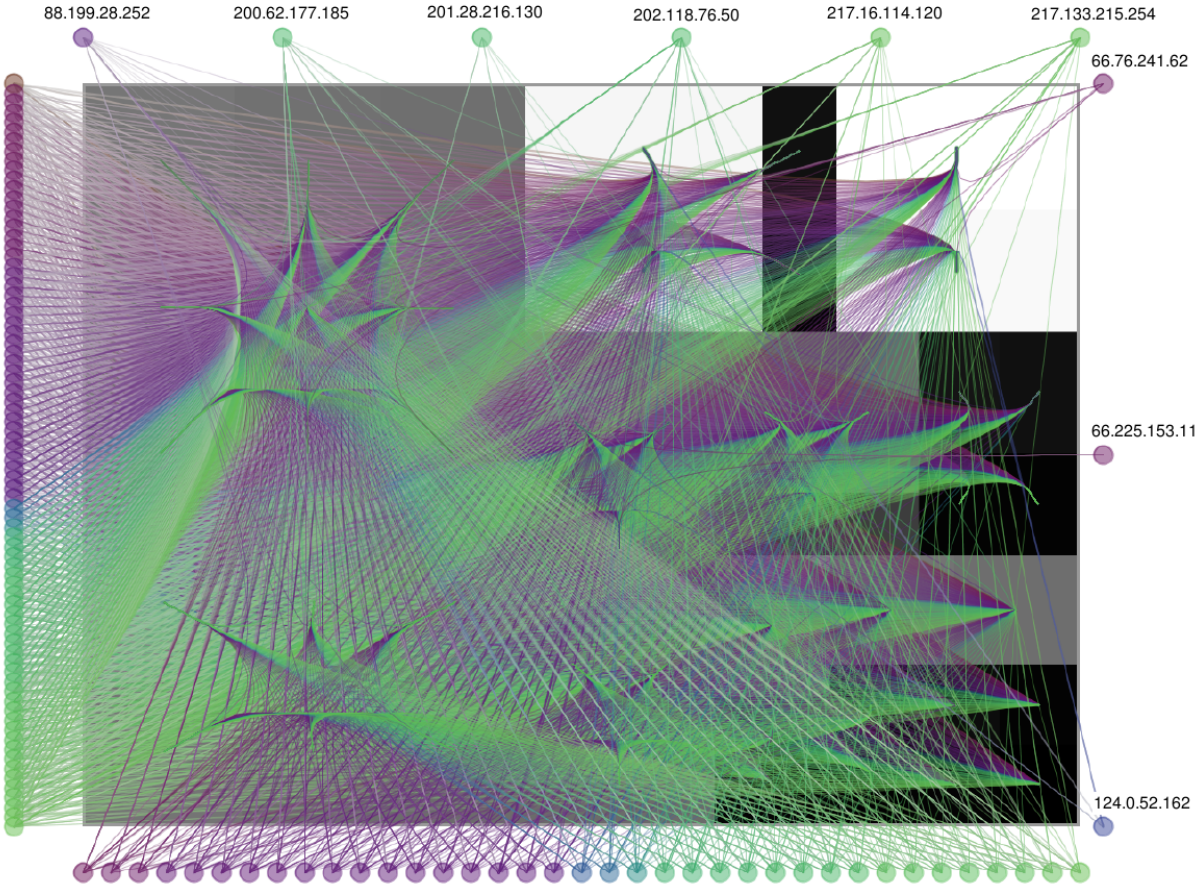
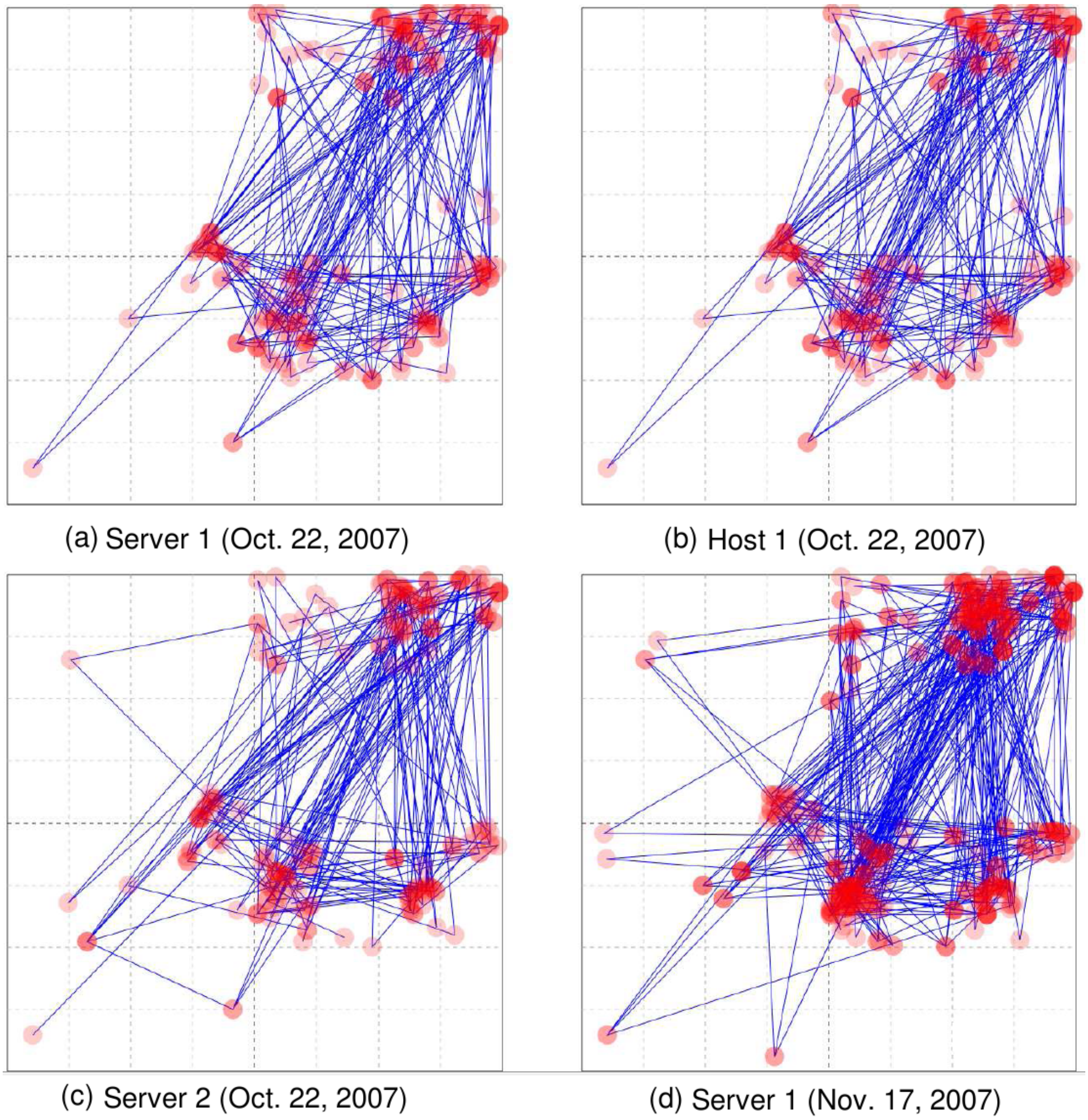


Figure 5.3: Treemap/link-graph visualization of a distributed SSH brute-force attack. Attackers are represented as circles surrounding the treemap, and internal hosts are inside the treemap [34]. Best viewed in colour.

Malecot et al. [57], uses quadtree maps to generate visual fingerprints of distributed SSH brute-force attacks. By parsing authentication logs and looking for failed attempts, the tool places a red circle at the attacker's IP address. IP addresses which are close (e.g., in the same subnet) will create red circles that slightly overlap, giving a stronger colour. Blue lines are drawn between sources as password guessing attempts are received from new IP addresses.

Once visual fingerprints are generated (Figure 5.4 shows four of these fingerprints), the analyst should be able to compare new attacks to previously-seen attacks. The advantage of building these "visual fingerprint libraries" remains to be proven, but Conti [25] suggests that they can help provide the analyst with a deeper understanding of the attacker.



(a) Server 1 (Oct. 22, 2007)

(b) Host 1 (Oct. 22, 2007)

(c) Server 2 (Oct. 22, 2007)

(d) Server 1 (Nov. 17, 2007)

Figure 5.4: Visual fingerprints of two different distributed brute-force SSH password guessing attacks using quadtree maps [57]. Fingerprints (a) and (b) are visually similar and attributed to the same attacker. Fingerprints (c) and (d) are presumably from different attackers.

Tasks

We define the following tasks for security analysts to detect and analyze a brute-force password guessing attack:

1. Detect changes in the number of connection attempts or usernames guessed per unit of time.

2. Identify large numbers of hosts with similar activity patterns.

Analysis

The treemap/link-graph hybrid visualization of Fischer et al. [34] may be useful for other network attacks, but not for visualizing password guessing attacks. As we can see in Figure 5.3, occlusion is very likely to occur even with less than 100 attackers. The quadtree map approach of Le Malecot et al. [57] seems somewhat cleaner and is useful as a visual fingerprint (Figure 5.4), however the analyst can still learn only limited information from the image itself, since the display only shows circles at unlabeled IP addresses.

We suggest that detecting and understanding brute-force attacks requires a visualization technique which displays multiple variables, as well as on-demand access to other variables that may be required by the analyst. These features also help the analyst achieve the tasks outlined for this class of network attack. As is the case with the monitoring phase, histograms or line charts should be used to monitor activity on the service/port hosting the login facility. A line chart or sparkline displaying trends allows for an intuitive way to monitor the average and historical values of specified variables.

For in-depth visual analysis, we suggest the use of parallel coordinate plots or techniques that display many variables. A technique is also preferred if it can display activity patterns such as times at which IP addresses contact servers, or the distribution of usernames (what usernames were tried, and how many times each was used) tested. Finally, a technique that might seem intuitive for the analysis phase and brute-force attacks in particular is geographical mapping (i.e., showing the location of the attacker's IP address on a world map). Mapping, while revealing only a small amount of information can help the analyst prioritize the contacting of foreign ISPs if necessary.

Recommended Visualizations for Password Guessing Attacks

In Figures 5.5 and 5.6, we demonstrate how two existing visualization techniques (parallel coordinate plots and word clouds) can be tailored for application to analyzing of brute-force password guessing attacks. These visualizations provide more insight than previous approaches by Le Malecot et al. and Fischer et al. The images were created from a university lab data set logging failed connection attempts to the main SSH server over a time period ranging from December 1, 2008 to January 31, 2009. Sensitive data such as the valid usernames have been replaced with $userN$.

Parallel coordinate plots are excellent for displaying multiple variables and especially for showing how variables relate. In the case of brute-force password guessing attacks, the variables of interest are guessed username (note that we do not log passwords[4]), time of the login attempt, source IP address of the connection and type of error message logged. Figure 5.5 shows a 24-hour subset of our two month authentication log in parallel coordinate plot form. The data was taken from midnight to 11:59 P.M. on January 29, 2009. In this example we colour-encode failed connections with red, and successful connections with black. We immediately notice that *failed password*[5] authentication logs occur throughout the day, but they frequently come in groups spanning short periods of time. These failed password runs are brute-force password guessing attacks. We also notice that the most common form of successful login is *password* authentication rather than key-based logins. Finally there is a large number of red lines going to the user *root* on the user login axis. The root account is a common username guess on Linux operating systems because it has administrative privileges and can perform any change on the system.

Because parallel coordinate plots do not have a native way of displaying frequency (i.e., it is not easy to see which username was the most used), we suggest using a complementary

[4] Most authentication servers disallow logging of passwords to prevent sensitive information from leaking into log files. While it would be interesting to see what passwords an attacker is trying, legitimate users can also mis-type their passwords, which would get logged as well.

[5] Failed keyboard-interactive log entries identify remote hosts using newer versions of the SSH protocol, and are less common in Figure 5.5.

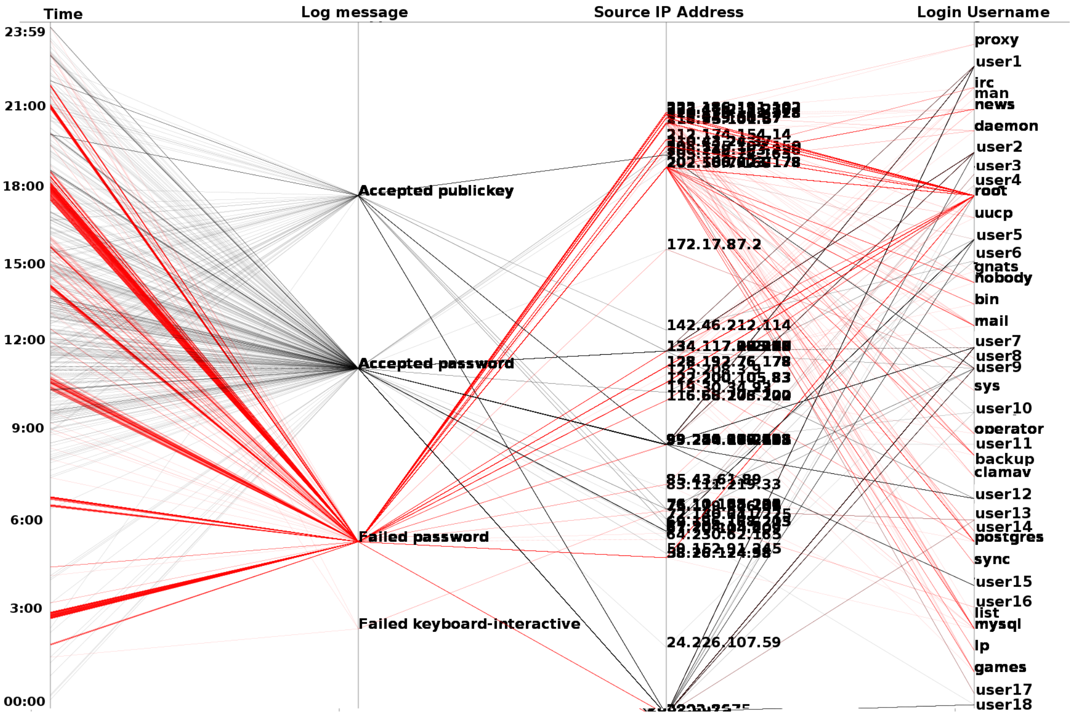


Figure 5.5: Parallel coordinate plot representation of an auth.log file

visualization technique (Figure 5.6) to see which usernames were used most frequently. We select the *word cloud* [43] technique which can provide this information easily, allowing an analyst to quickly spot usernames that have the most number of failed logins. Figure 5.6 plots all the tested usernames during the two-month period, with the font size of each username proportional to the number of failed attempts recorded in the authentication log file, making it very straightforward to identify that usernames *test* and *guest* were popular choices for attackers. The *root* username was removed from the word cloud due to its frequency being orders of magnitude more than any of the other usernames.

We expect the word cloud and parallel coordinate plot visualization techniques to be used when an analyst performs in-depth analysis of authentication logs. If the authentication server receives a small number of connections on average, there is limited value to be gained from constantly updating the parallel coordinate plot, therefore displaying the visualization statically is preferred. In this case, we suggest that these graphs be created at pre-determined intervals (e.g., every week or month) and analyzed with the purpose of updating access control rules, enforcing password policies, or explaining behaviour to upper management.

One of the limitations of the parallel coordinate plot for visualizing authentication logs is labeling. The *source IP address* axis in Figure 5.5 displays the entire IPv4 address space, which makes some numerically adjacent IP address labels overlap. Future work includes finding a solution to the labeling problem in parallel coordinate plots and other visualization techniques.

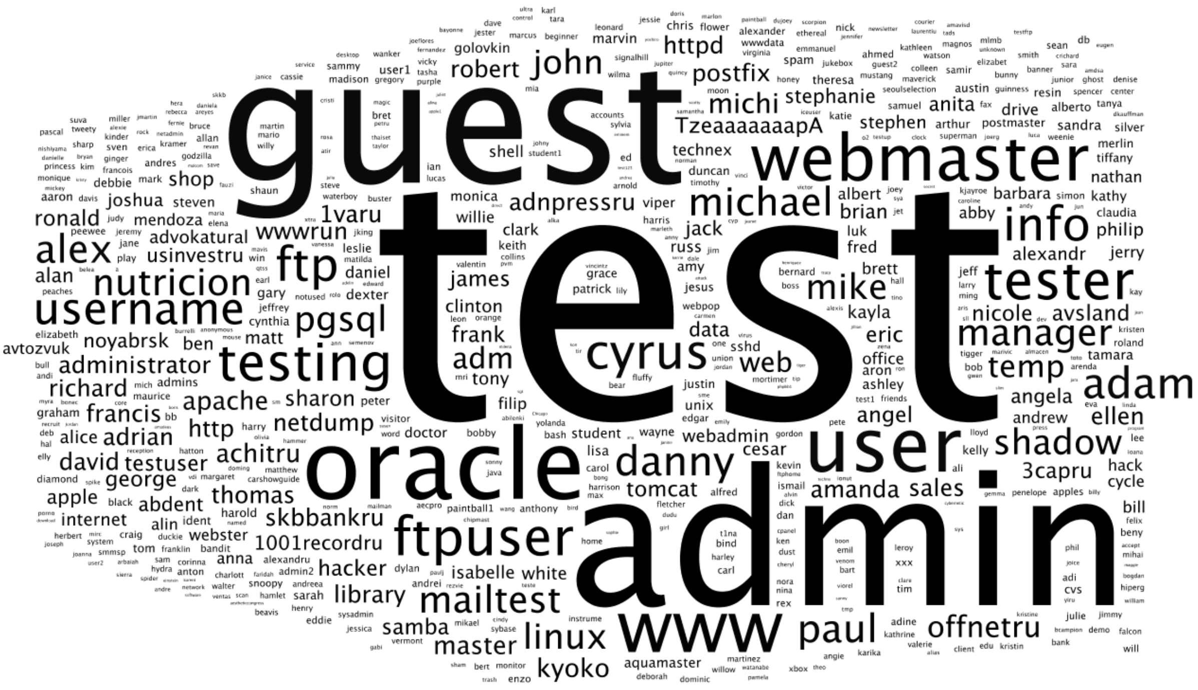


Figure 5.6: Word cloud visualization of failed usernames during a brute-force password guessing attack (excluding *root*, as discussed inline).

5.5 Summary

This chapter has presented a set of guidelines, analysis, and examples for selecting an appropriate visualization technique for specific classes of network attacks. Table 5.1 summarizes the main requirements for each phase and class of network attack, and the visualizations we have recommended. In Chapter 4, we characterized the surveyed visualization techniques (see Table 4.1), listing features available in each one. Our recommendation of visualization techniques focuses on identifying requirements for detecting and analyzing

Phase	**Class of attack**	**Requirement**	**Preferred Technique**
Monitoring in real-time	Denial of Service	• Display connections between hosts • Display the number of hosts • Show changes in bandwidth usage	• *Grid heatmap • Sparklines, histograms, trend lines • Other link graphs
	Network scanning	• Display port numbers accessed over time • Display extended time periods • Display aggregated port information	• Scatter plots with zoom capabilities, and port-based aggregation
Analysis and Diagnosis	Brute-force password guessing attacks	• Display attack time patterns • Show how usernames and IP addresses relate • Display relative username guess frequency	• †Parallel coordinate plots • †Word clouds

Table 5.1: Summary of visualization technique recommendations. (*) denotes new proposals. (†) denotes new use of existing proposal.

each class of attack, and eliminating techniques that don't meet these requirements.

We emphasize that there is insight to be gained by using other techniques, and the visualizations we have proposed have yet to be tested on larger and more diverse data sets. The intent is that this chapter provides guidance as to how the selection of a tool may be carried out.

Chapter 6

Visualization of IPv6 Data Sets

This chapter considers the problem of security visualization tools dealing with data sets containing IPv6 addresses. Most current tools support IPv4, but not the new version of the protocol. In this chapter, we review the current state of IPv6 support in popular visualization tools, and propose two new techniques [18] for visualizing data sets with IPv6 addresses.

6.1 Introduction

The IPv4 protocol has been used for practically every type of network communication for over 25 years. Its limited address space (of about 4 billion) is projected to be exhausted by April 2012 [47]. As IPv6 gains popularity, attackers are updating their tools and techniques to use it and benefit from the fact that IPv6 is often overlooked, disabled or otherwise unsupported in many network analysis tools.

For security visualization, the IPv6 address space represents a challenge due to its massive space (2^{128} possible addresses). Techniques that work well for IPv4 such as displaying the entire address space on a single plot are not possible for IPv6, allowing attackers who are using IPv6 to bypass analysis and detection. Furthermore, many current security visualization tools assume that the underlying protocol is IPv4 and simply ignore any IPv6 traffic included in a data set.

6.1.1 Extension of the Visualization Technique Selection Methodology for IPv6

Selecting visualization tools for visualizing IPv6 data sets is difficult since there is currently a limited availability of tools which support it. Our methodology in Chapter 5 applies equally for IPv6 related data, since it focuses mainly on the tasks the analyst needs to accomplish. However, the selection of a specific tool might not be possible if there are no tools which allow the analyst to visualize the IPv6 address space.

6.1.2 Related work

There has been little related work in the literature that specifically addresses visualization of IPv6 traffic. The existing literature on security visualization also does not explicitly state that the focus is IPv4 traffic, but typically implicitly assumes that IPv4 is the underlying protocol of the analyzed network traffic. Several visualization tools [54, 55, 76] which provide good insight into IP behaviour are limited to IPv4 addresses.

There have been attempts to visualize the structure and hierarchy of the IPv6 address space. Nakamae el al. [75] attempted to visualize the structure of an IPv6 network by using a 3D link graph display as shown in Figure 6.1. The 3D link graph shows the Widely Integrated Distributed Environment (WIDE) testbed network connectivity with universities and ISPs in Japan. While their approach creates attractive graphs, the insight gained remains questionable. CAIDA have also studied IPv6-enabled ISPs around the world, and produced visualizations displaying IPv6 autonomous system (AS) interconnections [21].

6.2 Differences Between IPv4 and IPv6

IPv6 and IPv4 operate at the Internet layer of the IP stack (see Section 2.3.1), therefore the only differences between protocols are at this specific layer. TCP, UDP and any other protocols that run over IP are unchanged, and remain fully compliant with current visualization tools regardless of the underlying IP layer. The implication this has for

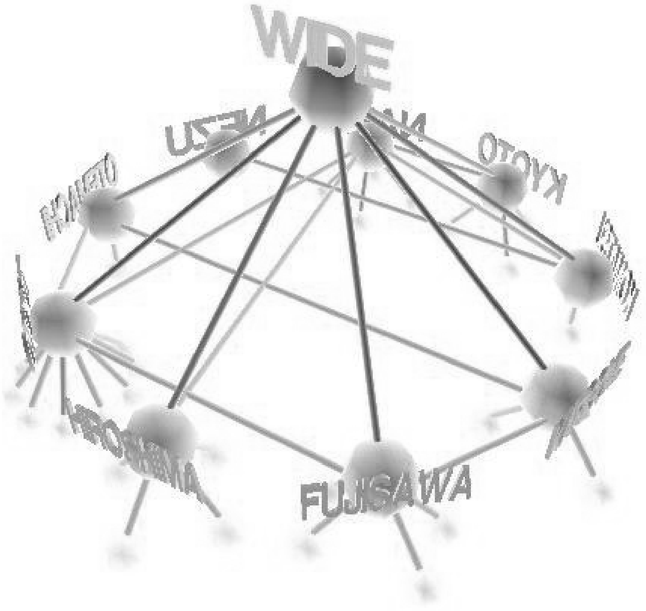


Figure 6.1: 3D Link graph view of an IPv6 network [75].

network security is that only visualization tools and techniques that display segments of the IP header will be affected. Visualization tools that display transport data such as port numbers or application data such as log files will require no modifications to work over IPv6.[1]

Address Representation. IPv6 addresses are not written in the IPv4 dotted-decimal notation, but rather as a group of eight 16-bit *hextets* (as opposed to four 8-bit pieces in IPv4). Leading zeroes in each hextet can be omitted, and up to one group of two or more hextets consisting of only zeroes can be expressed as "::". The following examples show different possible notations for an IPv6 address.

```
2001:0DB8:0000:0078:9ABC:0000:0000:0000

2001:0DB8:0:0078:9ABC:0:0:0

2001:DB8:0:78:9ABC::
```

The representation of prefixes in IPv6 is similar to the Classless Inter-Domain Routing (CIDR) notation in IPv4. The prefix length, which specifies how many of the leftmost

[1] No changes are required provided these tools can actually parse IPv6 successfully and extract the upper layer payloads.

contiguous bits compromise the prefix, is appended after the IP address. For example, a network block where 64 bits are used for assigning to hosts and 64 remain constant would be expressed as: 2001:DB8:0:78::/64.

6.3 Visualization Proposals and Examples

To represent IP addresses in IPv6, naïvely trying to grow the range of an axis from 2^{32} to 2^{128} does not work. We must rethink how to display information to gain insight on how connections are distributed and where they are originating. For our IPv6-compatible visualization proposals, we seek to work with the existing visualization tools with as little disruption as possible, even though such tools might not be ready for IPv6.

In the following sections, we propose a filtering technique that helps reduce the occlusion of IPv6 sources in graphs and a treemap display for representing the vast space of addresses in IPv6.

6.3.1 Indexing

One of the problems for visualization in IPv6 is that the address space is very sparsely populated. Most of the leftmost 4 bits are reserved by the IETF for future use, which leaves only one prefix (2000::/3) that can be globally routed.

In IPv4, for an analyst to detect patterns and trends using a visualization tool, it is useful to look at neighboring addresses (in the same subnet or class). This can help identify hosts that are collaborating or exhibiting similar behaviour. Furthermore, being able to see the behaviour of all subnets simultaneously proves to be useful in detecting Internet-wide attacks. IPv6 doesn't immediately add more hosts to visualizations, just a much wider address range. Removing the unused or unpopulated addresses, i.e., the whitespace (sometimes referred to as *darkspace* in scan detection literature), should restore the same, Internet-wide visualizations as with IPv4.

Eliminating the whitespace is easily done as follows [18]. During the network capture, build a list of unique remote IPv6 addresses observed. To preserve relative placement of addresses (so that if address A numerically follows B, this relation is retained in the new list), sort the resulting list by numerical value. Finally, when displaying data, display the index of the IP address in the list instead of the full address.

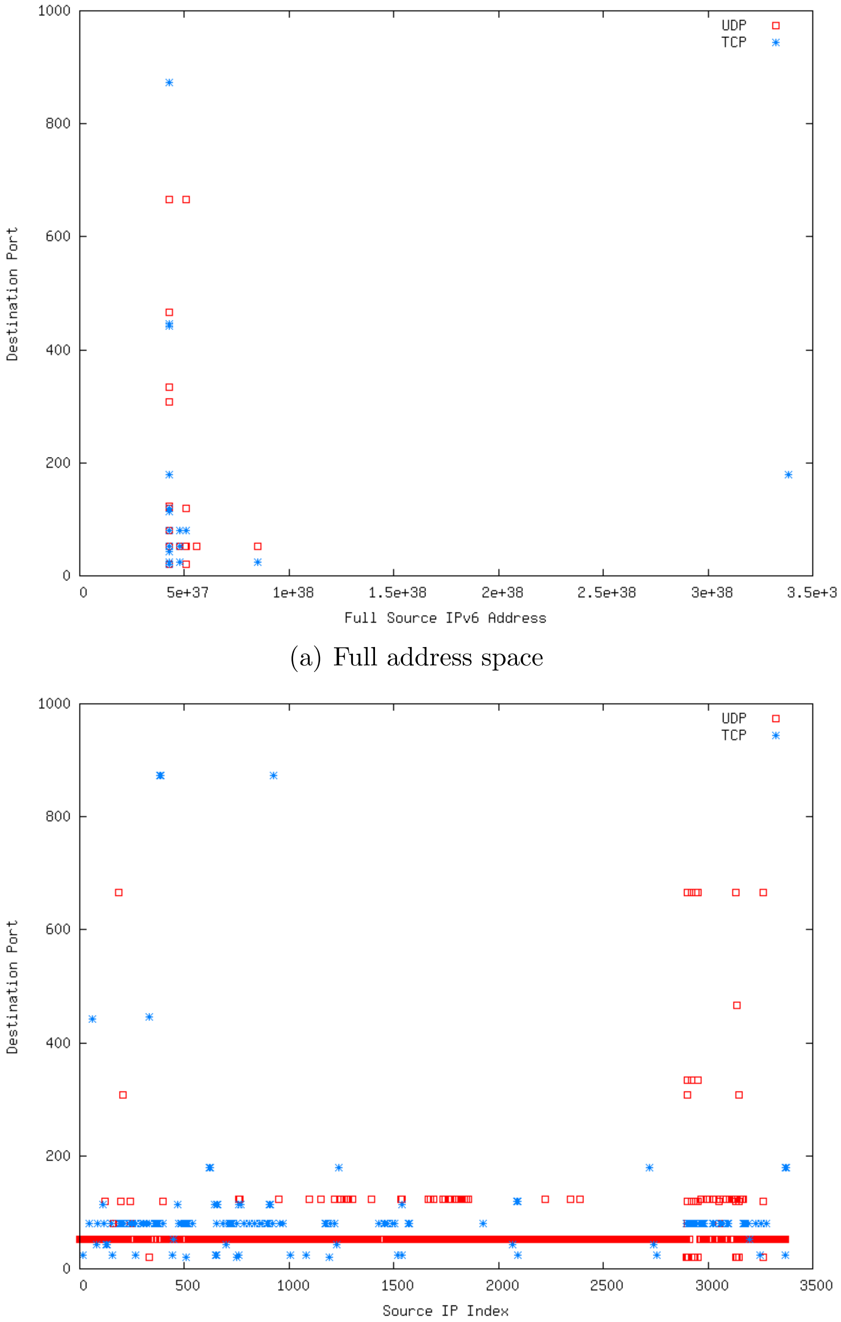
As an optional step, subnet information can be preserved in the graphs, by deliberately inserting gaps. We insert 2 null elements into the previously mentioned IPv6 address list between each populated 64-bit subnet. The end result is a list where all addresses in the same /64 subnet are contiguous, and all /64 subnets are separated by a small space. This optional step may help the user identify subnets visually rather than trying to otherwise validate findings in the original raw data set.

To save the analyst from changing his work flow by having to look-up the address that matches a given index, visualization tools could be modified to display the address when mousing over an index.

Figure 6.2(a) presents our data set as an example IPv6 capture being visualized with no special treatment for IPv6 addresses. The x axis displays the 128-bit addresses as an integer value. Due to the uneven distribution of addresses, as well as the huge IPv6 address space, the populated addresses crowd the left side of the scatter plot. In Figure 6.2(b), unused address space has been eliminated, and each observed remote IPv6 address has been indexed. The resulting scatter plot is able to display over 1400 remote addresses and what destination ports each one targets during the capture period. This processed scatter plot reveals that traffic to UDP port 53 (DNS) was frequent throughout the entire capture period, and that no single host attempted connections to a large number of destination ports (noted by the absence of vertical lines in Figure 6.2(b)). Note that the horizontal axis in Figure 6.2(b) displays the *address index* rather than the full address.

6.3.2 Treemap Hierarchy

IPv6 address allocation standards dictate that addresses must be assigned in a hierarchical manner, due to the extremely large address space. Assigning IPv6 addresses otherwise (e.g., in an ad hoc way) would lead to huge routing tables on backbone Internet routers.



(a) Full address space

(b) Using index value in place of absolute IP address

Figure 6.2: Filtering out unpopulated address space.

RFC 3587 [45] specifies a global unicast format for aggregating IPv6 addresses in order to keep global routing tables efficient. In this format bits of the address are grouped together forming a three level hierarchy: the global routing prefix, the subnet ID and the interface ID. Moving down the hierarchy (i.e., reading more bits of the IPv6 address from left to right), reveals more about the IP address. The global routing prefix might tell us what regional registrar owns the address block, while the subnet ID might tell us what country and ISP own the sub-block. Finally the interface identifier will uniquely identify a host or endpoint.

As discussed in Section 4.4.1, treemaps can represent hierarchy effectively. We use the treemap technique to illustrate how treemaps can be used to visualize IPv6 remote addresses [18].

Figure 6.3 shows an example treemap created from a publicly available data set containing IPv6 packets.[2] The treemap represents all remote IPv6 addresses seen during the 4.5 hour capture period, the number of packets matching each port and their protocol. The size of each rectangle is proportional to the number of packets, and the colour represents the protocol. The treemap is hierarchical (using each hextet of the remote IPv6 address as a level) so viewing the first 3 hextets would display only the global routing prefix, and viewing the first 4 would display the subnet ID as well. The user can zoom in as far as the last hextet of the address, with the drawback of a more cluttered treemap. Rectangles that are too small to display a label can be expanded to full screen when clicked. Mousing over any rectangle gives a popup label displaying the packet count.

The treemap algorithm generated five large rectangles (marked with black borders in Figure 6.3) corresponding to the first 16 bits of the IPv6 address: 2001, 2A01, 2620 and 2A02 which are IANA assigned prefixes; and FE80 which is a link local block. The analyst can zoom in to the tree by double clicking on any of the labels, at which point the treemap will be redrawn using the selected node as the root of the tree. The tree is also searchable which might allow analysts to look for specific prefixes or network blocks. Manually correlating this data with the list of IPv6 unicast assignments [6] by IANA reveals geographical location of nodes.

One of the benefits of treemaps is that the analyst can quickly find sources behaving similarly by looking at the contents of nodes. For instance, in Figure 6.3 (displayed in

[2] The data set contains 2 million IPv6-only packets, generated June 18, 2008 between 9:00 am and 1:36 pm. Packets were anonymized using tcpdpriv and packet payloads were removed, leaving only header information.

landscape mode) to the left, we notice that the highlighted[3] hosts (mostly white nodes in the treemap) were seen sending mostly ICMPv6 traffic during the capture period. Some of these hosts were also seen sending TCP traffic on port 179, corresponding to the Border Gateway Protocol (BGP). These hosts are therefore probably routers or monitoring devices (or devices being monitored).

Another example of using treemaps for detecting specific types of hosts is presented in Figure 6.4. In this treemap all hosts from our data set except those seen generating UDP traffic have been hidden. The resulting treemap displays 3 nodes with a distinctive destination port layout. All three nodes show the most common destination port contacted was 32769 UDP (the second ephemeral port available by default on Unix-based operating systems) followed by port 53 UDP and then many other high-numbered UDP ports. At a first glance, the contents of their nodes would suggest the three hosts behave similarly. This intuition is confirmed by searching for these IPv6 addresses in the data set using the SiLK tools, revealing that all three hosts only responded to requests on port 53 during the 4.5 hour capture.

6.4 Discussion

This chapter has covered the work in a short paper at VizSec 2009 [18]. One of the main objectives of this paper was to raise awareness of the need for security visualization tools to support IPv6. It also presents two new techniques that allow security analysts to visualize IPv6 network traffic, without requiring a major rewrite of existing tools. While our proposals seem to give good results on a sample data set, their practical utility once IPv6 deployment becomes mainstream remains unclear. Current address allocation trends might change, making the treemap visualization more difficult to interpret. Reserved IPv6 blocks might also get allocated, complicating the use of our filtering technique.

[3] When a node or rectangle is clicked, the path from the root to that node is *highlighted* in yellow, helping the analyst visualize the hierarchy of the selected node.

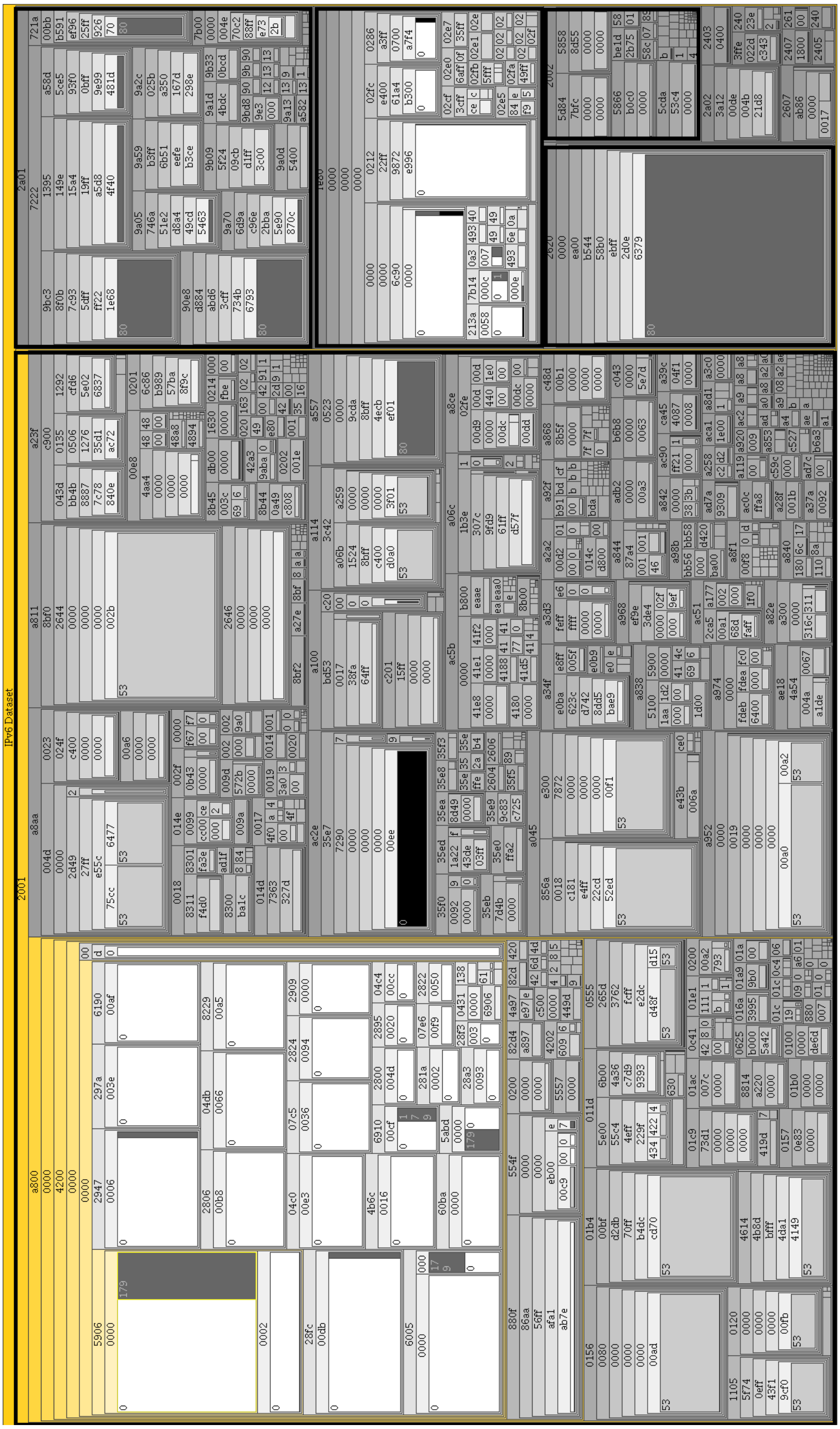


Figure 6.3: A treemap representing IPv6 source address, destination port and packet count. Colour key: ICMP (white), TCP (dark grey), UDP (light grey), other protocols (black). Best viewed in colour.

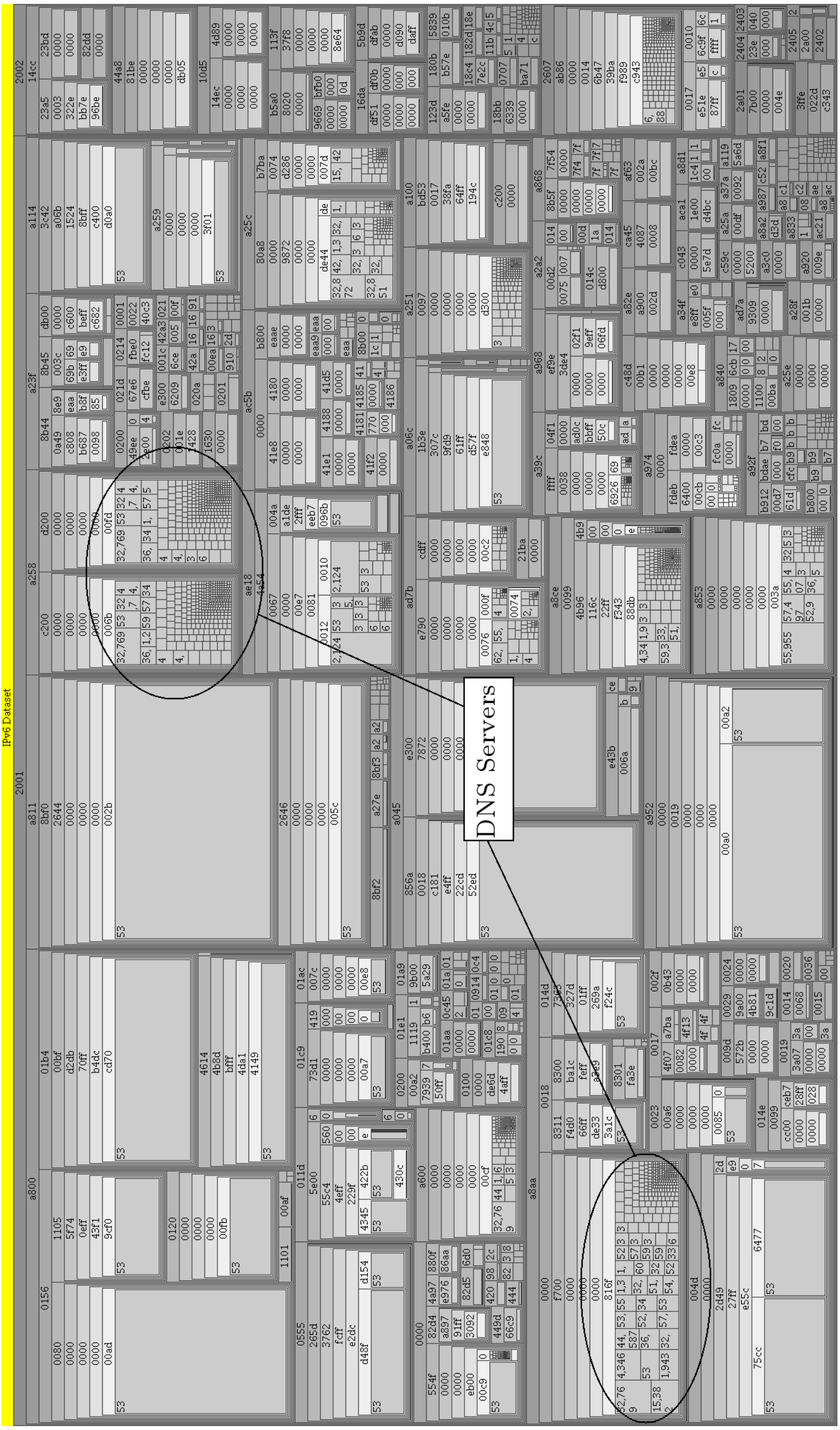


Figure 6.4: Finding DNS servers. Best viewed in colour.

Chapter 7

Conclusions

An objective of this thesis was to pair appropriate visualization techniques with specific classes of network security attacks. To make the pairing possible, we surveyed and classified many network security visualization tools and techniques. Our classification helps group visualization techniques based on the graph type, as well as identify the features that each graph type provides. By understanding in depth each phase of network security monitoring as well as properties of the different visualization techniques, we were able to systematically suggest techniques appropriate to visualize the specified classes of network attacks.

Our methodology for visualization technique selection used a task-based approach to clearly identify the requirements at each phase of network security monitoring. The methodology was applied on three common types of network attacks that network security administrators face regularly, and the results and examples seem to provide the analyst with more insight about the described attacks than previous approaches. We have identified features in visualization techniques that are required for better visualizing network scans. We have introduced three new visualization techniques: the grid heatmap (Section 5.3.1), the IP indexing of IPv6 data sets (Section 6.3.1), and the treemap display of IPv6 address hierarchy (Section 6.3.2); these techniques help analysts achieve tasks for denial of service and brute-force password guessing detection and analysis, and deal with IPv6 data sets. We have also recommended using existing visualization techniques in specific configurations that provide more insight (Section 5.4.1). While we do not claim that the presented visualizations are a perfect approach for visualizing these attacks, our proposed visualization techniques provide more insight with less occlusion than previous approaches in security literature.

Some of the challenges we faced included difficulties when working with different data formats for all the surveyed visualization tools. A subset of tools require raw network captures, while others are only capable of graphing data from text files (in comma or tab separated format). Other tools need netflow data as input, requiring the end-user (i.e., the authors) to pre-process our data sets. Some conversions are easy to perform, while others required high amounts of processing (especially large data sets).

To summarize, we have presented a method to help network security analysts choose visualization techniques which are more appropriate for a particular phase of network monitoring, or a particular class of attack. Our methodology uses elements of user-centered design and expert evaluation techniques to identify key requirements of network analysts for detecting and analyzing threats.

7.1 Limitations

The visualizations suggested in Chapter 5 seem to give analysts a better understanding of the scenario being analyzed. However, many network security problems tend to evolve as attackers adapt to security countermeasures and updated tools. The visualizations we have selected seem to be appropriate for the current state of network security offensive techniques, but we should expect attacks to change in the future. This may lead to the selected visualization tools or approaches becoming obsolete.

In this thesis we have found that there is no all-encompassing visualization tool or technique for all types of attacks. This is because each of the attacks reviewed exploits different aspects of computer networks. We believe that for the vast majority of cases, a single visual representation of a data set will not provide enough information to the analyst. This means that the best way to use visualization for network security is likely to include multiple views of the same data, as explained in Section 2.1.4.

7.2 Future Work

The work presented in this thesis is just a starting point for improving network security visualization. Our work has demonstrated some of the uses of visualization, but also highlighted some of the current problems in using visualization for network security.

Future work may include identifying other common classes of network security attacks and finding the preferred visualization techniques for those as well, which may end up being new techniques that have not yet been developed.

Improvements to our methodology could include the use of expert evaluation techniques such as Nielsen heuristics [68] or cognitive walkthroughs, with a stronger focus on the design and usability of the user interface itself. Combining these techniques might lead to more usable visualization tools that implement insightful visualization techniques.

Bibliography

[1] ARGUS - Auditing Network Activity. http://www.qosient.com/argus/. Retrieved August 23rd, 2009.

[2] Ask Edward Tufte: Sparklines - Theory and practice. http://edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001OR&topic_id=1. Retrieved on May 20th, 2009.

[3] Bro intrusion detection system. http://www.bro-ids.org/ Retrieved August 6th, 2009.

[4] Digital Fears Emerge After Data Siege in Estonia. http://nytimes.com/2007/05/29/technology/29estonia.html?_r=1. May 29th, 2007.

[5] Firewall Log in a Treemap. http://secviz.org/content/firewall-log-a-treemap. Retrieved August 17th, 2009.

[6] IANA IPv6 Unicast Address Assignments. http://iana.org/assignments/ipv6-unicast-address-assignments. Last updated 2008-05-13.

[7] IP address to country. http://www.ip2location.com/. Retrieved August 22nd, 2009.

[8] Port Details - Port 22. SANS Internet Storm Center. http://isc.sans.org/port.html?port=22. Retrieved August 6th, 2009.

[9] Root server attack on 6 February 2007 - ICANN. icann.org/announcements/factsheet-dns-attack-08mar07.pdf. Retrieved August 18th, 2009.

[10] SANS Institute - SANS Top-20. https://www.sans.org/top20/. Retrieved August 6th, 2009.

[11] SiLK Tools. http://tools.netsa.cert.org/silk/. Retrieved May 27th, 2009.

[12] Snort IDS/IPS. http://www.snort.org/. Retrieved July 5th, 2009.

[13] TCPDUMP/LIBPCAP Public Repository. http://www.tcpdump.org/. Retrieved August 23rd, 2009.

[14] Twitter Temporarily Knocked Offline. http://www.pcworld.com/article/170513/twitter_temporarily_knocked_offline_service_now_restored.html. August 20th, 2009.

[15] Walrus - Graph Visualization Tool. http://www.caida.org/tools/visualization/walrus/. Retrieved August 5th, 2009.

[16] Wireshark packet analyzer. http://www.wireshark.org/. Retrieved August 23rd, 2009.

[17] M. Alsaleh, D. Barrera, and P. van Oorschot. Improving security visualization with exposure map filtering. In *2008 Annual Computer Security Applications Conference (ACSAC 2008)*, pages 205–214, Anaheim, California.

[18] D. Barrera and P. van Oorschot. Security Visualization Tools and IPv6 Addresses. Short paper. In *Proceedings of the 6th International Workshop on Visualization for Cyber Security (VizSec)*, Atlantic City, NJ, USA, October 2009.

[19] R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, 1995.

[20] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher. Towards understanding IT security professionals and their tools. In *SOUPS '07: Proceedings of the 3rd Symposium on Usable Privacy and Security*, pages 100–111, New York, NY, USA, 2007. ACM.

[21] CAIDA. Visualizing IPv6 AS-level Internet Topology 2008. http://www.caida.org/research/topology/as_core_network/ipv6.xml, Accessed April 20, 2009.

[22] S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[23] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis.* Wadsworth, 1983.

[24] E. B. Claise. RFC3954 - Cisco Systems NetFlow Services. http://www.ietf.org/rfc/rfc3954.txt, Retrieved on June 3rd, 2009.

[25] G. Conti. *Security Data Visualization.* No Starch Press, San Francisco, CA, USA, 2007.

[26] G. Conti, J. Grizzard, M. Ahamad, and H. Owen. Visual exploration of malicious network objects using semantic zoom, interactive encoding and dynamic queries. In *VizSec '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, pages 83–90, Washington, DC, USA, 2005. IEEE Computer Society.

[27] A. D. D'Amico and K. Whitley. The real work of computer network defense analysts. In Goodall et al. [37], pages 19–37.

[28] S. Deering and R. Hinden. RFC2460 - Internet Protocol, Version 6 (IPv6) Specification. http://www.faqs.org/rfcs/rfc2460.html. Retrieved January 29th, 2009.

[29] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002.

[30] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37:32–64(33), March 1995.

[31] R. F. Erbacher, K. L. Walker, and D. A. Frincke. Intrusion and misuse detection in large-scale systems. *IEEE Computer Graphics and Applications*, 22(1):38–48, 2002.

[32] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. `http://tools.ietf.org/html/rfc2827`, Retrieved on June 6th, 2009.

[33] R. Finkel and J. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[34] F. Fischer, F. Mansmann, D. A. Keim, S. Pietzko, and M. Waldvogel. Large-scale network monitoring for visual analysis of attacks. In *VizSec '08: Proceedings of the 5th International Workshop on Visualization for Computer Security*, pages 111–118, Berlin, Heidelberg, 2008. Springer-Verlag.

[35] B. Fry. *Visualizing Data.* Oreilly & Associates Inc, 2008.

[36] C. Gates and J. McHugh. The Contact Surface: A Technique for Exploring Internet Scale Emergent Behaviors. In *DIMVA '08: Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 228–246, Berlin, Heidelberg, 2008. Springer-Verlag.

[37] J. R. Goodall, G. J. Conti, and K.-L. Ma, editors. *VizSec 2007, Proceedings of the Workshop on Visualization for Computer Security*, Mathematics and Visualization. Springer, 2008.

[38] J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi. Focusing on context in network traffic analysis. *IEEE Comput. Graph. Appl.*, 26(2):72–80, 2006.

[39] J. R. Goodall, A. A. Ozok, W. G. Lutters, P. Rheingans, and A. Komlodi. A user-centered approach to visualizing network traffic for intrusion detection. In *CHI '05: Extended abstracts on Human Factors in Computing systems*, pages 1403–1406, New York, NY, USA, 2005. ACM.

[40] J. B. Grizzard, J. Charles R. Simpson, S. Krasser, H. L. Owen, and G. F. Riley. Flow based observations from NETI@home and honeynet data. In *Proceedings from the Sixth IEEE Systems, Man and Cybernetics Information Assurance Workshop*, pages 244–251, June 2005.

[41] S. H. Gunderson. Measuring the current state of IPv6 for ordinary users. `http://www.ripe.net/ripe/meetings/ripe-57/presentations/Colitti-Global_IPv6_statistics_-_Measuring_the_current_state_of_IPv6_for_ordinary_users_.7gzD.pdf`, October, 2008.

[42] C. G. Healey. Perception in visualization. http://www.csc.ncsu.edu/faculty/healey/PP/index.html . Retrieved August 4th, 2009.

[43] M. Hearst and D. Rosner. Tag Clouds: Data Analysis Tool or Social Signaller? In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*. IEEE Computer Society Washington, DC, USA, 2008.

[44] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 296–304, 1990.

[45] R. Hinden, M. O'Dell, and S. Deering. RFC2374 - An IPv6 Aggregatable Global Unicast Address Format. http://www.faqs.org/rfcs/rfc2374.html, July 1998.

[46] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[47] G. Huston. IPv4 Address Report. Retrieved on April 14, 2009 from http://www.potaroo.net/tools/ipv4/index.html.

[48] IETF. RFC 1122 - Requirements for Internet Hosts – Communication Layers. http://tools.ietf.org/html/rfc1122. Retrieved June 14th, 2009.

[49] Information Sciences Institute - University of Southern California. RFC791 - Internet Protocol. http://www.faqs.org/rfcs/rfc791.html, Sep. 1981.

[50] B. Irwin and J. P. Riel. Using InetVis to Evaluate Snort and Bro Scan Detection on a Network Telescope. In J. R. Goodall, G. Conti, and K. L. Ma, editors, *VizSec 2007: Proceedings of the Workshop on Visualization for Computer Security*, pages 255–273. Springer, 2008.

[51] J. Janies. Existence Plots: A Low-Resolution Time Series for Port Behavior Analysis. In *VizSec 2008: Proceedings of the 5th International Workshop on Visualization for Computer Security*, pages 161–168, Berlin, Heidelberg, 2008. Springer-Verlag.

[52] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy 2004*, Oakland, CA, May 2004.

[53] G. Killcrece, K. Kossakowski, and R. Ruefle. Zajicek Mark (2003a): State of the Practice of Computer Security Incident Response Teams (CSIRTs). Technical report, CMU/SEI-2003-TR-001, Pittsburgh, PA, USA.

[54] H. Koike, K. Ohno, and K. Koizumi. Visualizing cyber attacks using IP matrix. In *IEEE Workshop on Visualization for Computer Security, 2005. (VizSec 05)*, pages 91–98, Oct. 2005.

[55] K. Lakkaraju, W. Yurcik, and A. J. Lee. NVisionIP: Netflow visualizations of system state for security situational awareness. In *VizSec/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 65–72, New York, NY, USA, 2004.

[56] S. Lau. The spinning cube of potential doom. *Communications of ACM*, 47(6):25–26, 2004.

[57] E. Le Malécot, Y. Hori, K. Sakurai, J.-C. Ryou, and H. Lee. (Visually) Tracking Distributed SSH Brute Force Attacks? In *Proceedings of the 3rd International Joint Workshop on Information Security and its Applications (IJWISA'08)*, pages 1–8, Seoul, South Korea, 2008.

[58] E. Le Malécot, M. Kohara, Y. Hori, and K. Sakurai. Interactively combining 2D and 3D visualization for network traffic monitoring. In *VizSec '06: Proceedings of the 3rd International Workshop on Visualization for Computer Security*, pages 123–127, New York, NY, USA, 2006. ACM.

[59] R. Lengler and M. Eppler. Towards a periodic table of visualization methods for management. *Proceedings of Graphics and Visualization in Engineering (GVE 2007)*, 2007.

[60] J. Leyden. US credit card firm fights DDoS attack. `http://www.theregister.co.uk/2004/09/23/authorize_ddos_attack/`. September 23rd, 2004.

[61] Y. Livnat, J. Agutter, S. Moon, and S. Foresti. Visual correlation for situational awareness. In *Proceedings of the IEEE Symposium on Information Visualization, 2005*, pages 95–102, Oct. 2005.

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

[63] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. PortVis: a tool for port-based detection of security events. In *VizSec/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 73–81, New York, NY, USA, 2004. ACM.

[64] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. *Internet Denial of Service: Attack and Defense Mechanisms.* Prentice Hall, 2005.

[65] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *Security & Privacy Magazine, IEEE*, 1(4):33–39, July-Aug. 2003.

[66] D. Moore, C. Shannon, and k. claffy. Code-Red: a case study on the spread and victims of an internet worm. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, pages 273–284, New York, NY, USA, 2002. ACM.

[67] J. Muir and P. van Oorschot. Internet Geolocation: Evasion and Counter-evasion. *ACM Computing Surveys*, to appear. 2009.

[68] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 249–256, New York, NY, USA, 1990. ACM.

[69] J. Oberheide, M. Goff, and M. Karir. Flamingo: Visualizing Internet Traffic. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 150–161, April 2006.

[70] S. Panjwani, S. Tan, and K. M. Jarrin. An experimental evaluation to determine if port scans are precursors to an attack. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 602–611, Washington, DC, USA, 2005. IEEE Computer Society.

[71] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, 2001.

[72] J. Pearlman and P. Rheingans. Visualizing network security events using compound glyphs from a service-oriented perspective. In Goodall et al. [37], pages 131–146.

[73] B. Schneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages, IEEE Computer Society*, pages 336–343, Washington, DC, 1996.

[74] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. In *Proceedings of IEEE Symposium on Information Visualization. INFOVIS*, pages 73–78, 2001.

[75] J. M. Shu Nakamae, Yuji Sekiya. A Study Into a Visualization of an IPv6 Network. In *Proceedings of the 9th Annual Conference of the Internet Society INET'99*, 1999.

[76] T. Taylor, D. Paterson, J. Glanfield, C. Gates, S. Brooks, and J. McHugh. FloVis: Flow Visualization System. In *Conference For Homeland Security. CATCH '09. Cybersecurity Applications & Technology*, pages 186–198, March 2009.

[77] S. T. Teoh, K. L. Ma, S. F. Wu, and X. Zhao. Case study: Interactive Visualization for Internet Security. In *VIS '02: Proceedings of the Conference on Visualization*, pages 505–508, Washington, DC, USA, 2002. IEEE Computer Society.

[78] E. R. Tufte. *Envisioning Information.* Graphics Press, 1990.

[79] E. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, pages 664–675, 1990.

[80] C. Wharton, J. Rieman, C. Lewis, and P. Polson. *The Cognitive Walkthrough Method: A Practitioner's Guide.* John Wiley & Sons, Inc., New York, NY, USA, 1994.

[81] D. Whyte. Network scanning detection strategies for enterprise networks. *PhD thesis, Carleton University*, 2008.

[82] D. Whyte, P. van Oorschot, and E. Kranakis. Tracking Darkports for Network Defense. In *Proceedings of the 23rd Annual Computer Security Applications Conference, ACSAC 2007*, pages 161–171, 2007.

[83] D. Whyte, P. C. van Oorschot, and E. Kranakis. Exposure maps: removing reliance on attribution during scan detection. In *HotSec'06: Proceedings of the 1st USENIX Workshop on Hot Topics in Security*, Berkeley, CA, USA, 2006. USENIX Association.

[84] W. Yurcik, J. Barlow, K. Lakkaraju, and M. Haberman. Two Visual Computer Network Security Monitoring Tools Incorporating Operator Interface Requirements. In *ACM CHI Workshop on Human-Computer Interaction and Security Systems (HCISEC)*, 2003.