

John R. Goodall  
Gregory Conti  
Kwan-Liu Ma (Eds.)

LNCS 5210

# Visualization for Computer Security

5th International Workshop, VizSec 2008  
Cambridge, MA, USA, September 2008  
Proceedings

 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

John R. Goodall Gregory Conti  
Kwan-Liu Ma (Eds.)

# Visualization for Computer Security

5th International Workshop, VizSec 2008  
Cambridge, MA, USA, September 15, 2008  
Proceedings

Volume Editors

John R. Goodall  
Secure Decisions division of Applied Visions  
Albany, NY, USA  
E-mail: johng@securedesigns.avi.com

Gregory Conti  
United States Military Academy  
West Point, NY, USA  
E-mail: gregory.conti@usma.edu

Kwan-Liu Ma  
University of California  
Davis, CA, USA  
E-mail: ma@cs.ucdavis.edu

Library of Congress Control Number: 2008934071

CR Subject Classification (1998): C.2, K.6.5, I.3, H.3.3, I.5.3

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743  
ISBN-10 3-540-85931-4 Springer Berlin Heidelberg New York  
ISBN-13 978-3-540-85931-4 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media  
springer.com

© Springer-Verlag Berlin Heidelberg 2008  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12515677 06/3180 5 4 3 2 1 0

# Preface

This volume contains the papers presented at VizSec 2008, the 5th International Workshop on Visualization for Cyber Security, held on September 15, 2008 in Cambridge, Massachusetts, USA. VizSec 2008 was held in conjunction with the 11th International Symposium on Recent Advances in Intrusion Detection (RAID).

There were 27 submissions to the long and short paper categories. Each submission was reviewed by at least 2 reviewers and, on average, 2.9 program committee members. The program committee decided to accept 18 papers.

The program also included an invited talk and a panel. The keynote address was given by Ben Shneiderman, University of Maryland at College Park, on the topic Information Forensics: Harnessing Visualization to Support Discovery. The panel, on the topic The Need for Applied Visualization in Information Security Today, was organized and moderated by Toby Kohlenberg from Intel Corporation.

July 2008

John R. Goodall

# Conference Organization

## Program Chairs

John R. Goodall	Secure Decisions division of Applied Visions
Gregory Conti	United States Military Academy
Kwan-Liu Ma	University of California at Davis

## Program Committee

Stefan Axelsson	Blekinge Institute of Technology
Richard Bejtlich	General Electric
Kris Cook	Pacific Northwest National Laboratory
David Ebert	Purdue University
Robert Erbacher	Utah State University
Deborah Frincke	Pacific Northwest National Laboratory
Carrie Gates	CA Labs
John Gerth	Stanford University
Barry Irwin	Rhodes University
Daniel Keim	University of Konstanz
Toby Kohlenberg	Intel Corporation
Stuart Kurkowski	Air Force Institute of Technology
Kiran Lakkaraaju	University of Illinois at Urbana-Champaign
Raffael Marty	Splunk
Douglas Maughan	Department of Homeland Security
John McHugh	Dalhousie University
Penny Rheingans	UMBC
Lawrence Rosenblum	National Science Foundation
George Tadda	Air Force Research Lab
Daniel Tesone	Applied Visions
Alfonso Valdes	SRI International
Kirsten Whitley	Department of Defense

## Local Organization

Robert K. Cunningham	Lincoln Laboratory
----------------------	--------------------

# Table of Contents

Visual Reverse Engineering of Binary and Data Files . . . . .	1
<i>Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster</i>	
Effective Visualization of File System Access-Control . . . . .	18
<i>Alexander Heitzmann, Bernardo Palazzi, Charalampos Papamanthou, and Roberto Tamassia</i>	
Visual Analysis of Program Flow Data with Data Propagation . . . . .	26
<i>Ying Xia, Kevin Fairbanks, and Henry Owen</i>	
A Term Distribution Visualization Approach to Digital Forensic String Search . . . . .	36
<i>Moses Schwartz and L.M. Liebrock</i>	
GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool . . . . .	44
<i>Leevar Williams, Richard Lippmann, and Kyle Ingols</i>	
A Graph-Theoretic Visualization Approach to Network Risk Analysis . . .	60
<i>Scott O'Hare, Steven Noel, and Kenneth Prole</i>	
Improving Attack Graph Visualization through Data Reduction and Attack Grouping . . . . .	68
<i>John Homer, Ashok Varikuti, Xinming Ou, and Miles A. McQueen</i>	
Show Me How You See: Lessons from Studying Computer Forensics Experts for Visualization . . . . .	80
<i>T.J. Jankun-Kelly, Josh Franck, David Wilson, Jeffery Carver, David Dampier, and J. Edward Swan II</i>	
A Task Centered Framework for Computer Security Data Visualization . . . . .	87
<i>Xiaoyuan Suo, Ying Zhu, and Scott Owen</i>	
BGPeep: An IP-Space Centered View for Internet Routing Data . . . . .	95
<i>James Shearer, Kwan-Liu Ma, and Toby Kohlenberg</i>	
Large-Scale Network Monitoring for Visual Analysis of Attacks . . . . .	111
<i>Fabian Fischer, Florian Mansmann, Daniel A. Keim, Stephan Pietzko, and Marcel Waldvogel</i>	
Visualizing Real-Time Network Resource Usage . . . . .	119
<i>Ryan Blue, Cody Dunne, Adam Fuchs, Kyle King, and Aaron Schulman</i>	

Wireless Cyber Assets Discovery Visualization .....	136
<i>Kenneth Prole, John R. Goodall, Anita D. D'Amico, and Jason K. Kopylec</i>	
NetFlow Data Visualization Based on Graphs .....	144
<i>Pavel Minarik and Tomas Dymacek</i>	
Backhoe, a Packet Trace and Log Browser .....	152
<i>Sergey Bratus, Axel Hansen, Fabio Pellacini, and Anna Shubina</i>	
Existence Plots: A Low-Resolution Time Series for Port Behavior Analysis .....	161
<i>Jeff Janies</i>	
Using Time Series 3D AlertGraph and False Alert Classification to Analyse Snort Alerts .....	169
<i>Shahruhniza Musa and David J. Parish</i>	
Network Traffic Exploration Application: A Tool to Assess, Visualize, and Analyze Network Security Events .....	181
<i>Grant Vandenberghe</i>	
<b>Author Index</b> .....	197



# Visual Reverse Engineering of Binary and Data Files

Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster

Department of Electrical Engineering and Computer Science  
United States Military Academy  
West Point, New York  
{gregory.conti,erik.dean,matthew.sinda,  
benjamin.sangster}@usma.edu

**Abstract.** The analysis of computer files poses a difficult problem for security researchers seeking to detect and analyze malicious content, software developers stress testing file formats for their products, and for other researchers seeking to understand the behavior and structure of undocumented file formats. Traditional tools, including hex editors, disassemblers and debuggers, while powerful, constrain analysis to primarily text based approaches. In this paper, we present design principles for file analysis which support meaningful investigation when there is little or no knowledge of the underlying file format, but are flexible enough to allow integration of additional semantic information, when available. We also present results from the implementation of a visual reverse engineering system based on our analysis. We validate the efficacy of both our analysis and our system with case studies depicting analysis use cases where a hex editor would be of limited value. Our results indicate that visual approaches help analysts rapidly identify files, analyze unfamiliar file structures, and gain insights that inform and complement the current suite of tools currently in use.

## 1 Introduction

Individual files are a fundamental component of today's computing paradigm as well as one of today's biggest threat vectors. With the advent of effective network security devices based upon firewalls, intrusion detection systems and similar security applications, attackers are moving away from network protocol attacks and toward attacking applications themselves. This transition is problematic because firewalls must pass some traffic in order to provide services to their users, particularly web and email. It is through these services that users send, receive, upload and download files, sometimes as email attachments, web downloads, or more worrisome, surreptitiously through encrypted channels such as HTTPS or SSH. The problem is worsened by the rapid evolution of file-based attacks that exploit vulnerabilities in parsing by applications and common software libraries, as well as by the attacker's use of packers which obfuscate the contents of files.

Legitimate files function as either stand alone executable programs or as data to be used by other applications, such as word processors, text editors or graphics programs.

Executable files are executed by the operating system, whereas, data files are loaded by applications. In both cases the operating system and application assume some knowledge of the file's underlying format and structure in order to operate on it successfully. In these legitimate cases, the user of the file only sees the interpreted version of the file as determined by the application or operating system. The underlying structure of the file is hidden. It is through this hidden nature that both attackers and legitimate programmers attempt to place security mechanisms. While file extensions and magic numbers serve the purposes of legitimate use, security analysts are forced to dig much deeper and face the frequent task of exploring the raw structure of files to detect modifications, determine file type, determine what a file does, determine authorship, create virus signatures, and detect code evolution, among other tasks. Further hampering analysis is that the file may be corrupted, obfuscated, or encrypted to slow analysis. In some cases, files contain code designed to crash analytic tools or function differently if used in a virtual machine. Sometimes files contain largely legitimate data with a small fraction containing a malicious aspect and others are dedicated malicious applications.

At their heart, files are just binary objects whose meaning is based on the applications or the operating systems that use them. However, in some cases little will be known about a given file. We therefore view the problem of reverse engineering binary files as having two levels, a context independent level where we assume the analyst has little or no knowledge about the structure and purpose of a given file, and analysis must occur initially in a context-independent manner. At the second level, the analyst knows some information, such as an expected file format, and can make informed assumptions as they analyze the file. The difference between these two levels is evident in the current suite of tools for reverse engineering files. At the context independent level, the analyst employs general purpose tools such as hex editors and byte frequency analysis to gain insight. At the semantic level, analyst tools are crafted specifically to a given file format or family of related formats, such as disassemblers and debuggers, leading to very precise insights.

Unfortunately, the vast majority of best of breed tools for reverse engineering of files are strictly text based and provide only a tiny viewport into the file under analysis. Visualization is underutilized in reverse engineering and bears great promise for assisting analysts in their work and augmenting their existing tool suite. In this paper we present an analysis of user tasks that is useful for the design of visualization systems, and employ this analysis to implement a system which combines proven text based techniques with innovative visualization approaches in order to augment the analyst and complement their tool suite for a number of essential tasks. We validate the efficacy of these contributions through several case studies. It is important to note that we are addressing the problem of reverse engineering of binary file formats across the entire range of possibilities and not focusing specifically on the special case of reverse engineering of executable files.

This paper is organized as follows. Section 2 places our research in the field of related work. Section 3 contains a user-level requirements analysis that we used to guide our development. Section 4 presents our system design and implementation. Sections 5 demonstrate the utility of our approach through case studies. Section 6 presents our conclusions and suggested directions for future work.

## 2 Related Work

The most commonly employed tool for reverse engineering of files and file formats is a hex editor, which typically displays files in hexadecimal and ASCII formats and assumes no knowledge of the underlying file structure. Traditional hex editors offer basic functionality, but modern hex editors contain advanced features including the ability to view, search, and convert between hex, ASCII, Unicode, decimal and floating point data types, among others. Such tools also include the ability to encrypt and decrypt, calculate checksums, encode and decode, calculate computer hashes, and compress and decompress blocks of data within a file. Navigation is straightforward, including the use of scroll bars and the ability to jump to a given offset within the file, but also includes the ability to place navigation labels at user specified locations. The Hiew hex editor is noteworthy because it integrates an assembler and disassembler. Other sophisticated editors, such as WinHex, also include the ability to edit memory, create a byte frequency histogram of both the entire file and a user selected region, and easily link to helper programs such as web browsers and media viewers. The 010 editor also includes binary templates which parse popular binary file formats into their associated variables and data structures. Hex editors are the definitive tool for text-based analysis of binary files and include powerful computation capabilities, even scripting, but lack the ability to provide big picture context, a significant problem due to the complexity of even moderately sized files.

Beyond hex editors, there are other general purpose tools for reverse engineering files of both executable and data formats, including the command line *strings* utility which outputs contiguous runs of printable ASCII. Similarly, the command line *diff* command, the *fc* (file compare command) and related variants allow the comparison of text and binary files in order to locate changes. Recently more powerful GUI-based tools have emerged, such as Compare It!, ExamDiff, Guiffy, Merge, Meld, and WindDiff which allow side-by-side comparison of files. Common attributes of this class of tools include side-by-side views in text or hexadecimal representations, scripting, directory comparison, reporting tools, syntax highlighting, the ability to detect changes (diffing) and to merge files. Compare It! and Merge also provide the ability to compare images. As is the case with hex editors, command line and GUI-based file comparison tools are text based and lack the ability to provide big-picture context that visualization can provide. However, there are several notable exceptions. The first is Visual Insights Difference Viewer, which combines the two-pane textual view with a two column graphical view similar to Eick's Seesoft technique to provide context when comparing two files, but this tool is apparently no longer available. The 010 editor also uses a similar combined text and Seesoft-like view technique. Nwdiff is another interesting approach. It plots pixels based on the actual bit values contained within a pair of files and uses four graphical panes. Two of the panes show the raw structure of the files; the other two panes graphically show similarities and differences between the files by using a logical or and xor. BinaryViewer and RUMINT's binary rainfall view [1] use similar bit-level views, to view binary files and network packets, respectively. Another, albeit rudimentary, technique for visualizing binary files is the *raw2tiff* program. *raw2tiff* converts raw byte streams to the tiff image format. Designed for image file conversion,

raw2tiff produces similar results by converting a binary file to a tiff image. We believe these bit and byte-level approaches bear great promise in helping analyze binary files; we will discuss these techniques later in the paper.

Other interesting approaches for visualizing binary files include Kaminsky's use of dot plot [2] patterns to explore self-similarity in binary files as well as his use of context free grammars to highlight hex dumps [3]. We have incorporated Kaminsky's dot plot technique into our system as we will discuss later in the paper.

As part of considering tools for analyzing individual files, or pairs of files as in the case of diffing, it is also useful to examine existing tools for visually displaying complete file systems. These include Firelight which uses concentric segmented rings, SequoiaView and GdMap which uses squarified treemaps, KDirStat, Baobab and WinDirStat combine a treemap with a textual tabular view in the same display window.

The file fuzzing community also employs tools designed to manipulate binary files in order to identify vulnerabilities in application parsing algorithms. Popular examples include SPIKEFILE for Linux and FILEFUZZ for Windows. Such tools are primarily text based, but would benefit from the interactive visualization techniques we present later in the paper to assist in identifying promising locations in files that users are attempting to fuzz.

So far, we've discussed related work regarding the analysis of binary files in general, however it is worth specifically discussing the special case of reverse code engineering (RCE) which focuses on the analysis of executable files. There are several ways to approach RCE. The first is static analysis, the examination of a file's contents without executing it. The second is dynamic analysis which studies the internal operation of the code as it is executing. Another approach is to execute the program and study how it interacts with the operating system and network. The primary tools are hex editors, disassemblers with IDA Pro being the best of breed, debuggers including tools like OllyDbG and SoftIce, and decompilers. It is important to note that IDA Pro allows user created plug-ins as well as scripting, and as a result, there is an active developer community surrounding IDA Pro, such as OpenRCE and IDA Palace. The wide range of *binutils* is often employed, including *objdump* which displays information about object files and *readelf* which displays information about ELF format object files. Tools from the Sysinternal's tool suite augment debuggers and decompilers, by allowing fine grained monitoring of an application's interaction with the operating system as it executes.

There has been a limited amount of work in the visualization of executable files. Yoo used self-organizing maps to detect viruses [4]. The most current work is found in the Zynamics' BinDiff, BinNavi, and VxClass tools which utilize directed graphs. Graph-based techniques have demonstrated great utility in analyzing malicious software [5, 6, 7, 8] including diffing of executable files [9, 10].

The novelty of our work springs from our analysis of reverse engineering tasks, novel visualizations, and our system for analyzing binary files. While there has been a great deal of work on text-based analysis of files, there is only limited work of visualization of files themselves, primarily only executable files, at both the context-independent and context-dependent levels.

### 3 Requirements Analysis

Reverse engineering of file formats is both an art and a science. As such, many analysts develop their own personal approaches to reverse engineering. These ill-defined individual approaches are a significant challenge when attempting to design systems that facilitate the work of many different users. To overcome this shortcoming and ground our work in real-world user requirements we analyzed five different approaches found in: *Fuzzing – Brute Force Vulnerability Discovery* by Sutton, Greene and Amini, Wikibooks' *Reverse Engineering*, *Hacker Disassembling Uncovered* by Kaspersky, *Hacking – The Art of Exploitation* by Erickson, and *Hack Proofing Your Network* by Russell et al. While each text provided unique approaches to reverse engineering, we did find significant commonality. In addition, to supplement our

**Table 1.** Scenarios which require low-level analysis of files

Goal	Examples
Analyze undocumented file format	<ul style="list-style-type: none"> <li>- Classify basic purpose of file</li> <li>- Understand structure of file format</li> <li>- Understand behavior of creating application</li> <li>- Identify algorithms used for compression, encoding and encryption within file</li> </ul>
Audit files for vulnerabilities	<ul style="list-style-type: none"> <li>- Locate structures for targeted fuzzing</li> <li>- Identify vulnerable code structures</li> <li>- Locate caves (empty regions within file)</li> </ul>
Compare files (Diffing)	<ul style="list-style-type: none"> <li>- Create signature of malware variant</li> <li>- Determine purpose of a patch</li> <li>- Track evolution of code between file versions</li> </ul>
Cracking <sup>1</sup>	<ul style="list-style-type: none"> <li>- Break copy protection</li> <li>- Alter player resources in game (e.g. gold pieces)</li> </ul>
Cryptanalysis	<ul style="list-style-type: none"> <li>- Validate protocol or algorithm operation</li> <li>- Confirm encryption occurred</li> <li>- Gain insight into encryption algorithm</li> <li>- Find key or password</li> <li>- Analyze files for steganographic content</li> </ul>
Forensic analysis	<ul style="list-style-type: none"> <li>- Determine authorship</li> <li>- Locate and extract metadata</li> <li>- Locate and extract hidden content</li> <li>- Identify compiler, language or application used to create file</li> </ul>
Identify unknown file format	<ul style="list-style-type: none"> <li>- Precisely determine application which created file</li> <li>- Classify type of application which created file</li> </ul>
Malware analysis	<ul style="list-style-type: none"> <li>- Reverse engineer file's function</li> <li>- Create antivirus or IDS signature</li> <li>- Locate malicious code within file</li> </ul>
Reporting	<ul style="list-style-type: none"> <li>- Create analyst annotated reports</li> <li>- Share analytic results with analysts, management, and customers.</li> </ul>

<sup>1</sup> While we don't support file cracking, it is nonetheless a scenario which requires low-level analysis of files and was repeatedly mentioned in our literature review.

**Table 2.** Representative reverse engineering high-level tasks

Category	Task
Analyze	-Identify and analyze non-standard file formats and algorithms -Understand, annotate and document the file's structure, including header/footer and block/record/field formats -Test and evaluate hypothesis as to the meaning of the data and file format
Calculate	-Perform decimal, hexadecimal, and binary calculations -Encrypt and decrypt, encode and compress blocks of values within a file, calculate checksums
Compare	-Compare two or more files and precisely locate differences.
Explore	-Understand the big picture context of a file's structure -Identify major structures within a file
Filter	-Remove undesired content
Identify	-Identify which algorithms and libraries were used -Identify and analyze regions containing executable code and data -Identify in-file references to data
Locate	-Locate regions that have been encoded, compressed or encrypted -Locate free/slack space
Modify	-Edit values within files -Fill regions with desired values -Load and save text and binary files
Navigate	-Easily navigate to regions within the file
Report	-Generate report of analysis
Search	-Locate specific values or sequences of values, including those in hex, floating point, binary, decimal, ASCII and Unicode representations.
Semantics	-Correctly parse binary file formats -Apply external knowledge of file structure and format to gain additional insight
View	-View files and regions in native viewers/formats, including assembly -View/convert values in native format/encodings/datatypes/byte orders (e.g. 2 and 4 byte integers, floats, strings, Unicode, real and string, signed and unsigned)

literature review of user requirements we conducted semi-structured interviews of 12 intermediate and advanced security analysts from the academic, industry and hacker communities, conducting the interviews primarily at the RSA 2008 and Shmoocon 2008 Conferences. From the results of our literature review and interviews we compiled scenarios which require low-level analysis of files, see Table 1, and categorized specific tasks analysts face when seeking to reverse engineer data and executable files, see Table 2.

When faced with an unfamiliar file, the analyst will also employ common command line utilities such as *strings*, which looks for sequences of ASCII characters contained within binary files, *file*, which attempts to identify a file's format. The next step is often to load the file in a hex editor and scroll the textual display looking for regions of interest. In the case of an executable file, the analyst will likely run the file and observe its interactions with the underlying operating system and network using tools which monitor system calls, network activities, file accesses, and registry changes. The analyst employs debuggers and disassemblers to understand the code in operation.

When the file is untrusted, analysis will almost certainly be conducted on an isolated malware analysis workstation, usually in a virtual machine environment to provide additional isolation. Depending on the analyst's objective, the machine may have network connectivity. Reverse engineering of both executable and data files is, in many ways, an adversarial relationship. For example, there is an increasing trend by malware authors to attempt to detect virtual machine environments and behave in an unexpected manner, such as crashing the debugger, to frustrate analysis. File extensions and other metadata, particularly in forensic analysis, are not fully trusted by the analyst. The designers of the file format will often go to great lengths to obfuscate file contents by using encryption, packing, or obfuscated coding techniques. There are legal issues as well. In some cases attempting to reverse engineer file formats, particularly when encrypted or deliberately obfuscated, can be considered a violation of intellectual property rights.

## 4 System Design and Implementation

There are a number of situations that necessitate low-level analysis of files and file formats, but they fall into two main categories: context independent analysis when little is known about a given file's format and semantic analysis where the analyst knows some information about the structure of the file. For our work we have chosen to design our system to facilitate context independent analysis where a hex editor and command line tools, such as *strings*, would be used. These include analyzing undocumented file formats, auditing files for fuzzing opportunities, and forensic analysis. More specifically, we designed our system to aid rapid analysis, provide big picture context, facilitate navigation, and assist identification of internal structures contained within files as we believe these are promising areas for visual support. We leave semantic analysis and other user tasks for future work. That being said, an understanding of file formats in general, is critically important even in the context independent case.

Visualization of files allows the analyst to see structures within files and it is useful to study file formatting techniques. File formats come in a myriad of different types, from extremely simple to highly complex. While it is impossible to determine the exact number of different file formats, the popular FILEExt database of file extensions currently tracks 24,048 different types and Wotsit, a leading file and data format website, provides information on 1,030 different publicly available and closed file formats. The end result is an environment with wide variety of commonly employed techniques as well as the likely possibility of unique file formats written by individual authors.

Common file structuring techniques include embedding metadata (e.g. serial numbers and magic numbers), storing fixed and variable length records, compressing and encrypting regions within a file, embedding images, as well as various approaches to storing and encoding strings, integers and floating point values.

Analysis of files needn't be constrained to data contained within the file itself, but could incorporate external information stored by the operating system, such as file name, file size, date of creation, date of modification. Similarly, a visualization system may employ a wide range of statistical techniques to add meaning to the visualizations, assist filtering, and aid navigation within the file, such as calculating the frequency of

bytes, calculating entropy, and performing n-gram analysis. Such calculations could occur across the entire file, or be constrained to a given window selected by an algorithm or end user.

We implemented our system using C# in Microsoft Visual Studio .NET 2005. We chose this environment because C# is a robust and comprehensive language and because of Visual Studio's strength in rapid GUI development. All testing was done on a commodity PC (Dual Core AMD 2500, 1GB RAM, Windows XP). For future work, we plan to explore implementing the system, including all interactive GUI elements, in a platform independent language such as Java, Perl or Python. As malware analysis often occurs inside virtual machines, it is also important that future versions perform well in such an environment.

#### 4.1 System Design Goals

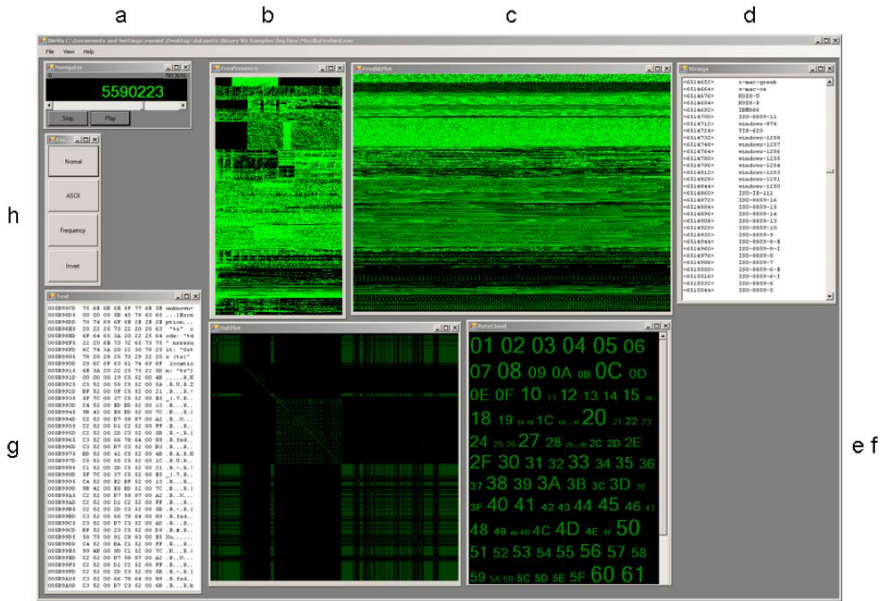
Given our analysis of user requirements and the environment in which users operate, we created the following design goals to guide our development. These goals, are just that, goals. Later sections in the paper will demonstrate which we accomplished in our current system implementation [11].

- *Useful* – Allow user to gain useful insight about the file, including big picture structure, embedded objects, obfuscated or hidden data, malicious content, and embedded metadata.
- *Ease of use* – The application should be easy to install, understand, and operate.
- *Extensible* – A small group of developers cannot compete with the ingenuity of an entire user base. An extensible design allows the open source community to develop plug-ins.
- *Incorporate best practices* – Don't rediscover fire. Create a design that can incorporate best practices from existing tools.
- *Open source* – In order to gain trust of our security conscious user base, releasing the source code helps increase adoption.
- *Context independent analysis* – Provide valuable insight into binary files, even if the underlying file format is unknown.
- *Semantic file analysis* – Incorporate relevant semantic information into the visualizations when file format is known or suspected.
- *Multiple coordinated views* – Provide useful windows into the file that complement existing textual tools.
- *Attack resistant* – Design the tool with the understanding that it may be attacked by a malicious file under analysis.
- *Platform independence* – The ideal system should function when used on all major operating systems employed by users.

#### 4.2 Visualization Design

Our system incorporated both textual and graphical visualization techniques in order to combine the functionality of command line tools and best practices from hex editors with insightful visualizations. In its current implementation, the system incorporates two





**Fig. 1.** System screenshot depicting each of the visualization techniques

textual views. The first view is the canonical hex/ASCII view commonly employed by hex editors and hex dump command line utilities, see Figure 1(g). While we only included a hex viewer window, a key idea is that a hex editor can be incorporated *in its entirety* into the design we propose. The second textual view displays ASCII strings contained within the file, see Figure 1(d). Both displays include the offset of the data displayed. The system includes a number of graphical displays which are described in the following sections. It is important to note that we view our textual and graphical displays as a starting point. Our ultimate aim is to create an extensible architecture that would inspire end users to create and share additional visualizations.

### 4.3 Byteview Visualization

The system includes four graphical displays<sup>2</sup>, the first is a byte plot visualization, see Figure 1(c), which maps each byte in the file to a pixel on the display. The first byte in the file is located in the top left corner, coordinate (1,1), the next byte is displayed at position (2,1). The byteview is 640x480 resolution, so each row can display 640 bytes. When the end of the line is reached, plotting begins at the next line below. At 640x480 resolution, the byteview visualization can display 307,200 bytes. Thus byte 307,200 will be displayed at coordinate (640,480). The color of each pixel maps to the value of the byte displayed, where a byte value of 00 would be black and FF would be bright green. We chose 640x480 resolution because its relatively small size would facilitate rapid drawing. In addition we believe choosing resolutions in multiples of 32

<sup>2</sup> The following sections describe three displays, the fourth, the Byte Map display (Figure 1(f)) alters font size based on byte frequency, but is still under development.

is important when analyzing files written for 32-bit PCs as many structures contained within files are multiples of 32. When testing performance we found that the display could be updated in 0.03 seconds, leaving open the possibility of creating byteview visualizations at greater resolutions while still providing a responsive interface.<sup>3</sup>

#### 4.4 Byte Presence Visualization

The byte presence visualization, see Figure 1(b), consists of 256 columns. Each row displays the presence and absence of byte values within a given window in the file being examined. This visualization is designed to act in concert with the byteview display and each of the 480 rows from the byteview visualization is displayed as a corresponding row in the byte presence display. For example, if the eighth row of the byteview contains only byte values in the printable ASCII range (i.e. 32-127) the eighth row of the byte presence visualization will have pixels in the 32<sup>nd</sup> through 127<sup>th</sup> columns illuminated. By designing these two visualizations to act in concert, an analyst is able to perform side-by-side comparison of a given region of interest. The byte presence visualization is particularly useful for identifying regions of text contained within a file (seen as vertical bars in columns 32-127), for detecting regularly changing byte values in the file (seen as diagonal lines, where the slope equals the direction and rate of change), for identifying regions of compression or encryption (seen as a nearly complete horizontal line), as well as for detecting the set of characters used by an encoding scheme, such as *uuencoding* which uses a subset of printable ASCII characters. Our current implementation indicates the presence or absence of a given byte value, a possible future enhancement is to use color to highlight bytes based on frequency or entropy.

#### 4.5 Dot Plot Visualization

The dot plot visualization, see Figure 1(e) is a powerful visualization technique used by bioinformatics researchers to measure self-similarity. Kaminsky demonstrated that the technique is also useful for the analysis of binary data, particularly for visually detecting repeated sequences of bytes contained within a file [12]. Due to the promise of Kaminsky's results<sup>4</sup>, we included a dotplot visualization in our implementation. Kaminsky's dot plot works by creating a matrix out of a sequence of bytes from the file. Similarly, in our system we used the file under analysis for labeling both the horizontal and vertical axes. Pixels in the display are illuminated at all locations where the horizontal and vertical axes values are identical. Note that the algorithm may also be used to compare two different byte sequences, such as two different files, and visually indicate each difference. In this case, one axis is labeled with the first file and the other axis is labeled with the second file. The dot plot algorithm is  $O(n^2)$ , thus plotting a full 1MB file would create a 1TB image, beyond the power of desktop workstations. To overcome this shortcoming, we implemented a 500x500 dot plot as

---

<sup>3</sup> We were able to achieve this level of performance by avoiding C#'s `GetPixel` and `SetPixel` methods and directly accessing image memory, see <http://www.bobpowell.net/lockingbits.htm> for more information.

<sup>4</sup> Note that Kaminsky's approach was not interactive. He generated extremely large dot plots of entire files. Our approach is interactive.

a tradeoff between functionality and processing requirements. As the user navigates the file, the dot plot is redrawn using a 500 byte window from the current offset onward. A full description of the dot plot is beyond the scope of this paper, for more information see Helfman as an introduction [15].

## 4.6 Navigation and Interaction Design

Navigation in our system is designed to be simple and intuitive, applying multiple coordinated visualization windows, both graphical and textual. It is accomplished via a small VCR-like display, Figure 1(a). The analyst may navigate to a new location by adjusting a horizontal scroll bar or by clicking the play/stop buttons. The play button causes each of the graphical displays to scroll automatically, allowing the user to rapidly scan large files. The numeric display on the VCR depicts the current offset in the file. The user may bring up specific textual detail by clicking the byteplot visualization. As a future enhancement, we plan to add similar functionality to all graphical visualizations. Similar navigation could be added to the *strings* and other textual displays by allowing the user to click on a textual item and each of the other displays would automatically change to reflect the new offset.

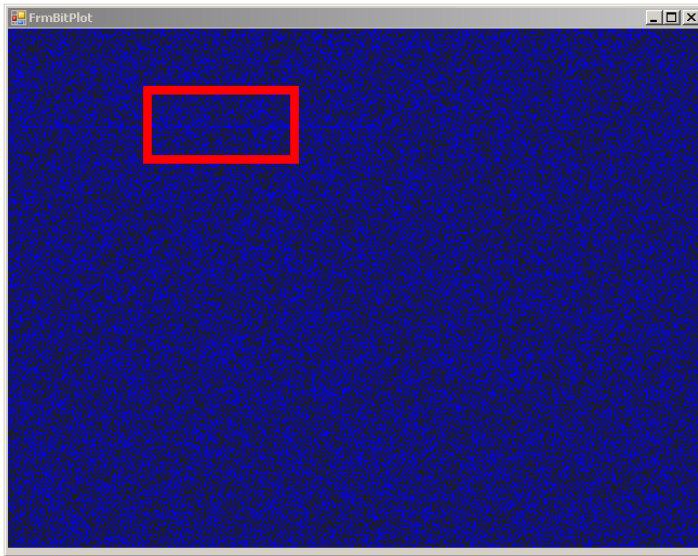
We use color coding to highlight specific attributes of the file under examination. In our system, color coding is accomplished using a small toolbar consisting of four buttons, see Figure 1(h). Our long-term intent is to allow individual analysts to create coloring rules of their own choosing and influence each display, but in our current implementation, we hard coded four, one per button, and they only affect the byte view visualization. These rules include highlighting printable ASCII characters (blue for bytes in the printable ASCII range and gray for all others), displaying byte frequency (blue/low frequency to red/high frequency), inverting the color scheme of the display, and finally a rule for the default color scheme.

## 5 Case Studies

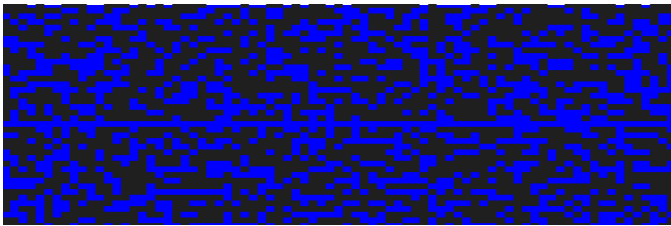
In this section we demonstrate the utility of our approach by using the system in four scenarios of increasing complexity: locating a hidden message contained within an MP3 file, identifying fixed and variable length records contained in database files, reverse engineering of a Microsoft Word document, and analyzing process memory of a Firefox browser running under Windows XP.

### 5.1 Hidden Message in an MP3 File

This example was inspired by Johnny Long's "Death of 1,000 Cuts" talk at the Defcon 14 hacker conference. Long demonstrated numerous ways to hide information from forensic investigators by creatively placing digital information in obscure locations. He showed that it is possible to hide a textual message inside an MP3 audio file by manually altering the file with a hex editor. The file could then be stored on an MP3 player. Short messages, on the order of several hundred bytes or less, cause little to no discernable distortion in the audio playback. Using this technique, we inserted a 331 byte message composed of a sequence of ASCII values in a 3.2MB, 3.5 minute



(a) Full screen display of file



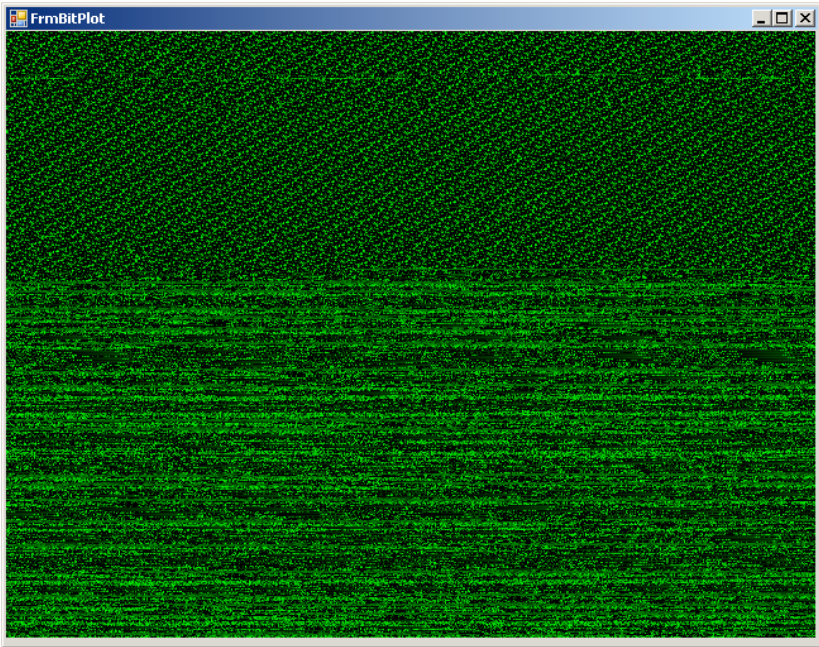
(b) Detail of message region

**Fig. 2.** Byte view visualization of an MP3 containing an ASCII message (a), the detail image (b) more clearly illustrates the message as a horizontal line

song, see Figure 2. Because we were searching for ASCII characters, we turned on the ASCII encoding filter to help highlight sequences of characters. As you examine the figure, note that the remainder of the MP3 file format appears as visual noise, due to the format's compression algorithm, which allows the regularity of the embedded message to become noticeable. By pointing to the suspected message and clicking, the analyst can learn the offset and view the message in the text view window.

While this is a straight forward example, it does illustrate a key aspect of the byte view visualization technique - internal structure is readily apparent. In this case, the deviations from apparent randomness due to compression are easily discerned. It is important to note that the ASCII encoding filter was not required to detect the region, but we chose to use the filter in this example to demonstrate one possible use case. Other means of encoding alphanumeric characters are also discernible using this visualization. For example, alphanumeric characters from the Basic Latin Unicode Set

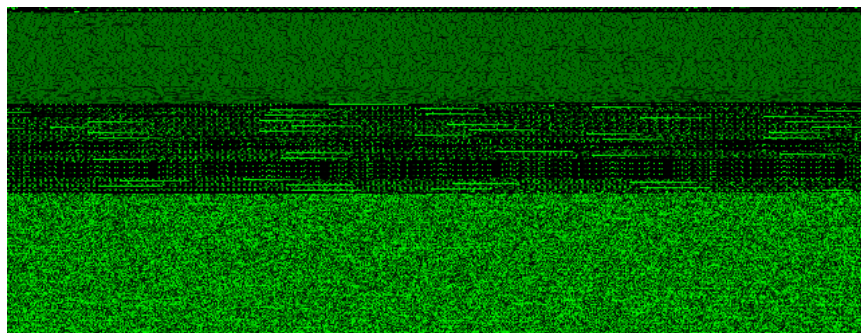




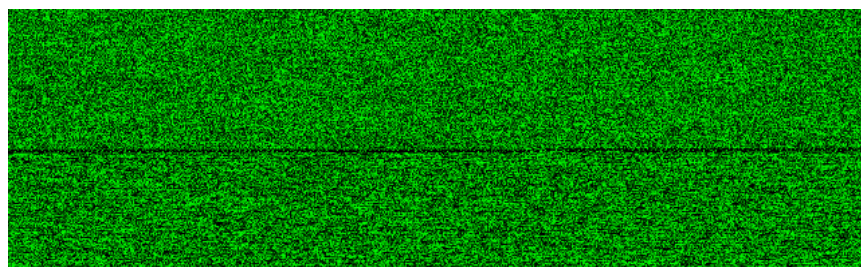
**Fig. 4.** Byte view of a PCAP file from the Defcon Capture the Flag competition. Notice the regularity of the fixed length record structure in the top half of the image and the variable length records in the bottom half.

the file in its entirety. However, this same size document would require approximately 1,024 pages when displayed in a textual hex editor-style format. In addition, the scroll bar on the VCR-like display greatly increased analysis speed. After initially loading the file and opening the byteview window, we used the scroll bar to scan the entire file, a process taking less than a minute. It quickly became apparent that the file contained a header region Figure 5(a), a large compressed or encrypted region, Figure 5(b), and a footer region, Figure 5(c). We used a combination of other visualization displays to provide deeper insight and confirm these initial assumptions. For example, by clicking on major structures in the header region and viewing the results in the text visualization, we confirmed the document's text was located in the top third of figure 5(a). Embedded images constituted the vast majority of the document and appeared as white noise. Each image was preceded by a short header, which was visible in the byteview visualization as a horizontal bar, see Figure 5(b). Closer examination of these image headers using the text view revealed that they were compressed PNG images. The footer contained a mixture of elements including a listing of all hyperlinks contained in the document stored as Unicode. Recall that basic Latin Unicode appears as vertical bars in the byteview visualization.

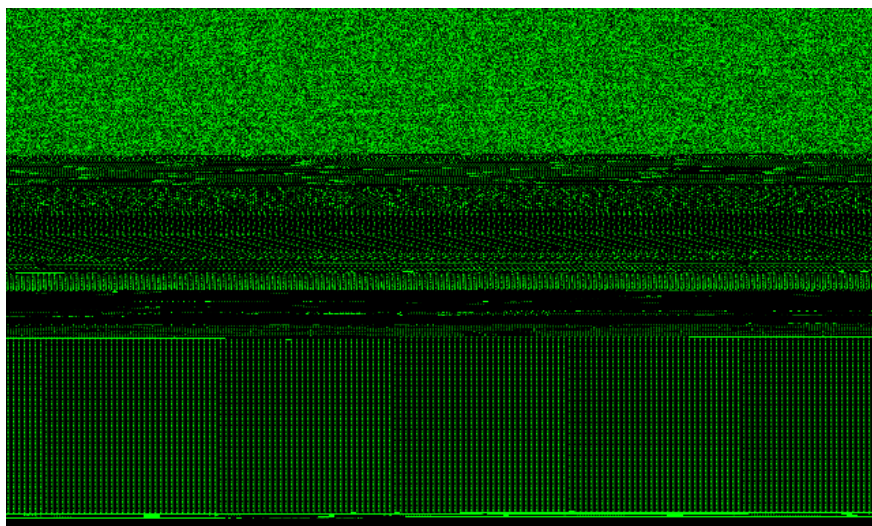
We believe our visual analysis approach bears great promise for analyzing documents stored in binary files. ASCII data, Latin Unicode, internal record structures, and compressed images are all readily apparent. Potential future applications include using visualization to help guide *fuzzing*, the stress testing of application parsers, by



(a) Header Region



(b) Embedded Image Region



(c) Footer Region

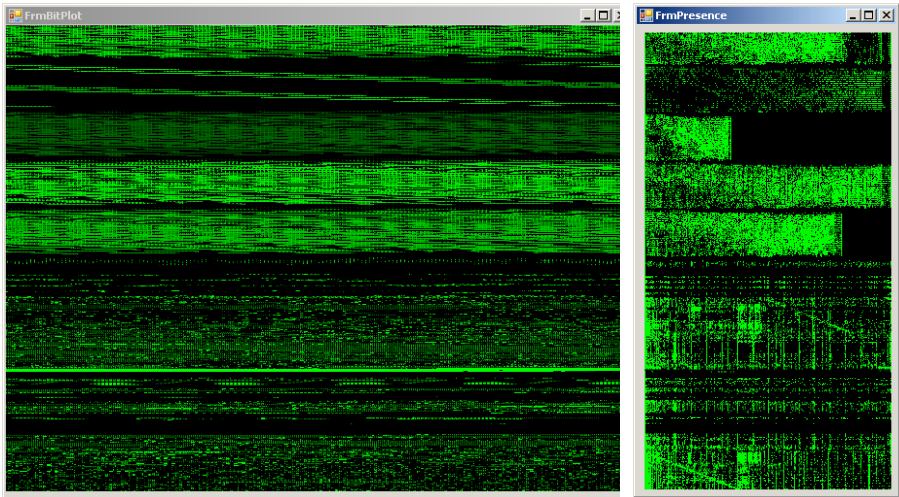
**Fig. 5.** Microsoft Word Binary. The byteview visualization allows the analyst to quickly discover the existence of three major regions in the file. A header region, which contains the text of the document, followed by a large region containing compressed images, and a footer region which includes hyperlinks stored as Unicode.

facilitating identification of internal structures. A common best practice in the fuzzing community is the study of complex file formats as the probability of discovering a vulnerability increases with complexity [13].

#### 5.4 Firefox Core Dump

This final example is a core dump created by a Firefox browser during a crash and differs significantly from the preceding examples, as it is a snapshot of process memory and not a static file format. As such, additional structures not seen during static analysis become visible. For example, Figure 6, shows an image stored by the browser in its process memory, note the gradients (left) and the corresponding byte utilization in the byte presence view (right).

Additional analysis indicated that our visualization approach is useful for related, and potentially very large chunks of binary data, including page files, hibernation files and process memory. It is important to note however, that sharing byteplot images in these cases is a security concern, because it is possible to convert the image back to the raw byte values without loss.



**Fig. 6.** Byteplot view of process memory dumped by Microsoft Windows after a Firefox browser crash. The left figure depicts an image stored in process memory (note the gradients) and the right figure shows the corresponding byte values.

## 6 Conclusions and Future Work

The future of visual analysis of binary data is promising, particularly when such visualization systems incorporate best practices from hex editors, a well-studied field for over 30 years. Our work demonstrated that it is possible to extend the current functionality of the hex editor metaphor by overcoming its significant constraint of a tiny textual window and helping fill the distinct gap between the hex editor and special case binary analysis tools such as disassemblers. Our intent was not to suggest



rejecting the hex editor, but instead buttress its weaknesses and complement its strengths via visualization and improved interaction. A key question we sought to answer was, “Is it possible to do better than the canonical hex/ASCII view provided by today’s hex editors?” The answer is yes. Carefully crafted visualizations provide big picture context and facilitate rapid analysis of both medium (on the order of hundreds of kilobytes) and large (on the order of tens of megabytes and larger) binary files. The traditional hex editor is an inadequate tool for dealing with files of these sizes. However, the traditional hex editor view provides a useful means of providing precise detail. It is possible to create a visualization-enhanced analysis system that combines the functionality of the best hex editors with the strengths of visualization. Key to this approach is the continued exploration of interaction techniques to seamlessly blend visual displays with hex editor interaction best practices. To be most successful, such a system should be based on an extensible plug-in architecture that allows intermediate and advanced end-users to easily create and share both visualization techniques and search/filtering/coloring rules, tapping the combined insight of the user-community.

## References

1. Conti, G., Grizzard, J., Ahamad, M., Owen, H.: Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries. In: IEEE Symposium on Information Visualization’s Workshop on Visualization for Computer Security (VizSEC) (October 2005)
2. Helfman, J.: Dotplot Patterns: A Literal Look at Pattern Languages. TAPOS Journal 2(1), 31–41 (1995)
3. Kaminsky, D.: Black Ops 2006. Blackhat USA (2006) (last accessed December 20, 2007), [http://www.doxpara.com/slides/dmk\\_blackops2006.ppt](http://www.doxpara.com/slides/dmk_blackops2006.ppt)
4. Yoo, I.: Visualizing Windows Executable Viruses Using Self-Organizing Maps. VizSec/DMSec (2004)
5. Carrera, E., Erdelyi, G.: Digital Genome Mapping – Advanced Binary Malware Analysis. In: Virus Bulletin Conference (2004)
6. Flake., H.: Structural Comparison of Executable Objects. Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), pp. 161–173 (2004)
7. A different look at Bagle. F-Secure Weblog (23 September 2005) (last accessed December 20, 2007), <http://www.f-secure.com/weblog/archives/00000662.html>
8. Graphing malware. F-Secure Weblog (25 October 2005) (last accessed December 20, 2007), <http://www.f-secure.com/weblog/archives/00000324.html>
9. Dullien, T., Rolles, R.: Graph-based comparison of Executable Objects. In: Symposium Sur La Securite Des Technologies De L’Information Et Des Communications (SSTIC) (2005)
10. Flake, H.: Diff, Navigate, Audit – Three applications of graphs and graphing for security, Blackhat USA (2004) (last accessed December 20, 2007), <http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-flake.pdf>
11. Nolan, B., Sinda, M.: File Visualization Environment (FiVe). In: National Conference on Undergraduate Research (2008)
12. Kaminsky, D.: Black Ops 2006 : Viz Edition. Chaos Computer Congress (2006) (last accessed May 1, 2008), [http://www.doxpara.com/slides/dmk\\_blackops2006\\_ccc.ppt](http://www.doxpara.com/slides/dmk_blackops2006_ccc.ppt)
13. Sutton, M., Greene, A., Amini, P.: Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley, Reading (2007)

# Effective Visualization of File System Access-Control

Alexander Heitzmann<sup>1</sup>, Bernardo Palazzi<sup>1,2,3</sup>, Charalampos Papamanthou<sup>1</sup>,  
and Roberto Tamassia<sup>1</sup>

<sup>1</sup> Brown University, Department of Computer Science, Providence, RI, USA

<sup>2</sup> Roma TRE University, Rome, Italy

<sup>3</sup> ISCOM Italian Ministry of Communications, Rome, Italy


{aheitzma,bernardo,cpap,rt}@cs.brown.edu

**Abstract.** In this paper, we present a visual representation of access control permissions in a standard hierarchical file system. Our visualization of file permissions leverages treemaps, a popular graphical representation of hierarchical data. In particular, we present a visualization of access control for the NTFS file system that can help a non-expert user understand and manipulate file system permissions in a simple and effective way. While our examples are based on NTFS, our approach can be used for many other hierarchical file systems as well.

## 1 Introduction

The access control model employed by current-generation file systems, such as Microsoft Windows XP and Vista, is rather complex and often insufficiently documented. In a large file system with multiple users, it is rather tricky to understand which users/groups can access which files and with which permissions. Also, the effect of simple operations (such as copy and move) on the permissions of a file are difficult to anticipate and sometimes unintuitive. For example, consider a Windows user who changes the permissions of a certain file to make it not readable by others and later moves this file to another folder where the read permission is inherited from the parent folder. The user is unlikely to realize that after the move, the file is no longer protected. This is due to the fact that in a Windows NTFS file system, there are three types of permissions associated with a file: the *local* permissions for the file, the *inherited* permissions derived from the permissions of the parent folder, and the *effective* permissions, obtained as the union of the local permissions and the inherited permissions.

Inherited permissions have many advantages and have been adopted by several file systems.

Also, inherited permissions and other features of access control mechanisms can make answering questions such as “What group has access to which files during what time duration?” or “If I implement this policy, what conflicts this result?” very difficult .

Understanding file permissions and setting them to achieve desired file sharing and protection goals can be a daunting task for non-expert users and is

non-trivial even for experts. A tool that helps users to understand how access-control permissions are determined and the effect of file system operations on file permissions would be extremely useful for both regular users and administrators.

We believe that an effective way to overcome difficulties of understanding file permissions is through visualization. Therefore, in this paper, we present our preliminary design of a visualization tool that displays access-control information in a way that is easily understandable and helps the user set the correct permissions to achieve file sharing and protection goals. Our visualization tool uses treemaps [6], a popular graphical representation of hierarchical structures based on a recursive decomposition of rectangles into sub-rectangles.

In Windows, advanced file system permissions are displayed as a list. Reeder et al. [9] propose using a square matrix to visualize the permissions of a file system and presents an example with changes of groups and users permissions. Montemayor et al. [8] present a solution for access control visualization based on representing the connections between groups, users, and resources, with a graph. The complexity of access control safety and the administrator's difficulty in dealing with it (which makes visualization of access control very important) is analyzed in [5]. The usability of access control systems is discussed in [3]. Some visualization solutions for access-control and file-sharing policies are presented in [10].

Treemaps were introduced in 1991 [6] as a method of representing a complex hierarchy in a compact space. Bladh et al. [1] provide a file system visualization based on treemaps in the 3D space. Interactive ways to explore a file system through visualization are presented in [4]. Stasko [12] gives an evaluation of different compact ways to represent hierarchical structures. The visualization of dynamic hierarchies is presented in [13]. Finally, in 1971, a method using a nested rectangle representation (that resembles treemaps but though not formally defined) to visualize program execution was presented [7].

## 2 Preliminaries

In this paper, we focus on the *access control list* (ACL) implemented in the Windows NTFS (New Technology File System). NTFS [11] allows to define access control information for each file system object. Using different security policies it is possible to allow or deny access to files and folders for determined users or groups. The file system driver manages all file system requests (i.e., create new files, open existing files, write to files. etc.) as the intermediary between the operating system and the storage device drivers. NTFS ACLs are composed of *access control entries* (ACEs). Each ACE allows or denies specific permissions (i.e., by a user or a group) to or from an object.

Starting with Windows 2000, NTFS allows to dynamically manage permission inheritance. That is, when you create a subfolder or a file in a NTFS folder, the child object not only inherits the parent's permissions but maintains a kind of link with its parent. Furthermore, parent's permissions are stored separately from any local permissions that are directly stored on the child. So for any changes performed on the parent folder, this method allows the child objects

to automatically inherit the changes from their parents and to prevent from overwriting all the local permissions.

This approach allows an administrator or a user to manage a hierarchical tree of permissions that matches the directory tree. Since each child inherits permissions *recursively* from its parent. So it is possible to perform changes of permissions with little effort.

The main downside of *dynamic* inheritance is the increase of complexity and the possibility to have conflicting ACEs. The NTFS security module combines the specified permissions (i.e. local and inherited ACEs, allows or explicit denies) and decides whether to grant or deny the access to a user, a group, or other security entities. Microsoft introduced in Windows XP the *effective permissions* tab to help the administrator in the quite tricky task of understanding the effective permission for a user or a group on a specific file system object.

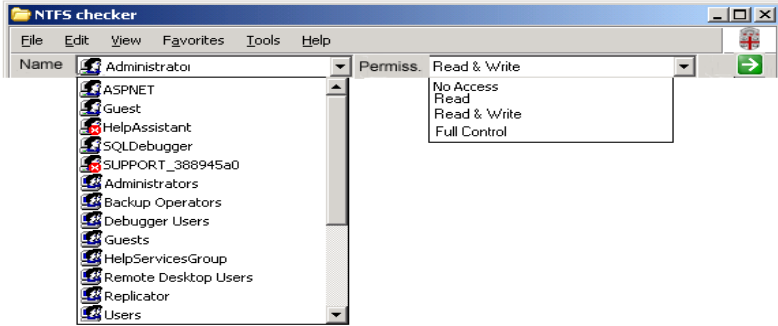
Treemaps (see, e.g., [2, 6]) were introduced in 1991 as a way to represent large hierarchical structures in a compact way. The main idea of creating a treemap can recursively be described as follows: Given a tree  $T$  with root  $r$ , assign a rectangle  $A$  to represent  $T$ . Then, for all the subtrees  $T_1, T_2, \dots, T_k$  of  $r$ , partition  $A$  into  $k$  rectangles  $A_1, A_2, \dots, A_k$  and assign  $A_1, A_2, \dots, A_k$  to  $T_1, T_2, \dots, T_k$ . This process continues until it reaches the leaves, where it assigns distinct rectangles for every leaf of the tree. Given a tree with  $n$  nodes, a treemap can be constructed in  $O(n)$  time using a bottom-up traversal.

Several algorithms have been proposed for assigning rectangles to subtrees. The standard method is based on the “slice-and-dice” algorithm, originally introduced in [6], which uses parallel lines to divide the rectangle assigned to a subtree  $T$  into smaller areas that correspond to the subtrees of  $T$ . It also alternates the direction of the parallel lines (horizontal/vertical) from one level the next, so that the change of levels is displayed. The standard treemap method often gives thin, elongated rectangles. A new method—the “squarified” algorithm—is presented in [2] to generate layouts in which the rectangles approximate squares.

### 3 Effective Access Control Visualization

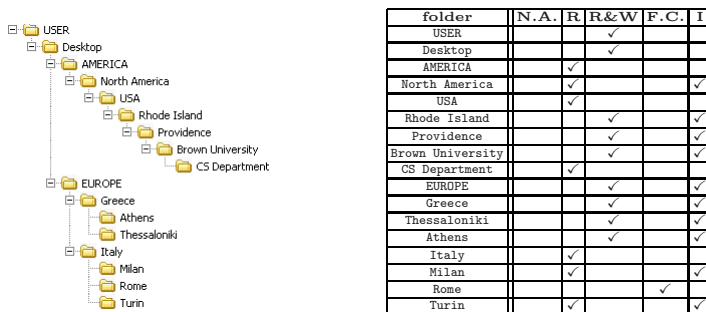
In this section, we present the main features of the tool we have designed to assist administrators and users in better understanding and managing access-control of a hierarchical file system. The tool employs treemaps to visualize the file system tree. We use colors to distinguish the permissions of files and folders, and we indicate where a *break* of inheritance occurs with a special border around the relevant node in the treemap. The input to our tool consists of two items:

1. The “user” input, which indicates the user or group whose permissions we are interested in. In Figure 1, this is indicated with the label “name”.
2. The “baseline” input, which basically indicates a certain combination of permissions upon which the color scheme of our visualization is based. In Figure 1, this is indicated with the label “permission”.

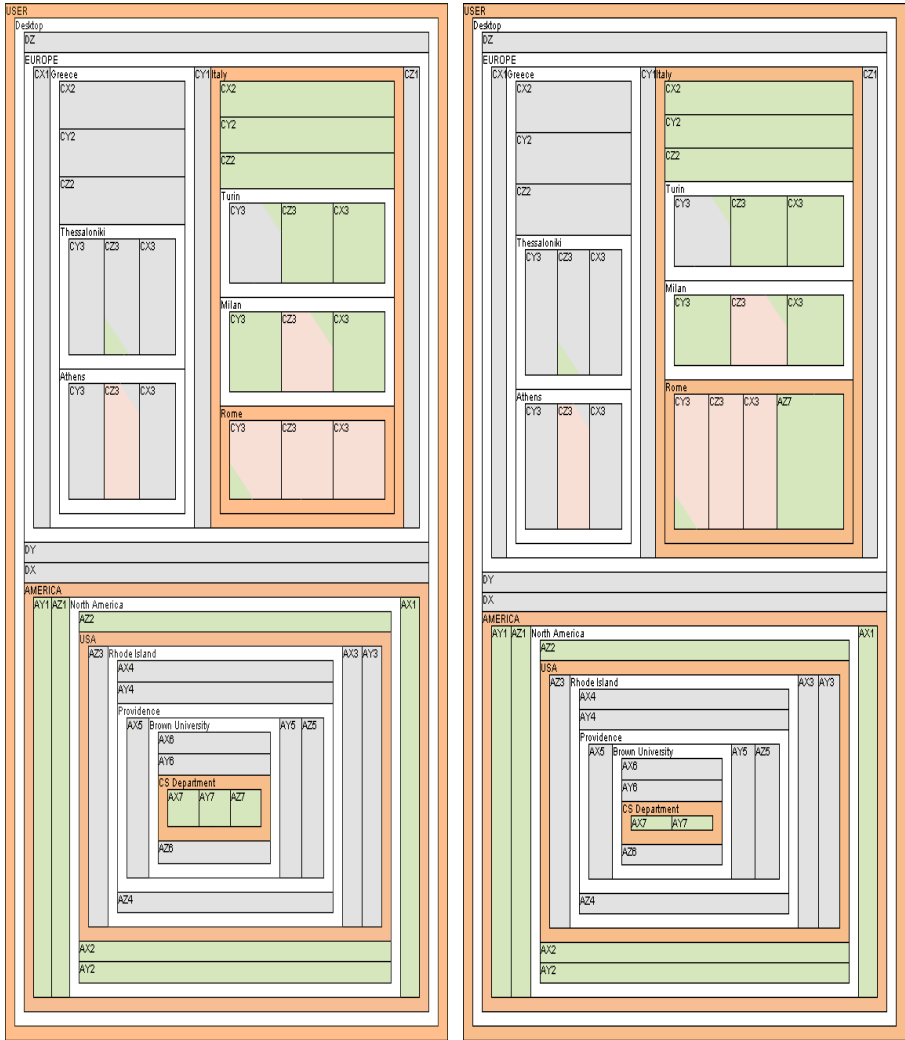


**Fig. 1.** The user interface of our visualization tool. The main screen consists of the “user” input and the “baseline” input.

In the current design, the “baseline” input can take four values, namely the values **no access**, **read**, **read&write**, **full control**. These are sorted in “increasing permission” order. The user can also propose (and insert into the drop-down menu) another combination of permissions (e.g., **read&execute**) and the administrator is responsible for putting the new feature in the correct order (see Figure 1). The visualization tool reads this value and parses the file system tree, building the treemap using the slice-and-dice algorithm. For every file encountered, the associated node in the treemap is painted green, red, or gray, if the file’s permissions are weaker (more restrictive), stronger (less restrictive), or the same as those specified by the baseline, respectively. The tool could potentially use different shades of the same color to declare intensity of permissions. Finally, the tool draws an orange border around treemap nodes associated with files or folders where inheritance is broken.



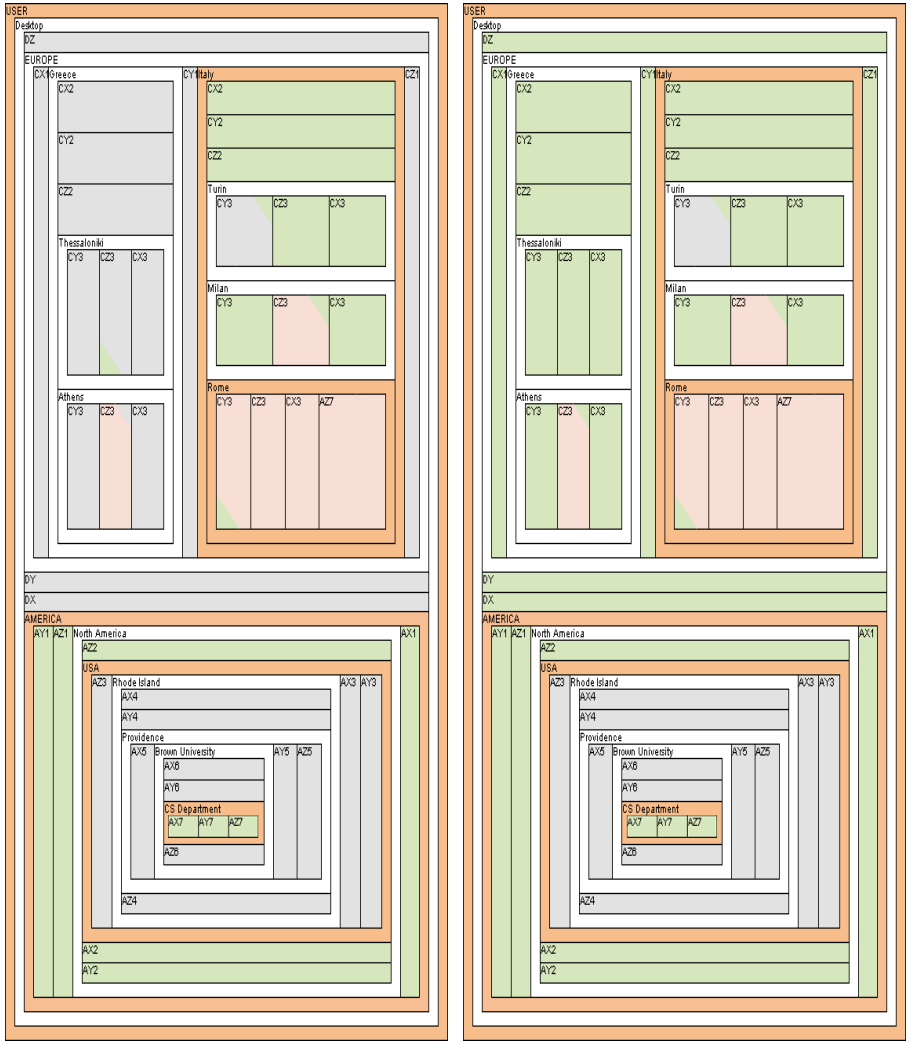
**Fig. 2.** The directory tree that we are going to visualize with our tool, as visualized by Windows explorer. Beside the tree, we also show the effective permissions of each folder of the tree. **N.A.** stands for “no access“, **R** stands for ”read“, **R&W** stands for “read and write“, **F.C.** stands for ”full control” and **I** stands for “inheritance“.



(a)

(b)

**Fig. 3.** Treemap (a) shows the access control permissions for user *administrator*. User *administrator* has permission **R** on file **AZ7** because the color is light green, indicating a weaker permission compared with the baseline **R&W**. One can see that these permissions were inherited by *CS Department* because it is the first orange parent folder, indicating a break of inheritance. Treemap (b) illustrates the result of moving file **AZ7** from the *CS Department* folder to the *Rome* folder. The permissions of this file, indicated by the light green color, are preserved after the move. Furthermore, the size of the rectangle associated with the moved file increases to accentuate the move. Also, the color of the top right corner (the inherited permissions) of file **AZ7** is light green because after the move there is no inheritance from the parent.



(a)

(b)

**Fig. 4.** Treemap (a) shows the result of copying file **AZ7** from the *CS Department* folder to the *Rome* folder. The permissions of the file change, as indicated by the light red color, because of the inheritance from the destination folder. Treemap (b) shows the result of changing the permissions of the *USER* folder from **R&W** to **R**. This change propagates down to descendant files and folders until there is a break of inheritance. Note that file **CZ3** in the *Thessaloniki* folder changes its color from *grey* to *light green* because the local permission (left bottom corner) has a level that is lower than that of the inherited permission (top right corner). It is possible to see the opposite behavior in file **CZ3** in the *Athens* folder, where the inherited permission changes from *grey* to *light green* but the color of the rectangle remains *light red* because the level of the local permission is greater than that of the inherited permission.

We believe that this scheme makes it is easy to gain a general sense of current permissions of the file system as far a certain user is concerned. Furthermore, a more detailed understanding can be achieved simply by exploring the treemap more thoroughly. For example suppose that a file is moved from a folder that has weaker permissions than the baseline to a folder that has stronger permissions than the baseline. The administrator, by using our tool, will be able to notice that difference (since a small green area will appear in a greater red area). There is no longer a need to manually (by exploring the directory with `cd` commands) find files with changed permissions, a task that quickly becomes arduous as more users and other commands such as `copy` or `cacls` are taken into consideration.

We show examples of using our tool to visualize the permissions of the directory tree of Figure 2. Figure 2 shows a directory tree and the effective permissions of every folder contained in this tree. We show in the table of Figure 2 four kinds of permissions, namely the permissions **no access**, **read**, **read&write**, **full control**. Also in the table of Figure 2 we have a column that indicates whether the certain folder inherits the permissions or not (the last column).

In Figure 3(a) we see the representation of our file system with the treemap colored with colors according to permissions, as defined before. In Figure 3(b) we see the treemap layout of the file system after moving a file into a directory that has different permissions from the file. Also, in Figure 4(a) we see the treemap layout of a copy operation and in Figure 4(b) we see the treemap layout of the file system where the permissions of the root node of the directory have changed. Also note that in the presented visualizations we distinguish between the *local* and *effective* permissions. Namely, if the local and effective permissions coincide the tiles are painted with only one color. When this is not the case, we use the upper right corner to indicate the inherited permissions and the bottom left corner to indicate the local permissions. In this way we have a good overview of the permissions that correspond to a file. Note that the figures do not present the break of inheritance of a file since this will clutter up the space. The frames have been produced with the software from University of Maryland (<http://treemap.sourceforge.net/>), where we use the slice-and-dice algorithm for the layout and the increased border option to better display the directory structure.

## 4 Conclusions and Future Work

In this paper, we have presented an effective method to visualize file system access control. We have outlined the design of a tool that visualizes both effective and local permissions and inheritance interruption for the Windows NTFS file system. Work already in progress is the implementation of a full prototype of our system and to perform user studies to evaluate our approach. As future work, we plan to develop further variations of the treemap layout to display additional file permission information. Also we plan to investigate the application of our technique to visualize the *tree-walking* protocol result used in the *RFID singulation* problem.



## Acknowledgments

This work was supported in part by the U.S. National Science Foundation under grants IIS-0713403 and OCI-0724806, by the Kanellakis Fellowship at Brown University, and by the Italian Ministry of Research, grant number RBIP06BZW8, project FIRB “Advanced tracking system in intermodal freight transportation”.

## References

- [1] Bladh, T., Carr, D.A., Schol, J.: Extending tree-maps to three dimensions: A comparative study. In: Masoodian, M., Jones, S., Rogers, B. (eds.) APCHI 2004. LNCS, vol. 3101, pp. 50–59. Springer, Heidelberg (2004)
- [2] Bruls, M., Huizing, K., van Wijk, J.: Squarified treemaps. In: Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG), pp. 33–42 (2000)
- [3] Cao, X., Iverson, L.: Intentional access management: making access control usable for end-users. In: Proc. of Int. Symposium on Usable Privacy and Security (SOUPS), pp. 20–31 (2006)
- [4] Foster, J., Subramanian, K., Herring, R., Ahn, G.: Interactive exploration of the AFS file system. In: Proc. of the IEEE Symposium on Information Visualization (INFOVIS), p. 215 (2004)
- [5] Jaeger, T., Tidswell, J.E.: Practical safety in flexible access control models. *ACM Trans. Information Systems Security* 4(2), 158–190 (2001)
- [6] Johnson, B., Shneiderman, B.: Tree maps: A space-filling approach to the visualization of hierarchical information structures. In: Proc. IEEE Visualization, pp. 284–291 (1991)
- [7] Johnston, J.B.: The contour model of block structured processes. *SIGPLAN Not.* 6(2), 55–82 (1971)
- [8] Montemayor, J., Freeman, A., Gersh, J., Llanos, T., Patrone, D.: Information visualization for rule-based resource access control. In: Proc. of Int. Symposium on Usable Privacy and Security (SOUPS) (2006)
- [9] Reeder, R., Bauer, L., Cranor, L., Reiter, M., Bacon, K., How, K., Strong, H.: Expandable grids for visualizing and authoring computer security policies. In: Proc. ACM Conf. on Human Factors in Computing Systems (CHI), pp. 1473–1482 (2008)
- [10] Rode, J., Johansson, C., DiGioia, P., Filho, R.S.S., Nies, K., Nguyen, D.H., Ren, J., Dourish, P., Redmiles, D.F.: Seeing further: extending visualization as a basis for usable security. In: SOUPS, pp. 145–155 (2006)
- [11] Russinovich, M.E., Solomon, D.A.: *Microsoft Windows Internals*, 4th edn. Microsoft Windows Server <sup>TM</sup>2003, Windows XP, and Windows 2000 (Pro-Developer). Microsoft Press, Redmond (2004)
- [12] Stasko, J.: An evaluation of space-filling information visualizations for depicting hierarchical structures. *Int. J. Hum.-Comput. Stud.* 53(5), 663–694 (2000)
- [13] Wilson, R.M., Bergeron, R.D.: Dynamic hierarchy specification and visualization. In: Proc. of the IEEE Symposium on Information Visualization (INFOVIS), p. 65 (1999)

# Visual Analysis of Program Flow Data with Data Propagation

Ying Xia, Kevin Fairbanks, and Henry Owen

Georgia Institute of Technology  
{yxia, Kevin.Fairbanks, owen}@gatech.edu

**Abstract.** Host based program monitoring tools are an essential part of maintaining proper system integrity due to growing malicious network activity. As systems become more complicated, the quantity of data collected by these tools often grows beyond the ability of analysts to easily comprehend in a short amount of time. In this paper, we present a method for visual exploration of a system program flow over time to aid in the detection and identification of significant events. This allows automatic accentuation of programs with irregular file access and child process propagation, which results in more efficient forensic analysis and system recovery times.

## 1 Introduction

Although many methods of system recovery after a compromise have been documented [1, 2, 3], this process is often complex and time consuming. This paper aims to illustrate that visualization of captured program flow data can better enable a forensics expert in their investigation for purposes of intrusion detection and recovery. This is achieved through the automation of accentuation and propagation of data gathered through detailed system call monitoring [4]. The resulting information is then rendered with the Prefuse visualization toolkit [5].

The motivation of our research is to address the following points.

- Visualization of program flow data gives the user a better overall view of system behavior.
- Irregular events are more intuitively identifiable when presented visually.
- Automatic accentuation of events lowers data analysis time and draws attention to trouble spots.
- Taint propagation of process flows allows for rapid estimation of damage suffered following an intrusion.

To achieve these goals, we first present our system call data as a flow diagram based on time. As additional system processes spawn and files are accessed, each event is displayed at the time interval in which it occurred. Over time, as repeated program executions and file accesses occur, a standard pattern for proper behavior emerges for the system. During periods of irregular activity, new flow patterns may emerge and are brought to the attention of the user. Finally, should any process or file become tainted, we then propagate that taint through the flow diagram based on program and file access to show the extent of potential damage occurred on the system.

The remainder of the paper is organized as follows. Section two provides the necessary background information about intrusion detection, visualization and related work. Section three presents our approach for visualization and automation. The results of our research are presented in section four, and we conclude in section five.

## 2 Background and Related Work

There are several methods of obtaining the required program execution and file access information such as through a combination of tools such as `pstree` and `lsuf`. However, these tools lack several key features, such as the lack of a history of prior processes and files opened, and the lack of a timestamp and duration of use. Even repeated calls to these tools will not guarantee the completeness of data. Thus, to obtain the requisite program information detailing executions, file accesses, and forked processes, we used the same method as discussed in [4]. By capturing necessary information from chosen system calls and exporting them via the kernel message buffer to a database, we minimize our impact on program execution. It was our intent that the data we wish to obtain and visualize remains as close to actual program execution as possible with minimal differences when our monitoring technique is in place. This technique makes it difficult for an intruder to disable until they have obtained sufficiently elevated privileges on the system. At that point, the points of intrusion and steps taken to reach that point have already been logged. Finally, directly modifying the system calls rather than replacing them with a kernel module avoids the possible detection methods discussed in [8].

Prefuse [5] is a software framework written in Java to aid developers in the creation of information visualization applications. The goal of Prefuse is to greatly simplify the process of data handling and representation. The reason we chose Prefuse over other visualization toolkits such as Piccolo, Dot, and the Visualization Toolkit is its flexibility, the ability to render large amounts of data in an efficient manner, and excellent documentation.

There are many similarities between the mapping of a program flow of an operating system and that of a process, such as the work presented in [6] and [7]. In [6], Balzer et al used a landscape metaphor to visualize structures of large software systems. They used a defined hierarchal layout, which can be arbitrarily deep, to show dependence and included packages. In [7], Bohnet and Dollner applied an effective 2½ D visualization to facilitate path discovery in the function call graph to identify feature-implementing functions. Both of these techniques share several similarities, such as using data clustering to show dependence and hierarchy.

## 3 Approach

Our approach to program flow visualization can be broken down into three stages: data acquisition and storage, analysis, and visualization. It is not necessary that these operations be performed in the order listed. In fact, further data analysis may be performed based on user input from examining the resulting visualization. Finally, only the data acquisition and storage aspects must be performed in real time on the target system to be monitored. The analysis and visualization aspects can be performed elsewhere. Figure 1 presents our system architecture.

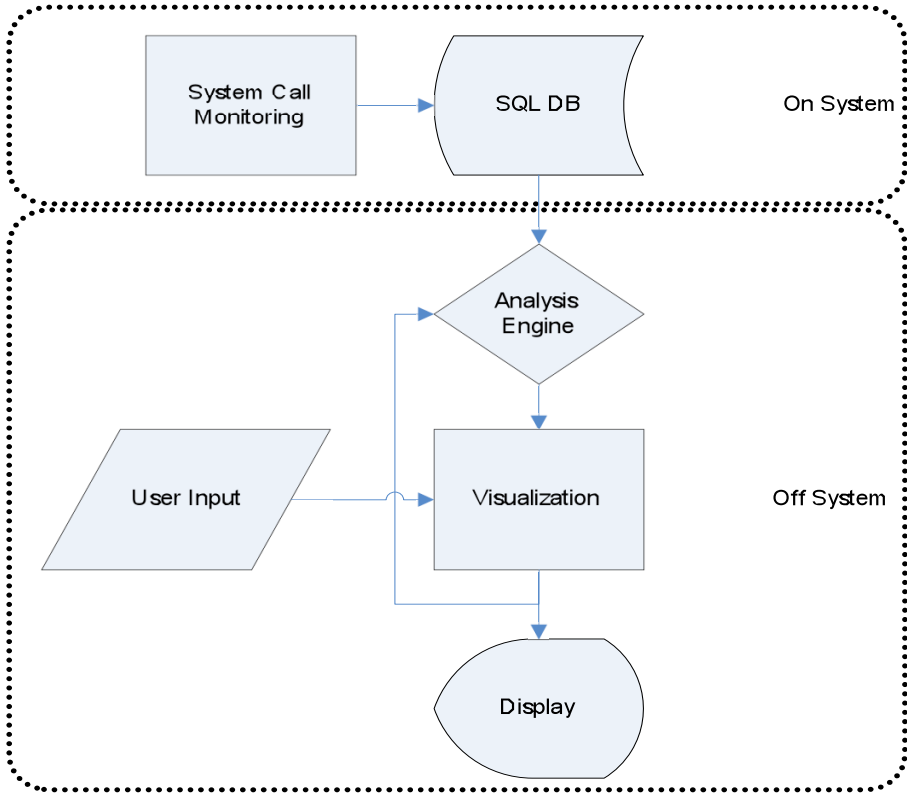


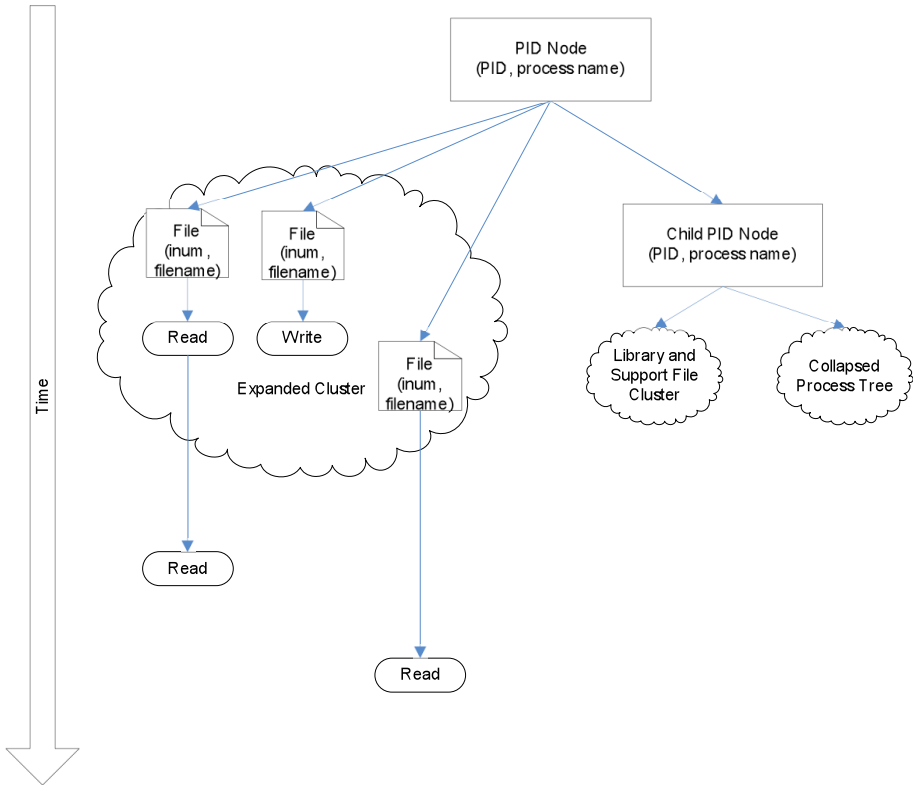
Fig. 1. System Architecture

### 3.1 Data Acquisition and Storage

As briefly discussed in section two, our data is obtained by monitoring specific system calls and exporting that information from kernel space to user space using the kernel message buffer. This is achieved by inserting additional code into our targeted system calls. In our research, we chose to target the system calls dealing with file access and program execution. Thus, `sys_execve`, `sys_fork`, `sys_open`, `sys_read`, `sys_write`, and `sys_close` were modified. In particular, the data we are targeting is the following: process identification number(PID), child process PIDs, executable name, file names, file inode numbers, file descriptor numbers, reads and writes, and time stamp information for all these actions. The acquired data is then stored in a SQL database for further processing. This phase of the architecture is performed in real-time on the target system.

### 3.2 Data Analysis

In order to present our data visually, we must first process the raw data obtained from system call monitoring. This is achieved through the creation of several databases



**Fig. 2.** Visualization Concept

containing subsets of the logged information. The first such database contains the program-file relationships. This database contains the file usage of programs during execution, as seen by `sys_open`. Our goal for creating this database is to identify the key library files used by the program for execution. The second database we create using the system call data is a time ordered database which shows the order in which a program accesses its library files. Generally, each program accesses files in a predefined manner at predictable intervals between each access. Using the program-file relationship database and the time ordered database, we are thus able to obtain a general idea of proper program execution behavior. A third database, containing the list of sensitive system files (such as `/etc/passwd`), is also generated for modification detection and accentuation in the visualization phase. Please see [4] for further details about the data acquisition and analysis stages.

### 3.3 Visualization

#### 3.3.1 Rendering

As previously discussed in section one, one of the goals of visually representing the program flow is to provide a better overall view of the entire system. To achieve this goal, we chose to visually represent our data in the following way:

- Each PID will be a node in the flow diagram; child PIDs and accessed files will be children of that node.
- The flow diagram will be time ordered using the available time stamp of each logged operation. Due to the fine-grained time unit used by Linux ( $1^{-10}$  seconds per step), we chose to categorize every event that occurs in a certain time period as concurrent. The way this is chosen is user defined. For example, the user may chose that all events that occurred within the same second be treated as concurrent, or perhaps all events that occurred within the same millisecond. The length and scope of the tree will be determined by this user input.
- The diagram will be collapsible at each node and the tree will be in collapsed mode by default. As the user clicks on each node, it will render itself into a sub-tree consisting of child processes and accessed files.
- The files identified as library files under the program-file relationship database will be displayed as a single collapsed cluster.
- A search option will also be provided for the user to quickly locate a particular PID or file within the tree.

Figure 2 shows a generalized view of our visualization. For actual results, see section 4.

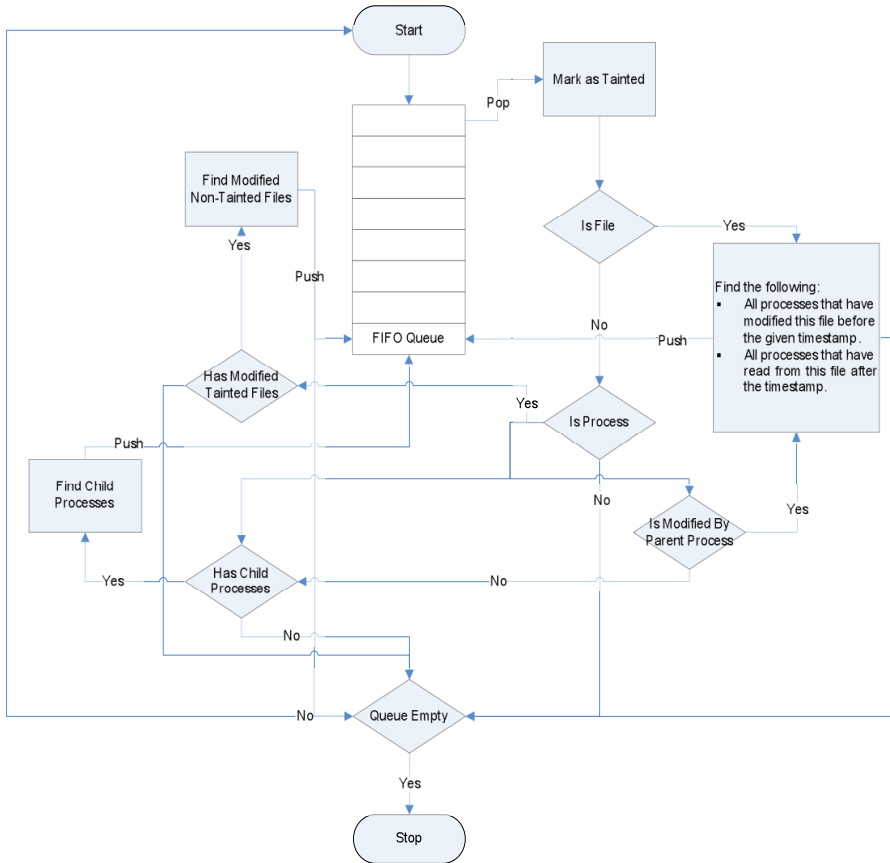
### 3.3.2 Accentuation

For our research, we chose to target program-file access behavior for automatic accentuation. Using the program-file relationship database, we can identify the necessary library and support files required by each program for execution. These files are then displayed as a cluster on our overall visualization. Should any particular program access a different library file or one that is not in the order specified as by the time ordered database, then this program will be expanded and brought to the attention of the user. Also, any program that modifies a declared sensitive file will also be noted for further investigation.

### 3.3.3 Propagation

Our goal for propagation is to judge the extent of influence that any particular file or process has on the entire system. This is also a way for the user to quickly estimate the extent of damage caused by this intrusion should this file or program be later determined as malicious. Once a specific PID or file has been marked on the tree as being tainted, propagation takes places using the algorithm in figure 3.

This algorithm begins by popping the top item for processing. If the object is a file, then any process that have modified the file as well as any process that have read from the file will be pushed onto the queue. If the object is a process, three checks are made. First, if the process has modified any tainted files, then any file modified by the program that is not marked as tainted is added to the queue. Second, any parent process that has modified the current process will be added to the queue. Third, any child processes created by this process is added to the queue. This algorithm terminates under two conditions, if the queue is empty, and if the distance to origin value has been exceeded.



**Fig. 3.** Propagation Algorithm

To terminate this propagation, we use a value called distance to origin. This value is specified by the user and is defined as the number of links between the current object to the original marked source. For example, the process that directly modified the marked tainted file is considered to have a distance of 1. The parent process or the files modified by that marked process have a distance value of 2. Once this distance value grows beyond a limit, then the current examined item is dropped from the queue.

## 4 Results

### 4.1 Visualization Results

For our experiment, we obtained the data from a modified 2.6.20 kernel running Ubuntu Linux. During the monitoring process, we executed several common programs, such as Firefox, gedit, cp, and cat, and several Linux viruses.

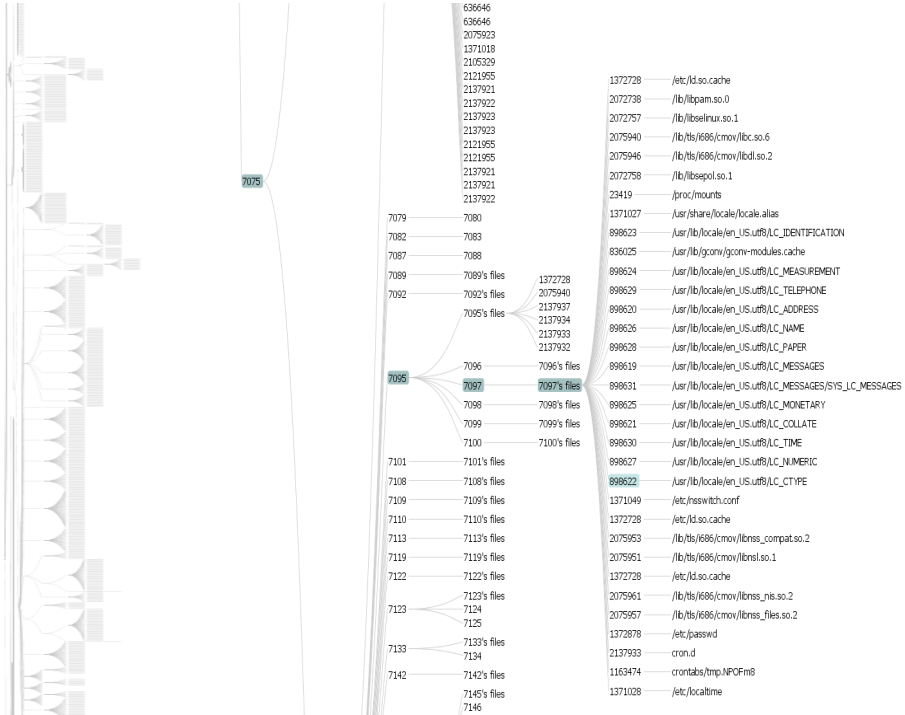


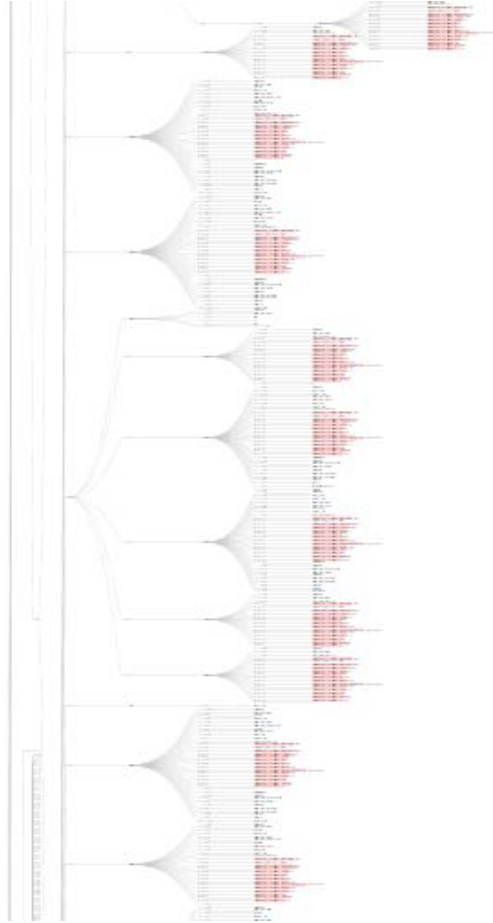
Fig. 4. Process Tree

Figure 4 shows a time slice of our process tree on the left and a view of the same tree zoomed in on a particular file on the right. When an object is selected by clicking on an object name, it will be highlighted as light green. All parent nodes within the tree will be displayed as dark green.

Figure 5 shows our selection of the “/usr/lib/locale/en\_US.utf8/LC” file package (a commonly used set of files) and the subsequent accentuation of all such packages on the process tree. Using file package selection quickly allows users to identify when and where programs perform actions on those files, such as reading or modifying. Highlighting file packages also allows users to identify programs that may not have been executed through normal channels or was renamed in order to disguise the program.

Figure 6 shows the results of marking a file as tainted and propagating the results through the process tree. Everything marked as red is at risk for potential infection, and a system restore must insure that every file marked thusly is examined. Further improvements to the taint propagation algorithm can trim down the number of files to be examined by the taint propagation.





**Fig. 5.** File Package Highlighting

## 4.2 Future Work

This method presented in this paper is still in the implementation phase and as such we are currently not ready to deploy this as an application. There are still several challenges that must be overcome.

First, for the purposes of demonstrating our method of visualizing program flow with data propagation, the algorithm used for taint propagation and automated accentuation are very simplistic and lacks a rigid definition. We hope to develop a better and more concrete algorithm to perform the aforementioned tasks. Second, better methods and optimization approaches must be explored to better streamline the visualization and to improve on the intuitiveness of this display. Third, due to the size of the visualization, it makes it difficult for users to easily identify particular processes



Fig. 6. Taint Propagation

without zooming in on a particular section. In the future, perhaps additional overlays and methods can be added for the user to be able to identify programs without zooming in. Finally, our current method of converting system call data into the visualization is broken up into several stages and requires several conversions between various types of data (text, sql, xml) using a variety of languages (c, python, java). In the future, a direct kernel logging to visualization data tool should be explored to minimize overhead and processing time.

## 5 Conclusions

We have presented our methodology to visually represent and analyze the program flow of a system. Using this visualization, it is possible for the user to quickly detect irregularities in program execution and accentuated trouble spots of illegal file access. Moreover, taint propagation gives the user the ability to quickly gauge the impact of a potentially malicious program or file to aid in the recovery process. We have demonstrated

this method by using it to visualize the program flow of the Ubuntu Linux operating system while performing several tasks. Finally, we have demonstrated how taint propagation tracking through the program flow can provide important information to an investigator by providing a way of quickly estimating program and file impact on a system. We will continue to refine, improve, and draw additional conclusions from our research based on the experiences gained.

## References

- [1] Ring, S., Esler, D., Cole, E.: Self Healing Mechanisms for Kernel System Compromises. In: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems. ACM press, New York (2004)
- [2] Grizzard, J., Owen, H.: On a  $\mu$ -kernel Based System Architecture Enabling Recovery from Rootkits. In: Proceedings of the First IEEE International Workshop on Critical Infrastructure Protection, Darmstadt, Germany (2005)
- [3] Forrest, S., Hofmeyr, S., Somayaji, A., Longstaff, T.: A Sense of Self for Unix Processes. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy (1996)
- [4] Xia, Y., Fairbanks, K., Owen, H.: A Program Behavior Matching Architecture for Probabilistic File System Forensics. In: ACM SIGOPS Operating Systems Review special issue on Computer Forensics (April 2008)
- [5] Prefuse: Information Visualization Toolkit, <http://prefuse.org/doc/faq>
- [6] Balzer, M., Noack, A., Deussen, O., Lewerentz, C.: Software Landscapes: Visualizing the Structure of Large Software Systems. In: Proceedings of the IEEE TCVG Symposium on Visualization, Konstanz, Germany (2004)
- [7] Bohnet, J., Dollner, J.: Visual Exploration of Function Call Graphs for Feature Location in Complex Software Systems. In: Proceedings of the 2006 ACM symposium on Software Visualization, Brighton, United Kingdom (2006)
- [8] Dornseif, M., Holz, T., Klein, C.: NoSEBrEaK, Attacking Honeynets. In: Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC (2004)
- [9] Abdullah, K., Lee, C., Conti, G., Copeland, J., Stasko, J.: IDS Rainstorm: Visualizing IDS Alarms. In: Visualization for Computer Security, VizSec 2005 (2005)
- [10] Takada, T., Koike, H.: Tudumi: Information visualization system for monitoring and auditing computer logs. In: Proceedings of Information Visualization (2002)

# A Term Distribution Visualization Approach to Digital Forensic String Search

Moses Schwartz and L.M. Liebrock

New Mexico Institute of Mining and Technology, Socorro, NM 87801, USA  
moses@nmt.edu, liebrock@cs.nmt.edu

**Abstract.** Digital forensic string search is vital to the forensic discovery process, but there has been little research on improving tools or methods for this task. We propose the use of term distribution visualizations to aid digital forensic string search tasks. Our visualization model enables an analyst to quickly identify relevant sections of a text and provides brushing and drilling-down capabilities to support analysis of large datasets. Initial user study results suggest that the visualizations are useful for information retrieval tasks, but further studies must be performed to obtain statistically significant results and to determine specific utility in digital forensic investigations.

**Keywords:** Term distribution visualizations, digital forensics, text string search.

## 1 Introduction

Digital forensic string search is a vital component of the forensic discovery process [2,3,12]. By searching through strings, an analyst may identify forensic artifacts residing in slack space, in deleted files and unallocated space, or in existing files without considering format details [2,12]. However, the state of the art method for identifying artifacts in these datasets is to use a conventional search tool such as Grep [7] and then rely on a human analyst to read through all of the identified hits [3,6]. This task is different from most other string search tasks in that the dataset is almost completely unstructured and the number of hits is extremely high [3]. There has been very little work on reducing the information retrieval overhead and information overload associated with this task [2,3]. Information visualization is one approach to addressing this problem [3].

We propose the use of term distribution visualizations (discussed in more detail in [16]) to ease the task of digital forensic string search. These visualizations, based on the TileBars method [11], show the frequency of a set of search terms throughout a document. This may act as a primary navigation aid for an analyst, allowing her to quickly identify sections of the dataset that may contain relevant information, and then present the text of identified sections. A Focus+Context mechanism provides support for large datasets by allowing the analyst to brush (or select) a large, potentially relevant section, and then drill

down (or zoom) to a finer granularity version of the visualization. The visualization and Focus+Context model is demonstrated in Fig. 2 in Section 4.

Section 2 of this paper provides an overview of related work. Section 3 presents the term distribution visualizations and the Focus+Context model. We provide an example of the use of the visualizations in a simulated forensic case in Section 4, and describe our user study and initial results in Section 5. Plans for future work and conclusions are presented in Sections 6 and 7.

## 2 Related Work

We have found no work that directly applies visualization to digital forensic string search. There has been some work on using advanced search methods for digital forensics [2,3] and much on visualizations that might be applied to the problem [4,5,9,10,11,14,15,17,18,19,20], but none that explicitly discuss the application of visualizations to digital forensic string search.

In [2], Beebe and Dietrich discuss the need for improved digital forensic text string searching; their focus is clustering algorithms. In [3], Beebe and Clark use a clustering algorithm for digital forensic string search. Visualization is suggested in [3] to improve the digital forensic search process, but is not elaborated on.

Early work on visualizations of term distribution focused primarily on their use as relevance-feedback mechanisms for conventional search engines. TileBars, as presented in [11], compactly indicates relative document length, query term frequency, and query term distribution throughout a document (e.g., see Fig. 1(a)). In [13] and [14], TileBars are included as part of a set of visualizations to be used for improving World Wide Web search results. Relevance Curves are also included, which are similar to the histogram visualizations presented here. In [4], a TileBar-inspired term distribution visualization is placed in a scrollbar as an unintrusive and effective within-document search aid.

There has been considerable work on visualizations for text mining, e.g., [5], [18], and [20]. Text mining to identify relevant queries is an important aspect of information retrieval; however, only [5] has a facility to directly view portions of the text and these projects do not place much emphasis on within-document information retrieval. None of this work considers digital forensics.

Visualizations of term distribution have been used for more general trend analysis, as in [10], [15], [17], and [20]. The interaction paradigm in [17] is very similar to the one presented here. It visualizes arbitrary time-series textual data in a histogram format, based on user-supplied queries. However, its primary applications are for information technology tasks such as auditing system logs, its visualizations are relatively coarse, and it is not intended to be used as a general-purpose information retrieval tool.

The most relevant work explicitly visualizes text for information retrieval, e.g., [8], [9], and [19]. [19] visualizes term distribution in a histogram to support information retrieval from speech archives; [19] is relatively specialized, does not explicitly support very large datasets, and uses a relatively coarse visualization. [9] and an accompanying case study, [8], present ProfileSkim, a tool to

visualize a document and provide a user interface for browsing and information retrieval. ProfileSkim uses a language modeling approach to relevance profiling and visualizes a document as a sequential histogram of relevance scores based on user-supplied queries. ProfileSkim’s visualization does not provide granular information on the distribution of each search term, nor support searching very large datasets. Although much of this related work could be applied to digital forensics, none explore the potential applications.

### 3 Visualization Techniques

TileBars and histograms, in conjunction with a Focus+Context model, comprise a Query-Browse (QB) information retrieval model [12]. Both visualizations support variable-granularity term distributions, which may be calculated using either a sliding window or discrete blocks of text. In this paper, we show only visualizations calculated with discrete blocks, exclude color TileBars [16] and present a new visualization variant (filled-line histograms). The visualizations shown in the following section correspond directly to those used in our user study, as discussed in Section 5. All example images in this section have been generated from a simulated digital forensic scenario, as discussed in Section 4.

#### 3.1 TileBars

As in the original TileBars [11], the TileBar visualizations in this work are matrices of tiles. Along the horizontal axis, each block represents a block of text. The darkness the block indicates the number of occurrences of a search term in the block. Fig. 1(a) shows an example of the results from our TileBar implementation. Term distribution appears to be obvious and intuitive in this visualization. However, with large numbers of terms this visualization may become harder to interpret and less intuitive. Quantifying this effect is a subject for future work.

#### 3.2 Histograms

Histograms [16] are an extension to the original TileBars [11] visualization concept, and very similar to Relevance Curves [14]. Here distributions are plotted on a graph as a sequential histogram. This supports identification of frequency as the height of a peak, as well as overlap by overlap of the distribution graphs.

Fig. 1(b) shows a greyscale histogram. Overlapping areas appear darker, so distribution overlap is very apparent and intuitive, but there is no indicator of which terms are overlapping or where each term occurs.

Fig. 1(c) shows a color histogram, where the lighter color (than the legend) is used to permit color mixing. Where overlaps occur, the colors are mixed based on how many terms are in the block of text. In this case, term-specific information is readily available and distribution overlap is intuitive. However, interpreting color blending as distribution overlap requires additional cognitive effort.



**Fig. 1.** All visualization variants used in the user study. (a) shows a TileBar, and (b) through (e) show histogram variants. All visualizations were generated with a simulated digital forensic case, discussed in Section 4, and the search terms “Boondoggle,” “Digitech,” “Jessie,” “Maggiano,” “Million,” and “Watson.”

Fig. 1(d) and Fig. 1(e) show two variants of color histograms—line histograms, which present the same information but do not use color blending, and filled-line histograms, which represent overlap by dark fill underneath a color line. In line histograms, there is no need to lighten the color for mixing, so outline colors more closely match the legend. In the filled-line histograms, the fill is done in grey and terms’ occurrences are outlined in the legend color. This clarifies to some extent the concentrations and the set of terms in any overlap area.

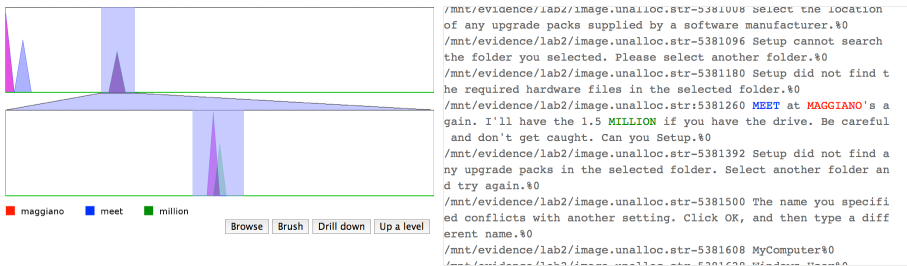
### 3.3 Focus+Context

The Focus+Context model allows a user to brush an area of interest within a TileBar or histogram and drill down to visualize the dataset with finer granularity. The previous visualization remains visible to indicate relative location within the overall dataset (see Fig. 2).

## 4 Digital Forensic Analysis

To illustrate the use of our visualization model for digital forensic string search, we apply it to a digital forensics training module developed by Sandia National Laboratories. In this exercise, we are presented with a seized hard drive image and must perform a digital forensic string search on unallocated and slack space on the drive image to find artifacts. In the fictional scenario, Roberta Hutchins has been accused of attempting to sell trade secrets for Digitech’s Project Boondoggle to an individual named Jessie. Interviews have revealed that Roberta planned to meet Jessie at the restaurant Maggiano’s, at which point she would be given 1.5 million dollars.

We preprocess the dataset by running Grep, with a list of search terms, on the extracted strings. Next, we use our visualization tool. Fig. 2 shows the reduced dataset in the visualization utility with relevant sections brushed and drilled down to a particular artifact supporting the case against Roberta.



**Fig. 2.** The visualization tool applied to a notional forensic case. The left pane shows the dataset visualized with the queries “maggiano,” “meet,” and “million.” The section that appears to contain all terms (as indicated by color blending) is brushed, and the visualization drilled down. The drilled section may then be easily browsed. The right pane shows the text of the selected area.

This simple digital forensics example shows that these visualizations can be effective in focusing attention very quickly to the area of the data set that is most likely of interest. This tool is even more effective in complex data sets where search terms appear in many locations and the user must find where certain terms appear in proximity to others to quickly find relevant evidence.

## 5 Usability Study

We performed a small pilot user study. Here we describe the study, show some preliminary results, and draw what conclusions we can from the preliminary data.



## 5.1 Study Design

A pilot study was conducted on five subjects from New Mexico Institute of Mining and Technology. Subjects were senior undergraduate students, graduate students, or faculty in the Department of Computer Science. All subjects had prior exposure to this research, but none had previously used the interface.

The study used eight electronic documents and was administered through a web interface. Lewis Carroll’s “Through the Looking Glass” was used for training and 8000-line excerpts from the United States Federal Register were used for testing. Unique information was identified in each excerpt by randomly identifying a 400-line section and then selecting specific facts within that section. From these “answers” we created questions and multiple-choice quizzes to determine whether the subject found the correct information.

The study used five visualization variants and two web-based versions of Grep. The visualization variants included: TileBars, greyscale histograms, color histograms, line histograms, and filled-line histograms (see Fig. 11). The Grep variants showed either all occurrences of all search terms as generated by:

```
grep -C2 -aif terms_file target_file
```

or all overlapping sections as generated by:

```
grep -C15 -i term0 target_file | ... | grep -C15 -i termN
```

Subjects filled out a survey before beginning the study, after each trial, and at the end of the study. The first survey gathered basic demographic information. The other surveys elicited qualitative feedback.

Before the actual trials, subjects were given as much time as they wanted to familiarize themselves with the interface. Training used Lewis Carroll’s “Through the Looking Glass” with the Grep interface, TileBars, and color histograms.

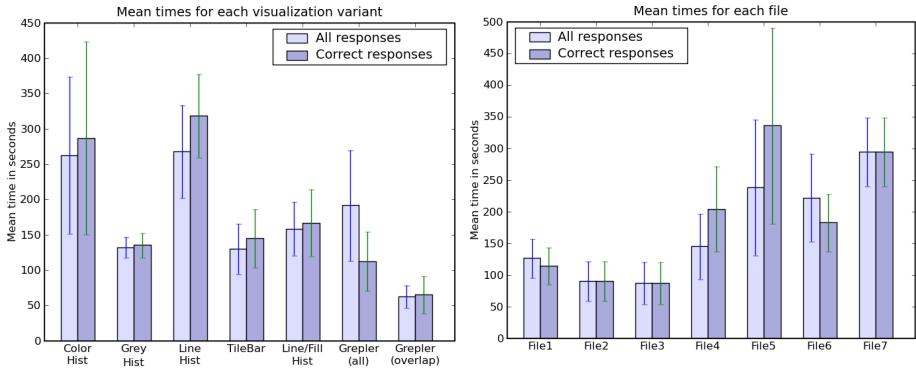
Subjects were presented with each document sequentially, with one of the described tools. The order of the documents was fixed, but the search aids (visualization or Grep) were counterbalanced to minimize carryover effects. In each trial, the time until a subject answered the quiz (which presumably coincides with finding desired information) was measured. Answer correctness was also recorded.

## 5.2 Usability Study Results

Fig. 13 shows the mean time to complete an information retrieval task for each search aid and the time to complete the information retrieval task for each file.

## 5.3 Usability Study Analysis

Since the pilot study results involve a very small sample, we do not dwell on analysis, as any claims would be suspect. However, many of the visualizations appear to perform comparably to Grep with all hits shown—this is a very positive early result. To see why, consider how long these searches would have taken with no search aid at all! Grep that only shows overlapping occurrences of terms seems to have been the most effective search aid, but excluding so much data may render the technique unsuitable for digital forensic purposes.



**Fig. 3.** Mean time to complete an information retrieval task for each search aid and mean time to complete the task for each file. Error bars show standard error.

The time taken for each file varied considerably, indicating that the information retrieval difficulty was not uniform; this casts some doubt on the utility of our results. However, with more subjects the counterbalancing will largely negate these effects.

## 6 Future Work

We have identified numerous avenues of future work. Maximizing efficacy of the visualizations is an obvious extension and will be greatly aided by performing further work on the user study. Extensions to make the visualization more applicable to digital forensics, such as providing support for collaborative analysis and showing file boundaries in the visualization will be explored.

## 7 Conclusions

Visualization for digital forensic string search is a virtually untouched field of research. Our visualization model appears to be effective as an aid for digital forensic string search, but needs full validation through further user studies, as well as further specialization for digital forensics.

**Acknowledgements.** This work was supported in part by NSF Grant #0313885.

## References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
2. Beebe, N., Dietrich, G.: A New Process Model for Text String Searching. Springer, Norwell (2007)
3. Beebe, N.L., Clark, J.G.: Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. In: Digital Investigation, September 2007, vol. 4(suppl. 1) (2007)

4. Byrd, D.: A scrollbar-based visualization for document navigation. In: Proceedings of the Fourth ACM International Conference on Digital Libraries (1999)
5. Don, A., Zheleva, E., Gregory, M., Tarkan, S., Auvil, L., Clement, T., Shneiderman, B., Plaisant, C.: Discovering interesting usage patterns in text collections: integrating text mining with visualization. In: CIKM 2007: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 213–222. ACM Press, New York (2007)
6. Forte, D.: The importance of text searches in digital forensics. In: Network Security, April 2004, pp. 13–15 (2004)
7. Free Software Foundation. Tool: GNU Grep
8. Harper, D., Koychev, I., Sun, Y., Pirie, I.: Within-document retrieval: A user-centred evaluation of relevance profiling. In: Information Retrieval, vol. 7, pp. 265–290 (2004)
9. Harper, D.J., Coulthard, S., Yixing, S.: A language modelling approach to relevance profiling for document browsing. In: JCDL 2002: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital Libraries, New York, NY, USA (2002)
10. Havre, S., Hetzler, E., Whitney, P., Nowell, L.: ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics* 8(1), 9–20 (2002)
11. Hearst, M.A.: Tilebars: visualization of term distribution information in full text information access. In: CHI 1995: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, pp. 59–66. ACM Press/Addison-Wesley Publishing Co (1995)
12. Mandia, K., Prorise, C., Pepe, M.: Incident Response & Computer Forensics. McGraw-Hill/Osborne, California (2003)
13. Mann, T., Reiterer, H.: Case study: A combined visualization approach for www-search results. In: Proc. IEEE Information Visualization 1999, pp. 59–62 (1999)
14. Mann, T.M.: Visualization of WWW-search results. In: DEXA Workshop, pp. 264–268 (1999)
15. Mao, Y., Dillon, J.V., Lebanon, G.: Sequential document visualization. In: *IEEE Transactions on Visualization and Computer Graphics*, November/December 2007, vol. 13(6), pp. 1208–1215 (2007)
16. Schwartz, M., Hash, C., Liebrock, L.: Term distribution visualizations with a focus+context model. Technical report, New Mexico Institute of Mining and Technology (2008), <http://cs.nmt.edu/~liebrock/papers/SchwartzHashLiebrock.pdf>
17. Splunk, Inc. Application: Splunk
18. Paley, W.B.: TextArc: Showing word frequency and distribution in text. Poster presented at IEEE Symposium on Information Visualization (2002)
19. Whittaker, S., Hirschberg, J., Choi, J., Hindle, D., Pereira, F.C.N., Singhal, A.: SCAN: Designing and evaluating user interfaces to support retrieval from speech archives. In: Research and Development in Information Retrieval, pp. 26–33 (1999)
20. Wong, P.C., Cowley, W., Foote, H., Jurrus, E., Thomas, J.: Visualizing sequential patterns for text mining. In: INFOVIS 2000: Proceedings of the IEEE Symposium on Information Visualization 2000, p. 105 (2000)
21. Zhang, J.: Visualization for Information Retrieval, 1st edn. Springer, Heidelberg (2007)

# GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool\*

Leevar Williams, Richard Lippmann, and Kyle Ingols

MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02173

{LCWILL, LIPPMMANN, KWI}@LL.MIT.EDU

**Abstract.** Attack graphs enable computation of important network security metrics by revealing potential attack paths an adversary could use to gain control of network assets. This paper presents GARNET (Graphical Attack graph and Reachability Network Evaluation Tool), an interactive visualization tool that facilitates attack graph analysis. It provides a simplified view of critical steps that can be taken by an attacker and of host-to-host network reachability that enables these exploits. It allows users to perform “what-if” experiments including adding new zero-day attacks, following recommendations to patch software vulnerabilities, and changing the attacker starting location to analyze external and internal attackers. Users can also compute and view metrics of assets captured versus attacker effort to compare the security of complex networks. For adversaries with three skill levels, it is possible to create graphs of assets captured versus attacker steps and the number of unique exploits required. GARNET is implemented as a Java application and is built on top of an existing C++ engine that performs reachability and attack graph computations. An initial round of user evaluations described in this paper led to many changes that significantly enhance usability.

**Keywords:** attack graph, visualization, treemap, security metrics, adversary model, network, vulnerability, exploit, attack path, recommendation.

## 1 Introduction

Attack graphs have been proposed by many researchers as a way to identify critical network weaknesses, construct adversary models, analyze network security, and suggest changes to improve security. Researchers and commercial companies have developed many differing approaches to generating attack graphs [8, 12, 14, 16, 18]. An annotated review of many of these approaches is available in [7].

Although there are various representations, the overall concept of attack graphs remains the same: they show the ways an attacker can compromise hosts within a network. Attack graphs are constructed by starting an adversary at a given network location and examining how the attacker can progressively compromise vulnerable hosts, using information about software vulnerabilities and network reachability.

---

\* This work is sponsored by the United States Air Force under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

We have developed a system called NetSPA, or Network Security and Planning Architecture, which efficiently generates attack graphs for large complex networks. A full description of the system can be found in [4, 8]. It imports data from common sources, including vulnerability scanners, firewall configurations, and vulnerability databases. This information is used to generate attack graphs, make recommendations for improving network security, and compute important network security metrics. In a previous paper [21], we described a new interactive cascade display for attack graphs that incorporates treemaps to compactly display network subnets and shows host-to-host reachability as well as attack graph data. A preliminary Java-based tool that provides a Graphical User Interface (GUI) to NetSPA and creates these displays using NetSPA as a computation engine was also presented.

This paper describes GARNET (Graphical Attack graph and Reachability Network Evaluation Tool), an improved tool that incorporates the interactive cascade display [21] with the addition of many new capabilities and features. First, it provides a greatly extended and redesigned GUI. This new interface was designed based on careful evaluations and feedback (described in this paper) that were provided by five users familiar with attack graphs. Second, it supports “what-if” analyses for determining the effects of following recommendations for patching hosts, adding and removing vulnerabilities, and modeling adversaries with three skill levels that start from either inside or outside a network. The differing network models created through consecutive applications of “what-if” changes are saved and the results for different variants can be easily compared.

A final major new feature of GARNET is the ability to compute security metrics for complex networks that indicate how security has changed and if one network is more or less secure than another. This addresses a major shortcoming in the security field. Our ability to rapidly construct attack graphs using NetSPA provides an opportunity to develop attack metrics that overcome the limitations of past attempts.

The remainder of this paper describes GARNET in detail. The following section provides an overview of related work on attack graph displays and security metrics. Section 3 describes the NetSPA system. Sections 4 and 5 show the visual representations used by the tool and describe the supported user interactions, including the generation of “what-if” scenarios. The security metrics and adversary models used by GARNET are presented in Section 6, and Section 7 presents results from user evaluations of the tool. This is followed by a discussion of future work in Section 8 and a conclusion in Section 9.

## 2 Related Work

### 2.1 Attack Graph Displays

Significant past research has focused on visualizing and interacting with attack graphs as summarized in [7, 21]. Most recently, two commercial companies have begun to provide attack graph displays. The RedSeal Security Risk Manager [16] reads vulnerability information from network vulnerability scanners and topology information from firewall and router configuration files to create a node-link network topology diagram. This network diagram is initially laid out automatically. System administrators can then collapse and manually reposition hosts and subnets to create easily understandable

displays that accurately represent a conceptual view of the network topology. The display identifies exploitable vulnerabilities and, on top of the network diagram, displays threat paths that an attacker can use to gain access to resources in the network. This tool only computes a few security metrics for a single adversary model.

The second commercial product, Skybox [18], provides a similar network view. However, it requires active agents to capture network topology and host vulnerability information. Reachability is computed and attack paths are shown in a separate display as arrows between individual hosts or servers. The application allows “what-if” analyses to be performed through the simulation of attacks and proposed changes to the network. It is limited, though, by the fact that it does not show the entire attack graph but only displays parts of the overall graph that contain specified target hosts.

GARNET incorporates good aspects of the above commercial displays as well as the cascade display we described in [21] that uses treemaps to display subnets. It provides a compact and fully interactive view of an attack graph that can be related to the underlying network and allows users to generate hypothetical, “what-if” scenarios. As in the RedSeal display, hosts and subnets are laid out automatically but can be repositioned manually to obtain a more intuitive display. Unlike Skybox, no network agents are required, vulnerability data is read from a number of open-source and commercial vulnerability scanners, and network topology information and filtering rules are read from firewall and router configuration files. GARNET computes hosts-to-host reachability, attack graphs, and multiple important security metrics for three graded adversary models to assess overall network security.

## 2.2 Security Metrics

Many different metrics have been proposed in the past to assess different aspects of security, such as the average number of vulnerabilities found per host by a network vulnerability scanner [5], but none accurately measure the overall security of diverse networks. These point measurements fail to take context into account, including the overall network topology, all vulnerabilities, and an adversary model that includes a starting location, goals, and steps that can be taken to achieve these goals. What is desired are metrics that: are accurate, objective, and well defined; can be measured automatically; are easy to understand and explain; and provide insight into underlying causes of both security and insecurity.

Our approach to developing metrics for inclusion in GARNET was motivated by current best practices for assessing network security. The current, most often-used approach is to use “red teams”. These are security experts who attempt to reach a specified goal in a network from a starting location and with a given amount of initial network knowledge. For example, [6] describes an interesting set of experiments where a red team attempts to access a database as extra layers of protection are added to provide “defense in depth”. The metric used in these experiments is red-team effort as measured by person hours required to develop and launch attacks. This approach includes an adversary model (the red team) and uses the actual network for experimentation. It also produces an objective metric (red-team person hours), has high face validity, and can uncover unexpected weaknesses. Unfortunately, it is expensive, cannot be automated, is difficult to replicate, and often is impossible to perform because it can disrupt essential services.

An alternative to using live red teams is an approach called Mission-Oriented Risk and Design Analysis, or MORDA, which is described in [2, 3]. Experts are enlisted to understand a network and its mission, suggest effective adversarial goals that disrupt the network, construct adversary models, and develop attacks that reach goals. Attacks are then analyzed by comparing their cost and visibility. One major goal of a MORDA analysis is to make sure that there are no low-effort, stealthy attacks that an adversary could use to compromise a network’s mission. This analysis has much of the flexibility of a live red team experiment and can be used in a planning stage before a network exists. It does not have the realism of a red-team experiment, but it explicitly includes multiple adversary models, attacker goals, and attacker costs. Unfortunately, it is labor intensive, requires the cooperation of a diverse group of experts, and thus is not frequently used.

Security metrics computed by GARNET are designed to support an automated form of MORDA analysis. Two metrics are provided to measure attacker effort, and one of these indirectly measures attack visibility. These measurements make it possible to identify the existence of attacks that can potentially be used to capture a network’s assets. GARNET also models adversaries with three distinct skill levels. These adversary models and metrics make it possible to compare the security of different networks by examining the adversary skill level and effort required to compromise these networks. The metrics and adversary types are further described in Section 6.

### 3 NetSPA

In previous work, we described an efficient approach to generating a new type of attack graph, the multiple-prerequisite (MP) graph, that scales well to large enterprise networks. Descriptions of the NetSPA tool that generates MP graphs are available in [4, 21]. Although MP graphs are not explicitly displayed in GARNET, an underlying MP graph data structure is used to create its interactive display.

NetSPA models both hosts and network infrastructure devices such as firewalls and routers. It assumes that hosts can have one or more open ports that accept connections from other hosts and that ports have zero or more vulnerabilities that may be exploitable by an attacker. Individual vulnerabilities provide one of four access levels on a host: “root” or administrator access, “user” or guest access, “DoS” or denial-of-service, or “other”, indicating a loss of confidentiality and/or integrity. Vulnerabilities can either be exploited remotely from a different host or only locally from the vulnerable host. Currently, it is assumed that an attacker obtains a host’s reachability if “root” or “user” access is achieved. Attackers can also obtain credentials when compromising a host. Credentials refer to any information that can be used to gain access to another host or other network resources such as a password or a private key, and they are used to model trust relationships. Reachability and credentials serve as prerequisites for exploitation of other vulnerabilities.

NetSPA also incorporates a simple model of host asset values. Each host is assigned an asset value representing the worth to a network defender of the worst-case compromise of that host’s confidentiality, integrity, or availability. Asset values currently default to 10 for all hosts and are typically hand-assigned to higher values for

critical hosts, such as key servers or hosts containing confidential information. They are primarily used for prioritizing recommendations and computing security metrics.

NetSPA uses an import utility, written in PERL, to read in raw data such as Nessus scans, firewall rulesets, and National Vulnerability Database (NVD) records [13], and convert the data into a custom binary file format. The main computation engine, written in C++, is responsible for reading in the binary file, computing reachability, generating attack graphs, analyzing the graphs to generate recommendations, and computing security metrics. The computation engine was not originally designed to support efficient “what-if” analysis. It was extended to support this capability by adding a network model “delta” system. GARNET can use this system to make small, hypothetical changes to the network as small “delta” objects to the network model, and the rest of the computation engine can then operate on the delta as if it were a full network model.

## 4 GARNET Tool and Network Visualization

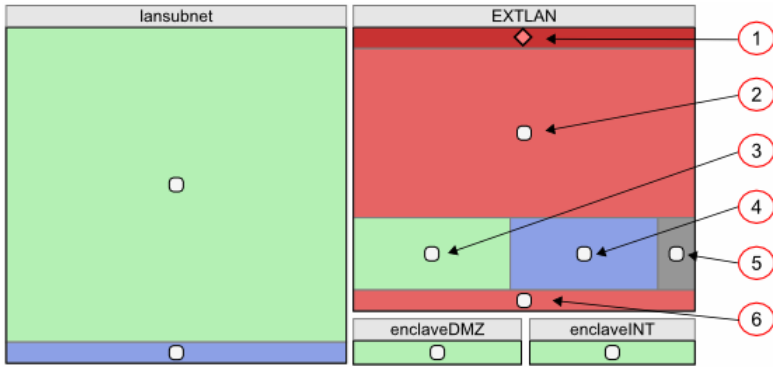
GARNET is a Java-based graphical user interface built on top of the NetSPA engine. We developed a set of bindings between the GUI’s Java and NetSPA’s C++ code using the SWIG toolkit [19]. It generates a shared library which the tool can load and drive programmatically to perform necessary tasks.

GARNET presents an MP attack graph in a readable and concise fashion while preserving much of the essential information. Important features of the nodes are conveyed by grouping, size, and color, while other attributes and edge information are initially hidden and can be displayed on demand. This approach is inspired by the semantic substrate displays described in [17].

Nodes from the MP attack graph are grouped by subnet and a treemap layout is used to illustrate these groupings. Subnets are represented within the display by rectangular blocks labeled with the name of the subnet. Smaller rectangles within each block correspond to host groups; the hosts in a host group can all be compromised to the same extent, are all in the same subnet, and are treated identically by all network filtering devices. Each group is colored according to its level of compromise, indicating one of four access levels (“root”, “user”, “DoS”, or “other”) or that the group of hosts cannot be compromised. The relative size of a host group is proportional to the number of hosts it contains.

Fig. 1 provides an example of this visualization for an actual network with 4 subnets: an external subnet labeled “EXTLAN” containing 119 hosts, an internal subnet labeled “lansubnet” containing 129 hosts, and two smaller subnets containing one host each. The attacker starting location is indicated by the dark red band with a red diamond in the center at the top of the “EXTLAN” subnet (1). The light red rectangle below this band (2) shows a group of hosts compromised at the “root” level. The light-green rectangle below (3) represents hosts that are not compromised, the blue rectangle to the right (4) represents hosts compromised at the “other” level, and the smaller gray rectangle (5) represents hosts compromised only at the “DoS” level. The lower red rectangle in this subnet (6) represents hosts also compromised at the “root” level but with different reachability to other hosts than those of the upper red rectangle (2). The meaning of the rectangles in the other subnets of this figure is similar. Rectangles





**Fig. 1.** Subnet groups arranged in grid layout with weighted sizing

within each subnet group are laid out according to the strip treemap algorithm presented in [1]. This particular algorithm was employed because it solves the bin-packing problem of completely filling a rectangular region with boxes of different areas, and it produces dimensions with reasonable aspect ratios.

Within the display area, subnet groups can be repositioned and resized by direct manipulations (clicking and dragging). A user can thus place subnets into an arrangement that represents a physical or similarly intuitive view of the network. The interface also provides a variety of automatic layouts. Users can choose vertical or horizontal layouts, or a grid arrangement for the subnets. An “auto-sizing” function can also be used to size the subnet rectangles to be proportional to the number of hosts represented. The default is a grid layout with the subnet rectangles sized to be proportional to the number of hosts contained. This is a useful initial configuration because it clusters the subnets and quickly conveys their relative sizes and the overall scale of the network. This layout was used to generate Fig. 1.

## 5 User Interaction

GARNET’s user interface supports three separate modes of interaction: Network Map, Attack Graph, and Summary Plots. Each mode differs in the information that is available and the ways in which the display can be manipulated. A user can toggle between these different modes by clicking one of three tabs located in the upper left of the GARNET side panel shown in Fig. 2.

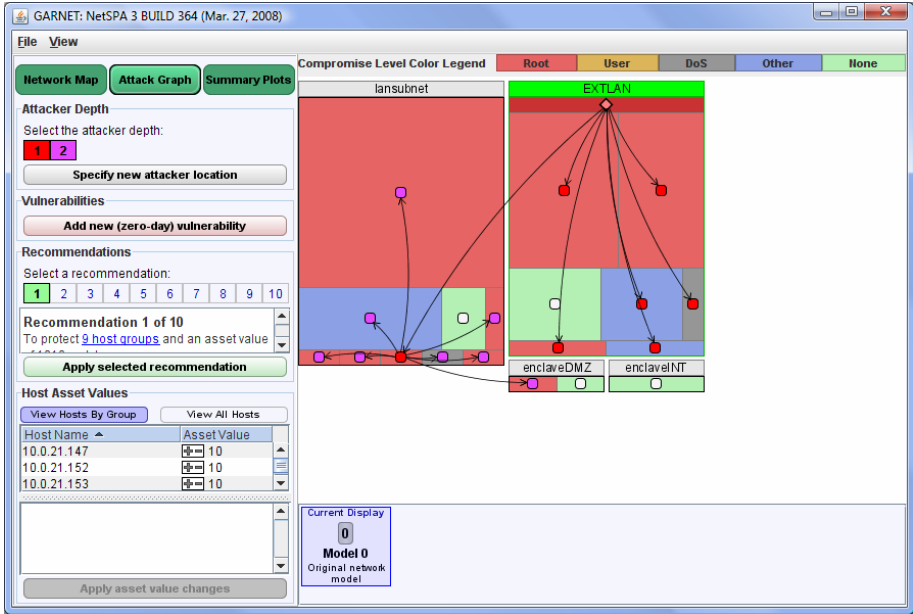
The Network Map mode provides an overview of the network topology and hosts. In this view, the side panel contains controls for displaying reachability between subnets. The user can select a subnet from a drop-down list or directly by clicking on its rectangle, and options are presented for showing incoming and outgoing connections between the groups of hosts within that subnet. The reachability amongst host groups is illustrated by directed edges drawn between pairs of groups. These edges indicate that the hosts in the source group can connect to ports on hosts in the target group. Fig. 2 shows the tool in this mode, with outgoing reachability being displayed from the “EXTLAN” subnet.



**Fig. 2.** GARNET in Network Map mode. The arrows indicate outgoing reachability for the EXTLAN subnet.

GARNET's side panel also contains an information pane that lists hosts and vulnerabilities for a particular host group, a subnet, or the entire network, depending on the selection that is made in the display area. The per-host information includes the IP address, asset value, and highest level of compromise achievable by the attacker, as well as a breakdown of the specific vulnerabilities that exist on the host's open ports. The vulnerability listings include details such as a description, the locality and effect, and the affected host ports. Providing this information allows a system administrator to drill down into the network and identify attributes of an individual host or vulnerability and enables them to understand the overall connectivity between subnets.

GARNET's Attack Graph mode, selected by the middle tab in the control panel, provides an interface for direct interaction with NetSPA's network model. The attacker entry point into the network or starting location is specified by selecting one of the subnets in the right-hand display. This selection defines a critical characteristic of the current adversary model. NetSPA, by default, builds its attack graph under the assumption that an attack is able to originate from any source IP address. This is a good model of an attack originating from anywhere on the Internet. The range of addresses can also be constrained to model a more limited adversary or environments where IP spoofing is restricted. Once a starting location is selected, the attack graph is built and groupings are created for the hosts based on reachability and level of compromise. The user is allowed to change these parameters for the attacker at any time, and the network model is immediately altered and redisplayed. The edges of the attack graph are displayed incrementally through the use of the depth control shown in



**Fig. 3.** GARNET in Attack Graph mode. The arrows indicate attack paths and show the first two attacker hops.

the upper left of Fig. 3 under the heading “Attacker Depth”. For the first attacker hop (corresponding to a depth of one), edges are drawn from the attacker node to all groups of vulnerable hosts that are directly reachable from the attacker’s initial location. For each subsequent level of depth, edges are drawn from the host groups compromised at the previous depth to the next set of compromisable hosts. As the edges are revealed, the target nodes become colored according to their level of depth. In Fig. 3, the first two attacker hops are shown for the given network. Nodes representing hosts compromised at one hop are colored red and nodes compromised at two hops are colored purple.

To aid administrators in defending their networks, NetSPA automatically generates recommended actions to improve a network’s security posture. The GUI allows users to explore these recommendations and their impact. A recommendation is characterized by a list of vulnerabilities that must be removed from a certain set of hosts to protect them from being compromised at an administrator or user level. This information, along with the combined asset value of the protected hosts and the number of affected host groups, is presented for each of the recommendations. The user can immediately view the effect of patching a set of vulnerabilities by choosing the “Apply Selected Recommendation” button. This action creates a new network model with the vulnerabilities removed from the indicated host ports, rebuilds the attack graph, and updates the display. In practice, this takes only a few seconds even for networks with thousands of hosts. Host asset values can also be modified in GARNET to examine the effect on security metrics and recommendations.

In addition to viewing the effects of removing vulnerabilities by applying recommendations, a user can introduce vulnerabilities into the network. The “Add Zero-day

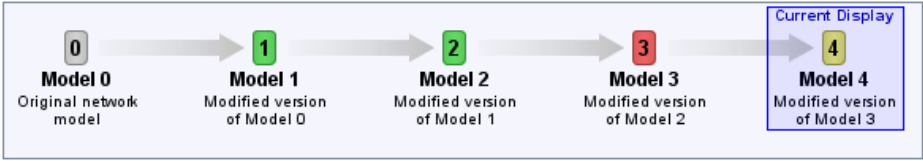
Vulnerability” button makes it possible to model adversaries with different skill levels by placing new vulnerabilities on selected ports of all hosts. We currently suggest performing analyses using three adversary models. The *simple* adversary has exploits for all known vulnerabilities. Since exploits are often available on the Internet a few days after a vulnerability is announced (e.g. [20]), this represents an attacker who downloads exploits at low cost but is not able to create new exploits. This is the default adversary model. The *single-zero-day* adversary has all the exploits of the simple adversary but is also able to create or buy one exploit for a currently non-public vulnerability. This represents a more capable attacker who can craft or purchase one zero-day exploit specifically designed to penetrate the network being analyzed. In the underlying NetSPA engine, the single zero-day exploit can be selected manually or it can be determined automatically by building an attack graph for each possibility to find the one that gains the most assets. The current GARNET interface supports manual selection of a zero-day exploit by selecting a protocol and port. The new vulnerability is then placed on all hosts with this open port. A *comprehensive-zero-day* adversary model is assumed to have an exploit for every open port in the network. This provides an upper bound on the percentage of network assets that can be captured by attackers that use server vulnerabilities. This adversary is selected by placing zero-day vulnerabilities on every port using the “All Open Ports” option in the “Add Zero-day Vulnerability” dialog box.

The third and final mode of GARNET interaction, selected by the right tab labeled “Summary Plots” in the side panel, allows users to compute and compare two security metrics and also to enumerate vulnerability types in the network. Selecting the “Vulnerability Types” plot creates pie charts that show the types of vulnerabilities in the network. Selecting the other plot types creates security metric graphs as described in the following section.

“What-if” experiments performed in GARNET generate new network variants and the Summary Plots mode makes it easy to compare these different networks. In particular, a new network model is created when a user applies recommendations, adds zero-day vulnerabilities, or changes host asset values. The interface uses a timeline component to visualize and manage the progression of models that are produced as a consequence of a user’s incremental modifications. These user-initiated changes, when applied in succession, are cumulative, and the timeline enforces the notion that each subsequently generated model is a different version of the previous one. Positioned at the bottom of the window, the component ties together the three modes of operation – changes made in one mode are reflected in the other two. The timeline contains an icon, label, and short description for each distinct version, beginning with the original network model that was loaded from disk.

Each item displayed in the timeline is selectable by mouse click and the current selection determines the model that is visually represented in the Network Map and Attack Graph modes. A view of the component is presented in Fig. 4, and it contains five different model versions, where “Model 4” is selected. The state of the side panel controls is also coordinated with the currently selected model, and the controls are updated whenever the selection changes.

The charts in Summary Plots mode are likewise populated with data from the current model. Additionally, this mode allows multiple items to be simultaneously selected from the timeline, enabling side-by-side comparisons of the data from different versions



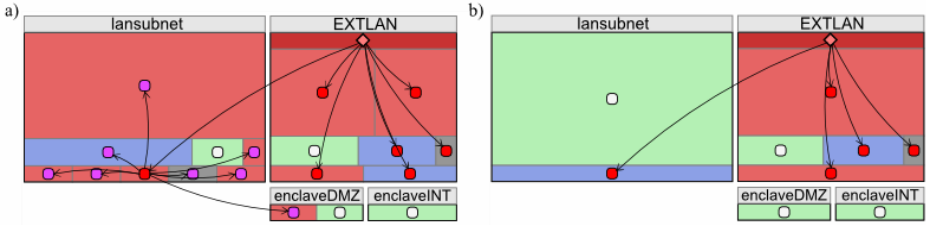
**Fig. 4.** Example of the timeline component displaying a series of network model versions

of the network model. This timeline control facilitates the “what-if” experimentation that GARNET supports. Several changes can be incrementally applied to the network model, and the user can quickly and easily jump between the different versions to examine the altered states of the network.

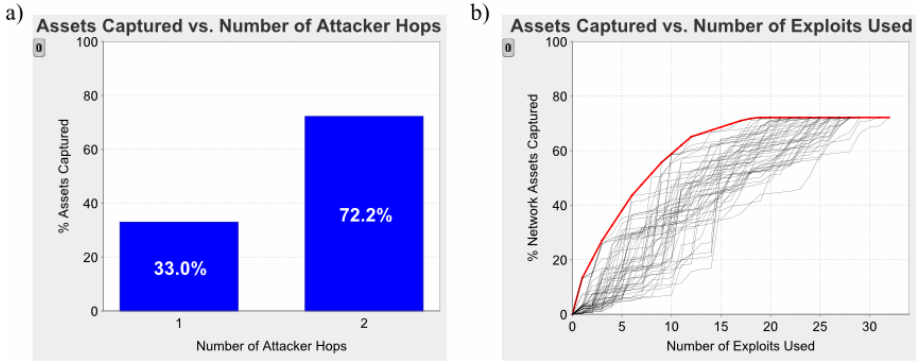
## 6 Security Metric Plots

GARNET’s security metrics graph the percentage of network assets captured by an attacker as a function of different measures of attacker effort. These metrics assume the attacker goal is to maximize the assets captured for each level of effort. They can be used to compare the security of different networks if it is assumed that a network is more secure when an attacker captures fewer assets for the same effort level. Our first security metric computes assets captured after each successive attacker hop. A hop indicates the set of hosts that become compromised at the corresponding depth in the attack graph. For example, the first hop represents the hosts that can be directly compromised from the adversary’s starting location. Each subsequent attacker hop represents those hosts that can be compromised from the set of already compromised hosts. A hop signifies extra effort on the part of the attacker since it delays capture of the most important assets, requires more involved attacks, and provides more opportunity for detection.

A second metric measures the number of unique exploits required by an adversary to capture network assets. This metric assumes that it is much more work to obtain or develop and test a new exploit than it is to reuse an existing exploit. One approach to computing this metric would be to find the optimal set of unique exploits that would be used by an omniscient attacker with full knowledge of a network for different numbers of exploits. We feel that this would be misleading because actual attackers have a limited horizon and can only probe hosts that are reachable from currently compromised hosts. Thus, at each attack step, we randomly sample from the exploits that have not yet been used and that compromise one or more hosts that are reachable from currently compromised hosts. We produce many curves this way that represent the range of capabilities for a limited-horizon attacker. If enough curves are created, the upper limit of all the curves approximates the best performance that would be obtained by an omniscient attacker and the spread of curves represents the spread that would be seen in actual attackers. Since NetSPA computes attack graphs rapidly, we currently sample 50 random exploit curves. This simple random sample approach avoids local minima caused by multiple firewalls and has been effective for actual and simulated test networks.



**Fig. 5.** Attack graph for a simple adversary a) before and b) after following a recommendation

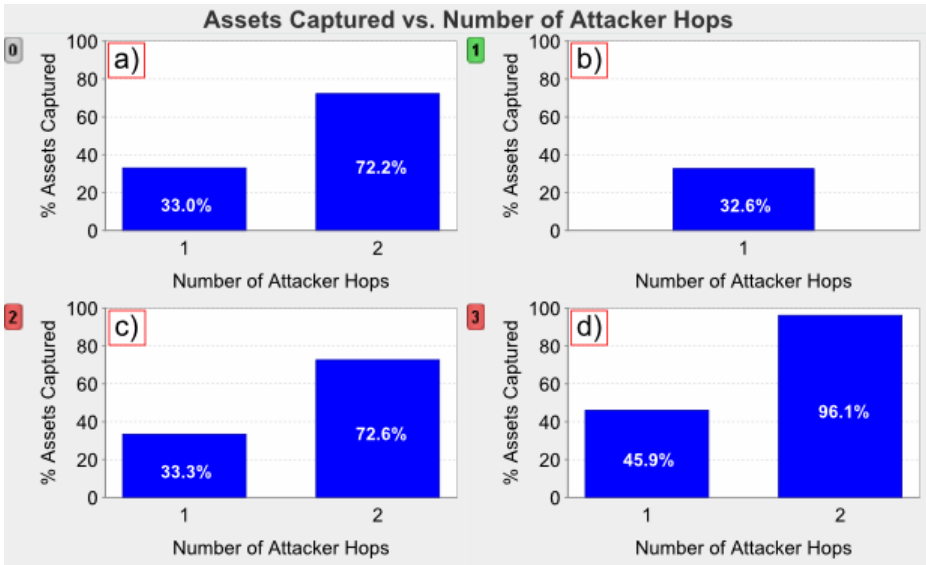


**Fig. 6.** Security metrics for a simple adversary on the baseline network

Fig. 5 shows the attack graph for a simple adversary before (a) and after (b) the most effective recommendation is applied. Before the recommendation is applied, the attacker compromises many hosts in the upper right “EXTLAN” network and one host in the upper left “lansubnet” network on the first hop at a root level. On the second hop, further hosts in the “lansubnet” are compromised at root level as well as a host in the small “enclaveDMZ” network. When the recommendation is followed and the stepping stone host in “lansubnet” is patched, the attacker can no longer compromise hosts outside the starting “EXTLAN” network at the root level. These attack graphs provide the low-level details that explain how hosts are compromised.

Fig. 6 illustrates our two security metric plots for the simple adversary before the recommendation is followed. These high-level metrics summarize the network security posture for the simple adversary. The graphs in Fig. 6 show the combined asset values of hosts compromised at a root or administrator level. The first plot of assets captured versus hops (a) shows that 33% of all network assets are captured after one hop and roughly 72% are captured after two hops. The second graph (b) consists of curves representing samples of 50 different randomized attackers, indicating that roughly 18 to 32 unique exploits need to be available to capture 72% of the total asset value in this network. A large number of unique exploits is required because this is an extremely heterogeneous network.

Fig. 7 shows four assets captured versus attack hop graphs displayed together in GARNET’s Summary Plots mode, after three “what-if” scenarios have been generated. The upper left plot (a) is the graph for the baseline network and the simple



**Fig. 7.** Assets captured versus hop curves for a) a simple attacker on the baseline network, b) a simple attacker on the patched network, c) a single-zero-day attacker on the patched network, and d) a comprehensive zero-day adversary on the patched network

adversary. The graph to the right (b) shows the effect of applying the most effective recommendation that blocks access into the second subnet. It illustrates that only one hop is possible and that the simple attacker captures roughly 33% of the network assets in that hop. These two graphs clearly indicate that following the recommendation produces a network that is more secure. The maximum asset value captured is more than halved and fewer assets are captured after patching. The bottom left graph (c) shows how many assets can be captured by a single-zero-day adversary on the patched network. This adversary uses a zero-day exploit to access the patched server and captures the same assets that were available to the simple attacker without the patch. The applied patch is thus ineffective for this more capable adversary. Protection could be provided by more advanced access control or by further compartmentalizing the network. The lower right plot (d) shows how many assets can be captured by a comprehensive-zero-day adversary. This upper bound on attacker capabilities shows that patching and filtering in the existing network is providing some protection. Although plots of assets captured versus unique exploits required are not shown due to space limitations, they support the above conclusions.

## 7 Usability Analysis

User evaluations were performed to assess the effectiveness of GARNET's visual representation and GUI design. Previous research in the area of user interface evaluation [11] suggests that formal methods, such as formulating a proper analysis model or applying a computerized procedure, are impractical for most applications. As a result,

we chose a more informal technique known as heuristic evaluation. This method involves presenting evaluators with a user interface and asking them to subjectively judge the interface according to a set of usability guidelines, known as heuristics.

A group of five evaluators was assembled, all of whom were knowledgeable about the target domain. Each person was given a brief overview of GARNET and the evaluation procedure, as well as a list of heuristic guidelines. For the list of guidelines, we used a widely accepted set of principles developed by Jakob Nielsen [10] consisting of ten heuristics. The evaluators were encouraged to explore the interface as thoroughly as possible, using the guidelines to help them identify aspects of the tool that represented either a positive feature or a usability problem. They were also asked to rate each problem in terms of severity and to suggest a solution if possible. The evaluations were performed independently and at each individual's own convenience. Results of the evaluations were used to refine GARNET's initial design and create the improved version presented in this paper.

Evaluators produced descriptions of strengths and weaknesses of the interface and recommendations for improvement. The number of comments ranged from 14 to 54 items, with a median of 44. In general, users perceived the layout of the interface as clean and simple, and they liked the use of the treemaps and colors for conveying information. They also responded positively to the supported interactions for generating new versions of the network, the ability to easily jump between these different models, and the responsiveness of the system in performing these actions. In addition, they commented on the intuitiveness of being able to directly manipulate the subnet groups within the display area.

From the remaining comments that pointed to drawbacks of the system, we extracted a list of 55 distinct items representing bugs and specific features that could be implemented or improved. An example of one of the more critical issues related to the overall organization of the interface controls. In the evaluated version of the interface, the controls for manipulating host asset values were only visible in Network Map mode, and the Network Information panel was in its own separate tab and could be accessed from all three modes. The majority of the evaluators found it difficult to locate information and thought some of the interactions were inconsistent across modes. To address these concerns, the side panel controls were reorganized so that the network information and reachability controls appear in Network Map mode and controls for all interactions that alter the network model are unified in the Attack Graph mode.

Another problem involved the implementation of the timeline. In its original form, a set of arrows was used to switch between models in two of the modes, while direct selection of the icons was allowed in the third mode. In addition, the labels and icons were not very informative. All of the evaluators mentioned that the interaction with the timeline was inconsistent and non-intuitive. The component was redesigned to have more descriptive icons and text, be consistent across modes, and use better visual cues to indicate the selection and progression of network models.

Several additional comments were addressed to further extend the functionality of the interface and increase its ease of use. These improvements included adding support for modeling a comprehensive-zero-day attacker by allowing vulnerabilities to be added to all open ports and enabling users to continuously modify the attacker starting location. Also, a legend explaining the colors used for levels of compromise in the attack graph was incorporated into the display.



The evaluations we collected were a valuable source of feedback about the usability of GARNET’s interface. They confirmed the tool’s effectiveness in conveying information about the attack graph and providing a set of interactions that allows users to experiment with different scenarios. The recommendations we received about problematic areas of the interface helped us develop a more functional design, while many of the comments pointed to larger issues that provide directions for future work.

## 8 Limitations and Future Work

Although GARNET successfully conveys a significant amount of information through its visual representation, it is still somewhat limited in its illustration of overall network topology. The user is able to view reachability links between groups of hosts in different subnets; however, for numerous host groups and dozens of subnets, displaying this reachability all at once can produce a confusing jumble of edges. A potential alternative to displaying the individual links would be to utilize a flow map technique [15], resulting in merged edges whose varying widths indicate the number of inbound/outbound connections. Furthermore, the tree structure of the flow map could be used to dictate the initial layout of the subnet groups, and filtering devices (such as routers and firewalls) could be shown along the edges between the subnets they connect. This view would provide a clearer picture of the physical connectivity of the network. We would also like to explore methods of subnet aggregation to display large networks with many subnets.

Further work could also extend our adversary models by enabling client-side attacks in which an attacker uses a malicious server to compromise a vulnerable client machine or sends malicious email attachments. We would also like to explore other measures of adversary effort such as those related to the complexity of launching attacks or the cost of obtaining attacks. Some of these measures, such as a field called “Access Complexity,” are already specified in the Common Vulnerability Scoring System [9] and can be automatically extracted from the NVD [13].

Finally, GARNET should be exposed to further rounds of user testing, including empirical evaluation by system administrators. This form of user evaluation would involve presenting a set of target users with the interface and measuring how well they perform various tasks that focus on important aspects of the tool’s functionality.

## 9 Conclusions

We have developed GARNET as a visualization tool for attack graphs and network reachability. It produces a compact visual representation of a network and the ways in which it can be compromised by an attacker, as well as metrics that summarize the overall security of the network and recommendations that suggest preventative actions. Information about individual hosts, the vulnerabilities they possess, and the reachability between them is easily accessible through the Network Map mode. The interface enables users to perform “what-if” experimentation by applying recommendations and modifying host asset values. The adversary model can be changed by

varying the attacker location and introducing new zero-day vulnerabilities. Because the computation times for constructing attack graphs with thousands of hosts are typically less than a few seconds [4], we can dynamically regenerate displays at interactive speeds. Finally, security metrics are calculated and displayed in chart form to facilitate comparisons between networks. User testing provided excellent feedback that improved many aspects of GARNET's design and also provided insights for future development which we hope to apply to subsequent iterations of this tool.

**Acknowledgements.** We would like to thank Seth Webster, Tamara Yu, and Chris Connelly for their participation in the user evaluations and their feedback on GARNET's interface.

## References

1. Bederson, B., Shneiderman, B., Wattenberg, M.: Ordered and quantum treemaps: making effective use of 2d space to display hierarchies. *ACM Transactions on Graphics* 21(4), 833–854 (2002)
2. Buckshaw, D., Parnell, G., Unkenholz, W., Parks, D., Wallner, J., Saydjari, S.: Mission oriented risk and design analysis of critical information systems. *Military Operations Research* 10(2), 19–38 (2005)
3. Evans, S., Heinbuch, D., Kyle, E., Piorkowski, J., Wallner, J.: Risk-based systems security engineering: stopping attacks with intention. *IEEE Security and Privacy Magazine* 2(4), 59–62 (2004)
4. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: *Proceedings Computer Security Applications Conference (ACSAC)*, pp. 121–130 (2006)
5. Jaquith, A.: *Security metrics: replacing fear, uncertainty, and doubt*. Addison Wesley, Reading (2007)
6. Kewley, D., Lowry, J.: Observations on the effects of defense in depth on adversary behavior in cyber warfare. In: *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June (2001)*
7. Lippmann, R., Ingols, K.: An annotated review of past papers on attack graphs. MIT Lincoln Laboratory, Lexington, MA, Tech. Rep., 2005, ESC-TR-2005-054 (2005)
8. Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Cunningham, R.: Validating and restoring defense in depth using attack graphs. In: *MILCOM 2006*, Washington, DC (2006)
9. Mell, P., Scarfone, K., Romanosky, S.: A complete guide to common vulnerability scoring system version 2.0 (2008) (Accessed 23 April 2008), <http://www.first.org/cvss/cvss-guide.html>
10. Nielsen, J.: Heuristic evaluation. In: Nielsen, J., Mack, R.L. (eds.) *Usability Inspection Methods*. John Wiley and Sons, New York (1994)
11. Nielsen, J., Molich, R.: Heuristic evaluation of user interfaces. In: *Proceedings ACM CHI 1990 Conference, Seattle, WA*, pp. 249–256 (1990)
12. Noel, S., Jajodia, S.: Understanding complex network attack graphs through clustered adjacency matrices. In: *Proceedings Computer Security Applications Conference (ACSAC)*, pp. 160–169 (2005)

13. NVD National Vulnerability Database (2008) (Accessed 11 April 2008), <http://nvd.nist.gov>
14. Ou, X., Govindavajhala, S., Appel, A.W.: Mulval: a logic- based network security analyzer. In: Proceedings of the 14th Usenix Security Symposium 2005, pp. 113–128 (2005)
15. Phan, D., Xiao, L., Yeh, R.B., Hanrahan, P., Winograd, T.: Flow map layout. In: Proceedings of the IEEE Symposium on Information Visualization 2005, pp. 219–224 (2005)
16. RedSeal Systems Inc. (2008) (Accessed 11 April 2008), <http://www.redseal.net>
17. Shneiderman, B., Aris, A.: Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics* 12(5), 733–740 (2006)
18. Skybox Security Inc. (2008) (Accessed 11 April 2008), <http://www.skyboxsecurity.com>
19. SWIG (2008) (Accessed 11 April 2008), <http://www.swig.org>
20. Symantec Corp. Internet security threat report (2008) (Accessed 11 April 2008), <http://www.symantec.com/business/theme.jsp?themeid=threatreport>
21. Williams, L., Lippmann, R., Ingols, K.: An interactive attack graph cascade and reachability display. In: *VizSec 2007*, Sacramento, CA (2007)

# A Graph-Theoretic Visualization Approach to Network Risk Analysis

Scott O'Hare<sup>1</sup>, Steven Noel<sup>2</sup>, and Kenneth Prole<sup>1</sup>

<sup>1</sup> Secure Decisions, Division of Applied Visions Inc., 6 Bayview Ave., Northport, NY, USA

<sup>2</sup> Center for Secure Information Systems, George Mason University, Fairfax, VA, USA  
{ScottO,KennyP}@securedesigns.avi.com, snoel@gmu.edu

**Abstract.** This paper describes a software system that provides significant new capabilities for visualization and analysis of network attack graphs produced through Topological Vulnerability Analysis (TVA). The TVA approach draws on a database of known exploits and system vulnerabilities to provide a connected graph representing possible cyber-attack paths within a given network. Our visualization approach builds on the extensive functionality of the yWorks suite of graphing tools, providing customized new capabilities for importing, displaying, and interacting with large scale attack graphs, to facilitate comprehensive network security analysis. These visualization capabilities include clustering of attack graph elements for reducing visual complexity, a hierarchical dictionary of attack graph elements, high-level overview with detail drill-down, interactive on-graph hardening of attacker exploits, and interactive graph layouts. This new visualization system is an integrated component of the CAULDRON attack graph tool developed at George Mason University.

**Keywords:** network security, attack graph, exploit analysis, vulnerability assessment, visualization, situational awareness.

## 1 Introduction

Powerful analytic tools generally require well-designed user interfaces in order to be fully effective in their designated applications. This is especially true for tools designed to enhance network security: There are order-of-magnitude complexity issues associated with network topology maps, traffic data collection, and expert systems built around attack profile data and traffic correlation. For such analysis to be comprehensible, we need sophisticated visualization and interaction mechanisms.

We describe a visualization component for network attack graph analysis. This is an integrated component of the CAULDRON tool developed at the Center for Secure Information Systems at George Mason University. CAULDRON analyzes network topology and vulnerability data, combined with a comprehensive attack profile database. This Topological Vulnerability Analysis (TVA) [1] generates a complete *attack graph* showing all possible attack paths through a given network. The attack graph represents vulnerable network hosts and exploits that may be launched against them, with attack state transition data determined by the exploits. This attack graph, is still generally too large to be viewed and understood in its entirety.

Our approach to the visualization problem for TVA is based on tools explicitly designed for displaying and interactively analyzing graphs of interconnected nodes. A number of such tools are available, including Tom Sawyer, JGraph, Prefuse, Jviews Diagrammer, and yWorks. After a careful survey, we selected yWorks [2] for this application, based on its extensive feature set, deep and comprehensive Java API, and attractive deployment licensing terms.

The resulting software component provides powerful visualization capabilities for CAULDRON TVA attack graphs. A key feature is the implementation of hierarchical node and edge grouping along lines of *protection domains*. Protection domains are sets of machines with *unrestricted access* to one another's vulnerabilities, forming a completely connected sub-graph. Within each domain, it is sufficient to encode a particular host exploit only once, then implicitly, all hosts within the domain can carry out that exploit. Across domains, the exploits are all explicit. Thus our protection domain abstraction preserves all the information of the complete (ungrouped) graph, including intra-domain exploits. This abstraction reduces complexity within each protection domain from quadratic to linear, providing significant scalability, facilitating analyst navigation and cognition [3].

In our visualization system, a high-level view clearly displays exploit relationships among protection domains, which can be opened individually or in groups for deeper views of attack properties and relationships. A complete listing of active exploits and their associated details is available at all times relative to any selected component. Interactive hardening of on-graph nodes and exploits can be emulated to study the effects of remediation and "what-if" scenarios. Additionally, a suite of interactive layout tools, including manual repositioning of entities, along with full-scale layout algorithms, is continuously available to restructure or simply clean up the display.

The next section gives an overview of the CAULDRON tool, including the nature and utility of network attack graphs. In Section 3, we discuss yWorks architecture and capabilities, in relation to requirements for CAULDRON attack graph visualization. We also describe custom components that meet special requirements, as well as architectural features for performance and visual comprehension. Section 4 then describes the resulting visualization capabilities, including operational scenarios and ideas for future improvements.

## 2 CAULDRON Tool Capabilities

The CAULDRON TVA approach simulates incremental network penetration, showing all possible attack paths through a network. This simulation is based on a detailed model of the network configuration, attacker capabilities, and desired attack scenario. Because of the inherent interdependencies of vulnerability across a network, such a topological approach is necessary for a full understanding of attack risk.

CAULDRON captures configuration details for a network by processing the output of network scanning tools (Fig.1). It combines scans from various network locations, building a complete map of connectivity to vulnerable services throughout the network. It integrates with Nessus, FoundScan, and Retina, and Symantec Discovery. Integration with Altiris is currently under development.

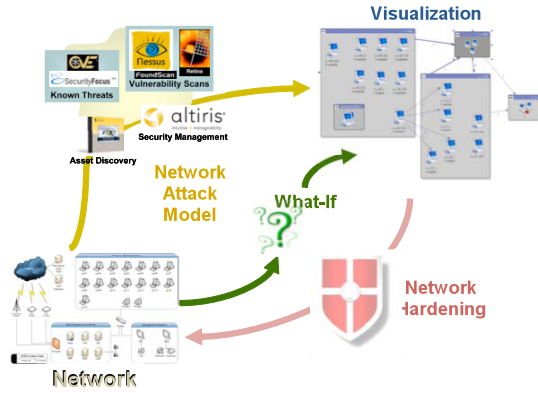


Fig. 1. CAULDRON architecture

CAULDRON maintains a comprehensive database of modeled attacker exploits (currently over 20,000), based on software vulnerabilities reported in various sources, including Symantec DeepSight (a direct XML feed of Bugtraq along with other data), and MITRE’s Common Vulnerabilities and Exposures (CVE). From the input model of network configuration and attacker exploits, CAULDRON computes a graph comprising all possible attack paths through the network. This graph is computed through simulated multi-step attacks according to a given user scenario.

The TVA approach as implemented by CAULDRON is not simply a cross-referencing of security data. Rather, it is a simulation of multi-step network penetration, with a full range of host vulnerability types and network configuration variations. For example, we have implemented exploit rules for buffer overflows, user logins, file transfers, port forwarding, traffic sniffing, spoofing attacks, client-side attacks, and denial-of-service.

Further, the ability to experiment through such what-if analyses is a powerful CAULDRON capability. The analyst can specify a starting point for the attack (the presumed threat source), as well as an attack goal (critical network asset to protect). The analyst can also model the effects of software patches or other mitigation solutions, which are included in the CAULDRON database. Once an attack graph has been computed, CAULDRON analyzes the results and provides recommendations for optimal network defenses [4].

### 3 Attack Graph Visualization

Capabilities for generating and visualizing TVA attack graphs have undergone significant evolution over time. TVA technology was originally limited to computing single attack paths, and the original presentation was a simple table, as in Fig.2(a). Later, the capability for efficient computation of all possible paths was developed, but visualization of the resulting graphs in their full detail is difficult to assimilate, e.g., Fig.2(b). Therefore primitive graph clustering techniques were developed, in special-purpose code, which was cumbersome and had limited interactive capabilities, as shown in Fig.2(c). Later, a more advanced visualization capability was developed

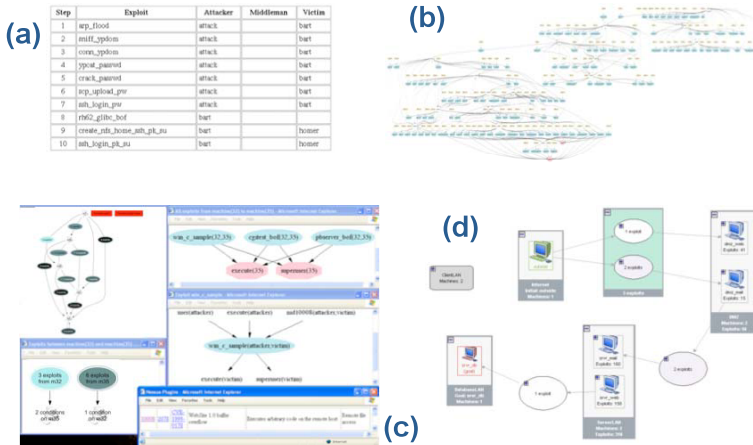


Fig. 2. Evolution of attack graph visualization capabilities

using Tom Sawyer, shown in Fig.2(d), although this exhibited performance problems for larger attack graphs. Our implementation using yFiles adds new analytic and display capabilities, while addressing these problems.

### 3.1 Loading the Attack Graph

The attack graph structure is delivered in an XML document conforming to a specific XML Schema Definition (XSD). We leverage Apache's XMLBeans technology, a Java-to-XML binding framework, to import the attack graph XML. Existing yFiles XML import capabilities require the existence of an Extensible Stylesheet Language Transformation (XSLT) into one of several standardized graph representations, which is not sufficiently general for our purposes. The XMLBeans component uses an XSD file to build a class library corresponding to the internal document structure. Utilities parse XML data structures and ensure conformance with the XSD, allowing us to acquire an attack graph as an organized collection of instantiated Java objects.

### 3.2 The Node Hierarchy

The attack graph is built within yWorks by transforming attack graph machine (host) objects into graph *nodes*, and exploit objects into graph *edges*. Several layers of graphical nesting are also performed. The most fundamental of these is based on protection domains, which are represented within the yFiles graph as *group* or *folder* nodes. (From a display or layout perspective, a group node is essentially a folder node that has been opened, and whose child elements are visible.) Machine nodes within a protection domain are represented as child nodes of the corresponding group or folder. We perform the layout of nodes within a group as the graph is assembled and initially present the graph in its *top-level* layout, in which only protection domains and the exploits connecting them are visible.

Presenting an initial top-level view yields an enormous improvement in the initial performance of the graph layout algorithm. Execution time of layout algorithms increases rapidly with the number of visible nodes and edges to be displayed. Earlier efforts to import large attack graphs were quite time-consuming, primarily due to this initial layout overhead. A top-level layout approach, combined with yFile's ability to perform so-called *incremental* layout algorithms allow us to import and display large attack graphs in seconds that formerly had taken several hours to load.

### 3.3 The Edge Hierarchy

Another feature to enhance layout performance involves the manner in which exploits are represented as edges. Only exploits that connect machines in different protection domains appear as edges in our graph, that is, we suppress edge creation for *intra-domain* edges. This significantly accelerates initial graph setup, as well as subsequent layout steps resulting from interactive and redrawing operations. We also aggregate into a single edge any multiple edges connecting the same pair of nodes. These edge policies provide significant improvements in graph readability in addition to performance enhancements.

In the hierarchical navigation of nodes, no information is lost; one has merely to expand a folder node to acquire information hidden at a lower level. With the edge representation policies described above, it is not possible, in general, to recover a full edge set through simple expansion. Special mechanisms have been implemented to remedy this. Aggregated edges are labeled with an edge count, and edge line thickness also indicates the total number of exploits being represented. Additionally, the tool contains an *exploit table* that displays the full list of exploits, with complete attributes, associated with any single aggregated edge. Simply selecting an edge of the graph populates this table, as shown in Fig.3.

The exploit table also tracks node selection: all exploits associated with a given machine (node), or protection domain (folder or group node) are displayed in the exploit table whenever the node is selected. This includes the display of *intra-domain* exploits, even though these are not explicitly represented by edges. Thus the full set of information provided by an attack graph is always available, and can be viewed in an intuitive way within the user interface. The exploit table allows a "microscopic" analysis of exploit details, while fundamental topology and network relationships are kept simple and understandable within the graph view.

Exploit	From	To	Preconditions	Postconditi...	Family
bt_MicrosoftWindowsMediaPlayer_ASXBuff...	client2	srvr_mail	execute, bugtraq 1980	execute	
bt_WindowsMediaPlayer_ASXBufferOverflow	client2	srvr_mail	execute, bugtraq 2677	execute	
bt_MicrosoftWindowsMediaPlayer_ASXBuff...	client1	srvr_mail	execute (initial), bugtraq 1980 (i...	execute	
bt_WindowsMediaPlayer_ASXBufferOverflow	client1	srvr_mail	execute (initial), bugtraq 2677 (i...	execute	
bt_MicrosoftWindowsMediaPlayer_ASXBuff...	client1	srvr_web	execute (initial), bugtraq 1980 (i...	execute	
bt_MicrosoftWindowsMediaPlayer_ASXBuff...	client2	srvr_web	execute, bugtraq 1980	execute	

Fig. 3. Exploit table from the attack graph visualization tool



### 3.4 Additional Graph Visualization Features

**Hardening:** Viewing vulnerabilities or potential exploits within a network, the analyst is generally faced with multiple options for remediation. These options often involve choosing a machine or set of machines to protect (harden), or identifying specific exploits to protect against. We visually display the effects, in graphical terms, that occur when a specific node or protection domain is hardened or when a specific exploit is neutralized. This involves determining which elements are no longer vulnerable after the hardening, and removing these elements from the attack graph. These elements are placed into a separate list, which in effect quantifies the benefits to be obtained from the specific hardening operation.

**Layout Algorithms:** The yFiles engine incorporates an impressive architecture for implementing layout algorithms on a given attack graph. Many of these algorithms are the end product of significant mathematical and computational research. Having a rich palette of alternate layouts to choose from greatly strengthens the analytic benefits of graph visualization, since viewing data with different layout schemes can often enable recognition of fundamental underlying patterns that might otherwise be invisible. *Incremental* layout algorithms are intended to optimize the results of small operations, such as opening a folder or dragging a node, while *from-scratch* or *global* layout algorithms generally produce radical transformations of the entire set into an entirely new view. We permit the invocation at any time of hierarchical, organic, circular, or orthogonal layout algorithms.

**Aggregation:** The ability to apply additional levels of aggregation to an existing display can be useful to an analyst wishing to study larger-scale behavior or simplify an existing region of the graph. We allow selection of multiple entities and aggregation into a single folder. This requires incremental layout to be performed, and edge aggregation quantities to be re-computed.

## 4 Visualization Features

Fig.4 shows the components of our attack graph visualization tool. The *main graph* view is the attack graph showing all possible (directed) paths through the network, in which the analyst may drilldown, perform what-if analysis, etc. In the scenario shown, a particular attack starting point (green) and ending point (red) are specified. Two protection domains are expanded to show their member hosts and the exploits among them. The *exploit table* displays the relevant exploits (as both attackers and victims) for the selected protection domain. Mouse hovering over an *exploit field* shows the full data for that field. The *overview pane* maintains the context of the overall graph. The *tree view* represents the entire attack graph in the form of a directory hierarchy. The *harden list* logs interactive what-if network hardening decisions, while the *defense* shows optimal network hardening recommendations automatically computed by CAULDRON.

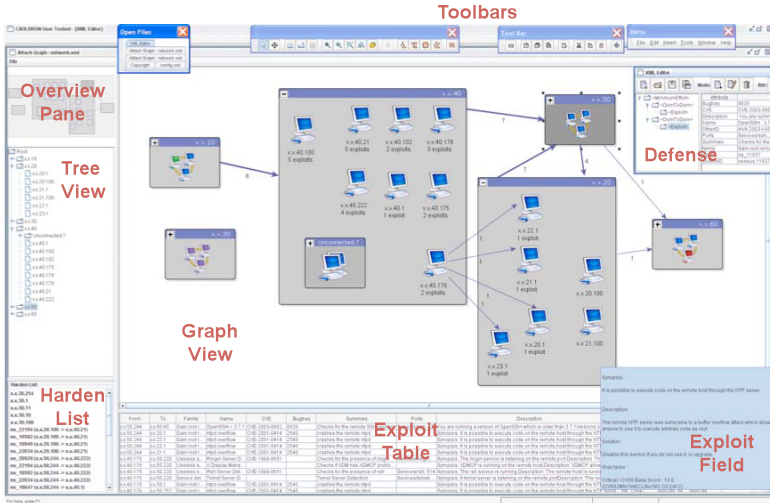


Fig. 4. Major components of the attack graph visualization tool

**Graph interactions:** The graph view supports a number of interactive edit mode features, including selection, deletion, relocation, and resizing of elements. Protection domains and other higher level nodes are opened and closed by clicking the +/- icon in the upper left corner. The graph can be zoomed to any magnification and positioned arbitrarily. The main graph view, tree view, and exploit table are all linked, so that user focus on any one component shifts focus on the others.

**Context Menu:** Context menu options are available by right-clicking an item or whitespace in the display, for network hardening simulation, deleting nodes, manipulating folders, etc. The context menu is also supports aggregating nodes into new folders. Another feature, called *traversal*, initiates an animated trace (in red) of all exploits originating or terminating in a selected node, providing focus on specific attack scenarios within a complex attack graph display.

**Toolbar Features:** A number of other useful features are implemented in the application toolbar, such as buttons for invoking layout algorithms, and the interactive functions of edit mode. The export function exports the graph view in \*.JPG, \*.GIF, or \*.SVG formats. The copy to clipboard transfers either the graph view or the entire graph to the clipboard. The magnifying glass zoom tool has arbitrary radius and power, and the cursor remains active at the center of the magnified view.

## 5 Related Work

Early work in automated generation of attack graphs involved explicit enumeration of attack states, which had serious scalability problems [5][6][7]. Under reasonable assumptions, complexity of attack graph generation was shown to be polynomial [8]. Attack graphs have also been generated efficiently through relational [9] and rule-based [10] approaches. Attack graph research has generally focused on efficiency,

rather than visualization methods. The approach in [11] visualizes single-step attacks and reachability only. Attack graph visualization capabilities in commercial tools remain limited [12][13]. Our work is unique in that it is the first practical application of the attack graph visual clustering framework proposed in [3].

## 6 Summary

This paper describes a graph-theoretic approach to the problem of network risk visualization. This provides powerful new capabilities for visual analysis of attack graphs, and is an integrated component of the CAULDRON tool developed at George Mason University. This approach is exemplary in that it leverages the comprehensive yFiles architecture, bringing flexible new visualization and analysis capabilities to the network security realm. This provides essential building blocks for analyzing, visualizing, editing, and drawing network attack graphs, opening the door to a wide range of new analytic capabilities.

## References

1. Jajodia, S., Noel, S.: Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response. Indian Statistical Institute Monograph Series. World Scientific Press, Singapore (2008)
2. yWorks – The Diagramming Company, <http://www.yworks.com/en/index.html>
3. Noel, S., Jajodia, S.: Managing Attack Graph Complexity through Visual Hierarchical Aggregation. In: Workshop on Visualization and Data Mining for Computer Security (2004)
4. Wang, L., Noel, S., Jajodia, S.: Minimum-Cost Network Hardening Using Attack Graphs. *Computer Communications* 29(18), 3812–3824 (2006)
5. Phillips, C., Swiler, L.: A Graph-Based System for Network-Vulnerability Analysis. In: New Security Paradigms Workshop (1998)
6. Ritchey, R., Ammann, P.: Using Model Checking to Analyze Network Vulnerabilities. In: IEEE Symposium on Security and Privacy (2000)
7. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated Generation and Analysis of Attack Graphs. In: Proceedings of the IEEE Symposium on Security and Privacy (2002)
8. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, Graph-Based Network Vulnerability Analysis. In: 9th ACM Conference on Computer and Communications Security (2002)
9. Wang, L., Yao, C., Singhal, A., Jajodia, S.: Interactive Analysis of Attack Graphs Using Relational Queries. In: Data and Applications Security XX (2006)
10. Ou, X., Boyer, W., McQueen, M.: A Scalable Approach to Attack Graph Generation. In: 13th ACM Conference on Computer and Communications Security (2006)
11. Williams, L., Lippmann, R., Ingols, K.: An Interactive Attack Graph Cascade and Reachability Display. In: Workshop on Visualization for Computer Security (2007)
12. Skybox Security, <http://www.skyboxsecurity.com/>
13. RedSeal Systems, <http://www.redseal.net/>

# Improving Attack Graph Visualization through Data Reduction and Attack Grouping<sup>\*</sup>

John Homer<sup>1</sup>, Ashok Varikuti<sup>1</sup>, Xinming Ou<sup>1</sup>, and Miles A. McQueen<sup>2</sup>

<sup>1</sup> Kansas State University, USA  
{jhomer, ashokv, xou}@ksu.edu

<sup>2</sup> Idaho National Laboratory, USA  
miles.mcqueen@inl.gov

**Abstract.** Various tools exist to analyze enterprise network systems and to produce attack graphs detailing how attackers might penetrate into the system. These attack graphs, however, are often complex and difficult to comprehend fully, and a human user may find it problematic to reach appropriate configuration decisions. This paper presents methodologies that can 1) automatically identify portions of an attack graph that do not help a user to understand the core security problems and so can be trimmed, and 2) automatically group similar attack steps as virtual nodes in a model of the network topology, to immediately increase the understandability of the data. We believe both methods are important steps toward improving visualization of attack graphs to make them more useful in configuration management for large enterprise networks. We implemented our methods using one of the existing attack-graph toolkits. Initial experimentation shows that the proposed approaches can 1) significantly reduce the complexity of attack graphs by trimming a large portion of the graph that is not needed for a user to understand the security problem, and 2) significantly increase the accessibility and understandability of the data presented in the attack graph by clearly showing, within a generated visualization of the network topology, the number and type of potential attacks to which each host is exposed.

**Keywords:** attack graph, attack graph visualization, dominator, graph clustering, network security analysis.

## 1 Introduction

Attack graphs have been developed to aid in identification and correction of misconfigurations in enterprise network systems, by providing a visual representation of potential attack paths [1,2,3,4,5,6,7,8]. Much work has already been done in the generation of attack graphs, producing more efficient techniques for building them [6,7]. Attack graphs, however, are difficult for a human to utilize effectively because of their complexity [9,10,11]. Even a network of moderate size can have dozens of possible attack paths, overwhelming a human user with the amount of information presented. It is not

---

<sup>\*</sup> This work is partially supported by the National Science Foundation under Grant No. 0716665, and U.S. Department of Energy. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the U.S. government agencies.

easy for a human to determine from the information in the attack graph which configuration settings should be changed to best address the identified security problems. Without a clear understanding of the existing security problems, it is difficult for a human user to evaluate possible configuration changes and to verify that optimal changes are made.

Previous works have introduced improvements in the visualization of attack paths and the overall presentation of attack graph data. Noel, *et al.* suggested that complexity can be reduced through the use of protection domains to represent groups of machines with unrestricted interconnectivity [9,10]. Lippmann, *et al.* introduced visualization approaches to emphasize critical attack steps while clearly showing host-to-host reachability [11].

In this paper, we show that by utilizing the logical semantics of an attack graph, one can 1) distinguish attack steps based on their usefulness for a human to quickly understand the core security problems in an enterprise network, and trim those that do not contribute much for this purpose; 2) identify attack steps that share similar semantics, and thus can be grouped and presented as a single virtual node. These techniques can further improve the visualization of attack graphs to make them more useful in practice.

For our implementation, we use the MulVAL attack graph tool suite [7,12], which provides reasonable performance and scalability for enterprise networks of a realistic size. MulVAL produces complete logical attack graphs, which are easily mapped back to a visualization of the network topology, based on the input data.

Figure 1 depicts an example enterprise network that is based on a real (and much bigger) system; we will return to this example throughout the paper. The network includes three subnets: a DMZ (Demilitarized Zone), an internal subnet, and an EMS (Energy Management System) subnet, which is a control-system network for power grids. In this

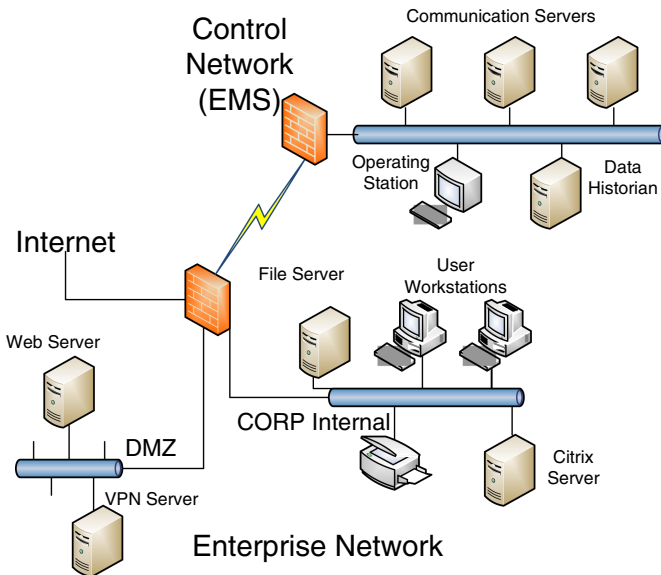
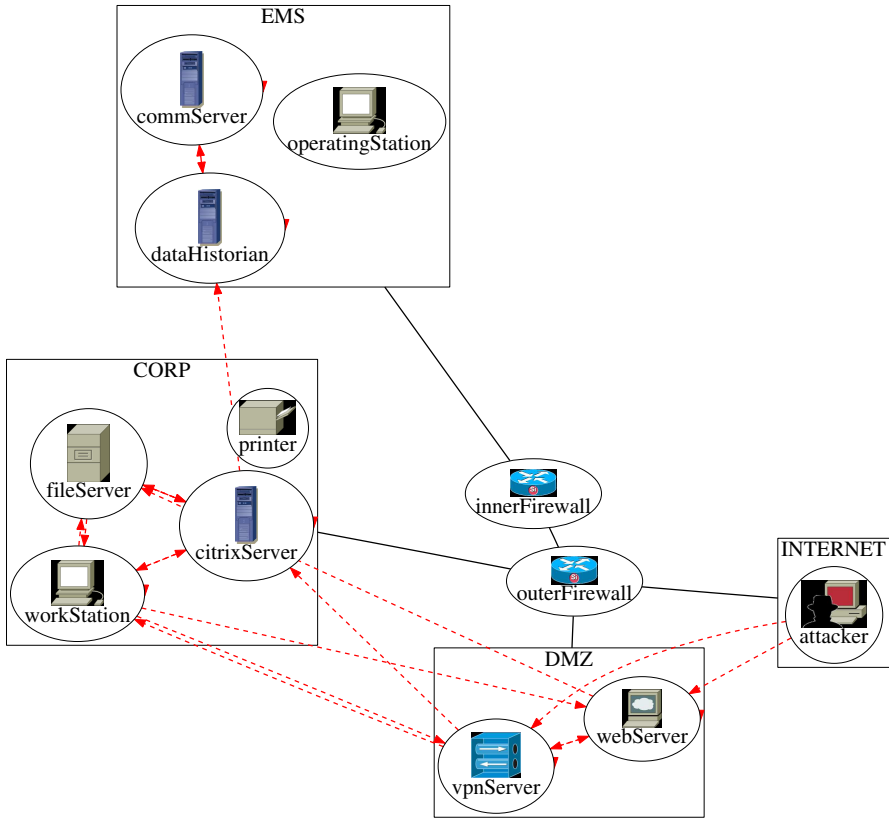


Fig. 1. An example enterprise network

example, we will assume that host-grouping has already been applied, based on similar configurations; the workstation node, for example, might be an abstracted grouping of one hundred workstation machines with comparable setups. Both the web server and the VPN server are directly accessible from the Internet. The web server can access the file server through the NFS file-sharing protocol; the VPN server is allowed access to all hosts in the internal subnet (but not the EMS subnet). Outside access to the EMS subnet is only allowed from the Citrix server in the internal subnet, and even then only to the data historian. In this example, we assume that the attacker's goal is to gain privileges to execute code on the commServer. From the commServer, an attacker could send commands to physical facilities such as power-generating turbines, which can cause grave damage to critical infrastructures.

A visualization of the MulVAL attack graph is shown in Figure 2, identifying a large number of potential attack paths in this network. This visualization is produced by mapping the full MulVAL logical attack graph to the network subnet topology, using GraphViz to construct the image, and applying clustering techniques similar to Noel *et al.*'s approach [10]. The black solid lines represent connectivity between subnets and gateways (router, firewall, *et al.*). Machines in the same subnet (represented as a rectangular cluster) have unrestricted access to each other. The red dotted lines indicate attack propagation paths as mapped from the full logical attack graph. This visualization omits a large amount of information from the original full attack graph, such as pre- and postconditions for each attack step. We believe a simple visualization of attack paths directly on the network topology will be useful in practice, since it relates the information conveyed by the attack graph to the concrete entities in an enterprise network, and a system administrator will likely find it easier to understand than the full attack graph. The full logical attack graph (see Appendix A) contains all of the same information shown in Figure 2, portraying all possible paths by which the commServer could be compromised; the breadth of this information, however, leads to a graph so large and complicated as to be unreadable by a human user.

Even after this simplification, the attack paths in the visualization may still overwhelm a user. In this paper we will focus on how to further reduce the complexity through two techniques. First, we observe that there are a number of logically valid attack steps that probably need not be shown to the user for him/her to understand the security problem. For example, an attacker who has gained privileges on the workstation machine in the internal subnet has opportunity to exploit the webServer in the DMZ. Though possible, this attack step is intuitively unhelpful to understanding the security problem: the attacker would have to compromise DMZ to gain privilege on workstation in the first place. Showing users the attack steps "back" to DMZ does not provide any additional insights. In reality, these attack steps are likely to be useless to the attacker as well. Another example would be attacking the fileServer from the citrixServer; Since we assume the commServer is the attacker's goal, an attacker with privileges on the citrixServer already has all of the necessary privileges to attack the EMS subnet through the dataHistorian. The user gains nothing useful by knowing that the attacker can further attack the fileServer from the Citrix server. Second, we observe that the complexity of the attack graph does not necessarily reflect complexity in security vulnerabilities. Employing a compromised user account, an attacker can access the citrixServer from the vpnServer,



**Fig. 2.** Attack graph visualization

workStation, and fileServer; using a Trojan horse attack, an attacker can gain access to the citrixServer from the fileServer. (The edge from fileServer to citrixServer represents both potential attack steps). Although there are four distinct attack steps that enable an attacker to compromise the citrixServer, these attack steps utilize only two distinct exploitations. This fact is obscured in the attack graph by the separate attack steps leading to citrixServer from different host machines.

We believe that the attack graph complexity can be further reduced. In order to make an attack graph a useful tool for configuration management, we identify as a research challenge the need for presenting the security problems expressed by an attack graph in a manner that enables a human user to more quickly grasp the core of the security problem. Our contributions are:

1. We developed an algorithm to identify portions of an attack graph that are not helpful for a user to understand the core security problems, and reduce the amount of data presented to the user by trimming those portions. When the amount of information presented in the attack graph is reduced, we believe that core security problems will be more quickly identifiable from the attack graph.

2. We developed a method to create virtual nodes to represent groupings of similar exploitations. In this approach, each attack step edge leading into a host represents a unique attack on that machine. We believe that this approach will increase the understandability of the attack graph data by more clearly displaying the exploitability of each host.

Our approach to trimming attack steps ensures that all distinct attack paths will be retained in the trimmed attack graph, while removing data not beneficial to the understanding of core security problems. Host-grouping techniques have already been shown to be effective for reducing complexity [6,9,10]. We show that further gains can be made in grouping similar exploits from multiple sources, which makes clearly visible the number of exploits available on a given machine and all of the possible sources for each potential exploit. It is easy then to see all attack steps that can be eliminated by resolving the vulnerability enabling a specific exploit.

The trimming algorithm is presented in Section 2. The exploit grouping approach is presented in Section 3. We will discuss related work in Section 4 and conclude with a discussion of future work on these approaches in Section 5.

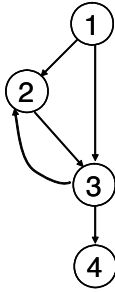
## 2 Identifying and Removing “Useless” Attack Steps

In examining the attack graph, we found that many of the attack steps, while valid from a logical point of view, are not helpful for a human user to comprehend the core security problems in the network configurations. They share a common characteristic which is they do not reveal the most important vulnerability in the system since the attacker does not penetrate “deeper” into the enterprise network along those steps. While these steps contain important information that would be useful if one wished to block every possible attack path, they can also be distracting to a human reader and often hides the root causes of the security problems. It is thus beneficial to remove these less useful attack steps from the attack graph so that the security problems become easier to grasp for a human reader.

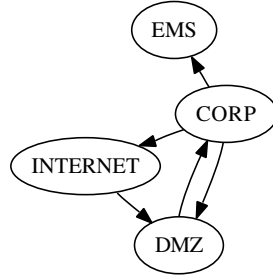
We refer to attack steps that are not useful for a human reader to understand the underlying security problems as “useless” attack steps. Generally, these “useless” attack steps involve an attack on a machine further from the goal machine than the machine from which the attack is made. In the example described earlier, for instance, one valid attack step enables an attacker with privileges on the workstation to gain privileges on the webServer, but this attack would not bring the attacker any closer to the presumed goal of accessing the commServer.

Our classification of “useful” and “useless” edges is meant to reflect a prioritization of the data contained in the attack graph. We can then provide a simplification of the original attack graph, highlighting attack reachability between hosts to enable a human user to more quickly comprehend fundamental vulnerabilities in the network. It is important to emphasize that the so-called “useless” attack steps are valid and important to consider when determining upon appropriate countermeasures. However, when the user is first presented with the attack graph, understanding these paths is not crucial for understanding overall security threats. It would be more beneficial if the user can quickly understand the core security problems from a simplified attack graph. For example, in Figure 3 the attacker’s starting machine is host 1, and his goal is to compromise host 4.





**Fig. 3.** Example useless attack steps



**Fig. 4.** Subnet graph, built from example in Section 4

There are two paths:  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$  and  $1 \rightarrow 3 \rightarrow 4$ . Intuitively the attack step from machine 3 to machine 2 is not useful, since it does not help the attacker to reach his goal. We would like to trim those steps.

The immediately obvious solution for identifying these “useless” attack steps is to implement a breadth-first search algorithm to compute the distance of machine from the goal machine, as measured by the minimum number of inter-host attack steps necessary to reach the goal from that machine. Machine 2 in the above example would have a distance of 2 and machine 3 would have a distance of 1. Attack steps that move from a machine to another machine with the same or greater distance (e.g.  $3 \rightarrow 2$ ) could then be labelled “useless” and trimmed. While this approach would work in some cases, in many cases useful attack paths would be trimmed. For example, this algorithm would also trim the step  $1 \rightarrow 2$ , and thus lose the complete attack path  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ . Actually only attack paths with the shortest length will be preserved and all other paths will be trimmed. This approach can lose valuable information, especially in cases when an attacker might follow a longer attack path due to ease of exploit or better stealthiness along the longer path.

Another seemingly correct solution is to perform a simple depth-first search and trim the back edges in the DFS tree. However, depending on the order of traversing a node’s multiple children, both edge  $2 \rightarrow 3$  and  $3 \rightarrow 2$  could be back edges in the DFS tree. So this method does not work either.

We have developed a two-level approach to identifying and removing “useless” attack steps. First, we create a directed graph with subnets as nodes and possible inter-subnet attack steps as edges. From this directed graph, we then construct a dominator tree to recognize dominance and post-dominance relationships between subnets with respect to presumed attacker location and his goal. The subnet where the attacker is located will be the source node and the subnet where the goal machine is located will be the sink node. Let  $d, n, p$  be vertices in a directed graph. Then  $d$  dominates  $n$  if every path from the source node to  $n$  must go through  $d$ . We write  $d \text{ dom } n$  for this fact. Also,  $p$  post-dominates  $n$  if every path from  $n$  to the sink node must go through  $p$ . We write  $p \text{ postdom } n$  for this fact. In the subnet graph of Figure 4, assuming that INTERNET is the source and EMS is the goal, some examples of dominance relationships are  $\text{DMZ dom CORP}$  and  $\text{CORP dom EMS}$ . We also have  $\text{EMS postdom CORP}$  and  $\text{CORP postdom DMZ}$ .

For any two subnets  $X$  and  $Y$ , we then identify as “useless” all inter-subnet attack steps  $X \rightarrow Y$  where  $Y \text{ dom } X$  or  $X \text{ postdom } Y$ . If  $Y \text{ dom } X$ , then an attacker who has gained privileges in subnet  $X$  must already have privileges in subnet  $Y$  (or the attacker would not have been able to transition to  $X$ ). If  $X \text{ postdom } Y$ , then moving from  $X$  to  $Y$  will not help the attacker either since he would have to return to  $X$  in order to reach his goal. Therefore, any transitions between two hosts in different subnets that fits one or both of these cases is “useless” and will be trimmed from the attack graph. They are distracting for a human reader trying to comprehend other, more enlightening attack paths. In the attack graph shown in Figure 2 every transition from the CORP subnet to the DMZ will be trimmed since DMZ dominates CORP. On the other hand, the transition from CORP to the control network subnet EMS will be retained.

After applying the inter-subnet transition trimming, we then address intra-subnet transitions. An attack step between two machines  $A \rightarrow B$  in the same subnet is retained in only two cases. First, if the subnet contains the goal machine, the transition is retained only if  $B$  is the goal. Any other transition within this subnet will be trimmed. In reality, an attacker might need to transition to other machines in the same subnet for the purpose of, *eg.* obtaining more computing power. However, these attack steps are not useful for a human to grasp the core security problem. Second, if the subnet does not contain the goal machine, the transition is useful only if  $B$  would provide an attacker with access to another subnet that would be deemed useful according to the subnet dominator tree, and even then only if that same access is not available from  $A$ . In the attack graph shown in Figure 2 the transition from fileServer to workstation is useless, since the workstation would not provide an attacker with new, useful access; however, the transition from fileServer to citrixServer is useful, since, from citrixServer, an attacker could access the EMS subnet.

Figure 5 shows the resulting graph after both levels of trimming levels are applied to the sample network. The attack graph now shows three key attack paths to reach the Citrix server, from which subnet EMS is accessible:

$$\begin{aligned} &Internet \rightarrow web\ server \rightarrow file\ server \rightarrow Citrix\ server \\ &Internet \rightarrow VPN\ server \rightarrow workstation \rightarrow Citrix\ server \\ &Internet \rightarrow VPN\ server \rightarrow Citrix\ server \end{aligned}$$

A careful reader might ask why the second attack path is retained, given the existence of a shorter path (3). As mentioned above, shorter attack paths cannot always subsume longer ones, since the exploits along the path may be different.

### 3 Abstraction of Attack Traces

Even after identifying and removing “useless” attack steps, many edges will likely remain in the visualization. Humans assessing the data presented in the attack graph will benefit from the reduced amount of data, but still face other obstacles to clear and straightforward understanding of the underlying security issues in the current network configuration. One hindrance to easy understanding of attack graph data can be the number of edges directed into a single host machine in the attack graph. In Figure 5 for example, the citrixServer node in the Corp subnet is the destination point of four attack

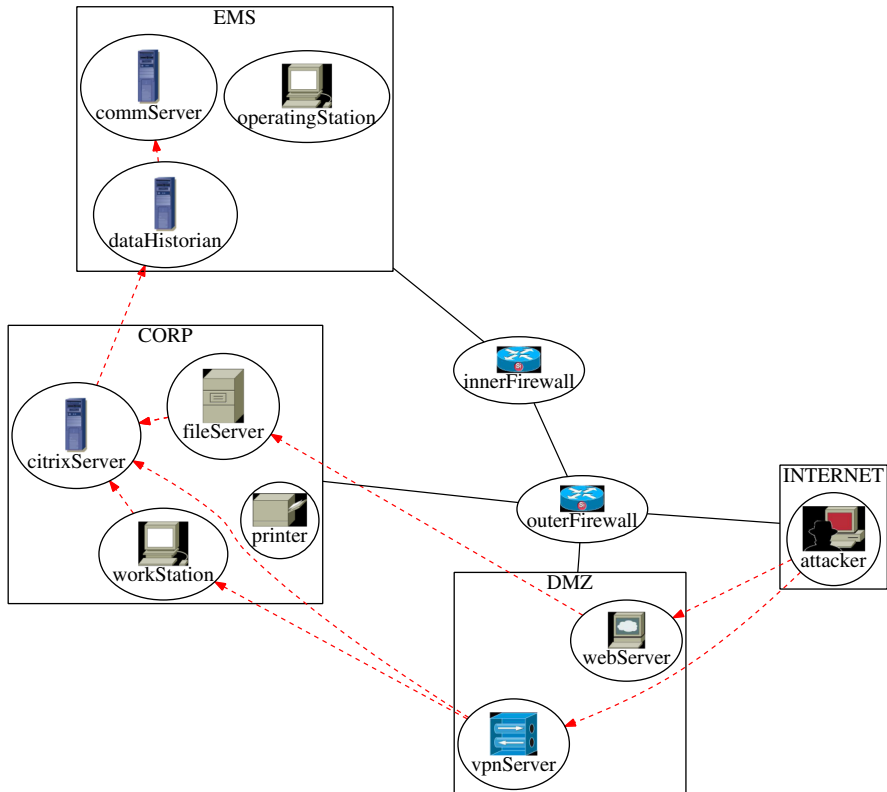
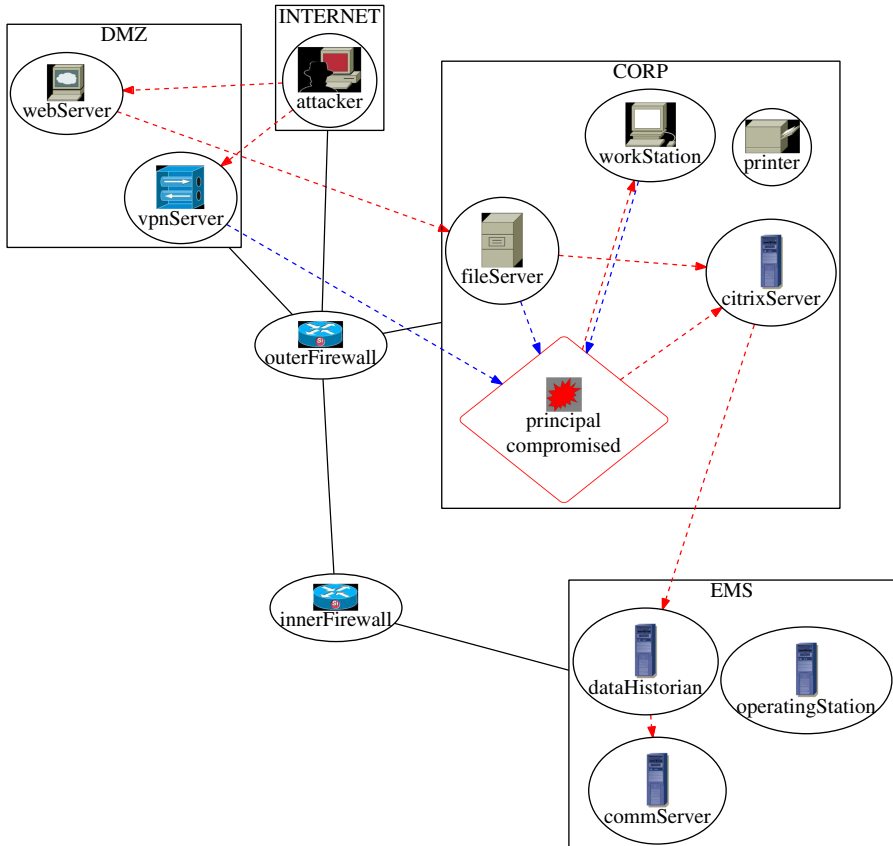


Fig. 5. Attack graph with both inter- and intra-subnet trimming applied

steps (shown in three edges), even after “useless” attack steps have been trimmed from the graph. It is not immediately clear to the user how many different possible exploitations are being represented, and how many sources for exploitations of the `citrixServer` are repetitions of a single attack type.

Our solution to this difficulty is to create an abstraction of each exploitation with multiple sources, from which only one edge will lead into the exploited node. In this way, it is much easier for a human user to see how many exploitations are possible on a given host and what potential attack steps could be eliminated by resolving the conditions enabling a specific exploitation. Potential exploits with only one source, on the other hand, will be represented by a direct attack step edge between the two machines, to maintain as much simplicity in the graph as possible.

In the attack graph shown in Figure 5 three of the edges that lead to the `citrixServer` represent different source points but only a single security issue in the network, namely the uncertain reliability of the user with account “`ordinaryUser`.” If this user account is compromised, an attacker could gain access to the `citrixServer` from any of the three host machines with edges leading to the abstracted exploitation node. By creating a virtual node to represent the existence of this security concern, it is much easier to see now that if the reliability of this user account can be verified, most of the possible



**Fig. 6.** Attack graph, trimmed, with virtual exploitation nodes

attacks leading to citrixServer will be eliminated. Our attack graph visualization will show transitions to an abstracted exploit node as blue lines, while red lines will indicate direct host-to-host attacks as well as attacks from an abstracted exploit node.

The full attack graph, shown in Figure 2, included a number of “useless” attack steps that are removed by the trimming algorithm. The trimming also reduces the number of multiple-source attacks and thus the number of abstract exploit nodes that can be created. For optimal effectiveness, this exploit abstraction technique is applied only to the trimmed attack graph. The final attack graph is shown in Figure 6.

## 4 Related Work

A number of other previous works addressed the problem of how to use attack graphs to better manage the security of enterprise networks [13, 14, 15, 16, 17]. The observations and insights from these previous works helped us develop the approach in this paper, and our work either complements or improves upon them. Our contribution is the

development of formal, logic-based approaches to simplifying an attack graph for a human to better understand.

Noel, *et al.* proposed a number of techniques for reducing complexity in attack graphs [10]. We adopted some of the approaches in our work, such as using clustering techniques to show the subnet topology of the network. Our approaches address complementary problems in visualization, namely identification and removal of attack paths that are not useful for a human to better understand the core security problems, and better represent attacks by grouping similar exploits targeted at a single host. Noel, *et al.* also presents a notion of graph trimming, by removing redundant exploits and allowing them to be implicitly conveyed in the graph. However, they do not systematically address how to identify and trim the “useless” attack steps described in this paper.

Lippmann, *et al.* have built on the multiple-prerequisite graphs produced by the NetSPA system with a goal of reducing attack graph complexity [11]. Their visualization employs spatial grouping and color-coding to represent levels of potential compromise. Groups of machines with similar levels of exploitability can then be collapsed, reducing the overall complexity of the graph. Our approach differs in that we do not group machines with similar vulnerabilities, but rather create abstract representations of attacks, with edges leading to the potentially affected machines.

## 5 Conclusion

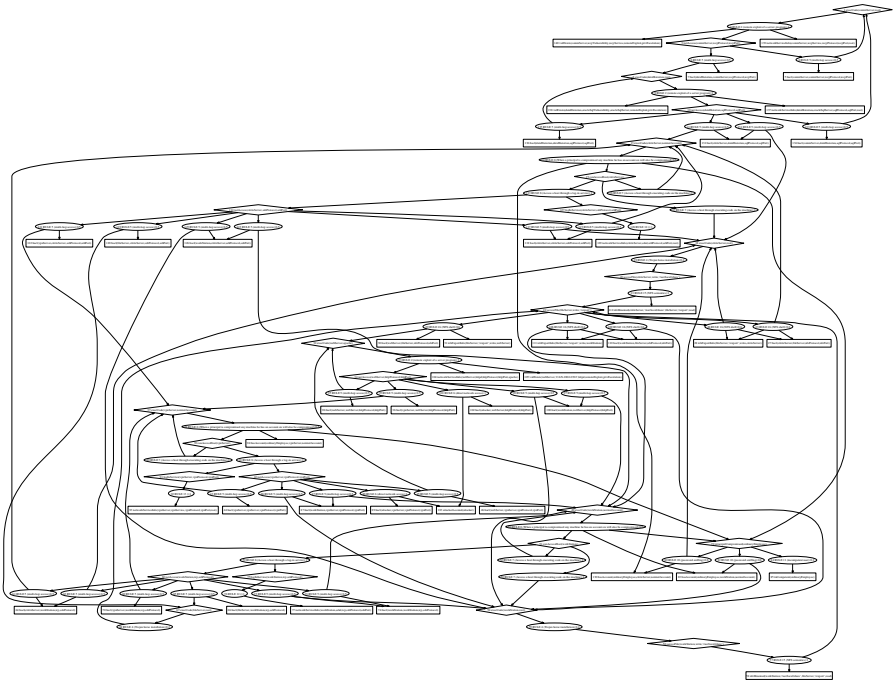
We have proposed two techniques for improving visualization of attack graphs — reducing the amount of data by identifying attack steps that are not crucial for a human to quickly understand the core security problems, and grouping similar attacks targeted at a single host to better represent the number and type of security problems. These techniques, in combination with visualization techniques developed by previous researchers, will display attack graphs to a more human-readable manner. This is crucial for using attack graphs to further automate enterprise network security management, since a human can only trust a tool if she/he understands its output. Our techniques will help a human user quickly identify the core security problems in an enterprise network without being overwhelmed by the amount of information contained in the full attack graphs.

## References

1. Swiler, L.P., Phillips, C., Ellis, D., Chakerian, S.: Computer-attack graph generation tool. In: DARPA Information Survivability Conference and Exposition (DISCEX II 2001), June 2001, vol. 2 (2001)
2. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 254–265 (2002)
3. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of 9th ACM Conference on Computer and Communications Security, Washington, DC (November 2002)
4. Jajodia, S., Noel, S., O’Berry, B.: Topological analysis of network attack vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (eds.) *Managing Cyber Threats: Issues, Approaches and Challenges*, ch. 5. Kluwer Academic Publishers, Dordrecht (2003)

5. Lippmann, R., Ingols, K.W.: An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory (March 2005)
6. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: 22nd Annual Computer Security Applications Conference (ACSAC), Miami Beach, Florida (December 2006)
7. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: 13th ACM Conference on Computer and Communications Security (CCS), pp. 336–345 (2006)
8. Li, W., Vaughn, R.B., Dandass, Y.S.: An approach to model network exploitations using exploitation graphs. *SIMULATION* 82(8), 523–541 (2006)
9. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: *VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pp. 109–118. ACM Press, New York (2004)
10. Noel, S., Jacobs, M., Kalapa, P., Jajodia, S.: Multiple coordinated views for network attack graphs. In: *IEEE Workshop on Visualization for Computer Security (VizSEC 2005)* (2005)
11. Williams, L., Lippmann, R., Ingols, K.: An interactive attack graph cascade and reachability display. In: *IEEE Workshop on Visualization for Computer Security (VizSEC 2007)* (2007)
12. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: A logic-based network security analyzer. In: 14th USENIX Security Symposium (2005)
13. Jha, S., Sheyner, O., Wing, J.M.: Two formal analyses of attack graphs. In: *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, Nova Scotia, Canada, June 2002, pp. 49–63 (2002)
14. Lippmann, R.P., Ingols, K.W., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., Cunningham, R.: Evaluating and strengthening enterprise network security using attack graphs. Technical Report ESC-TR-2005-064, MIT Lincoln Laboratory (October 2005)
15. Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., Cunningham, R.: Validating and restoring defense in depth using attack graphs. In: *Military Communications Conference (MILCOM)*, Washington, DC, U.S.A. (October 2006)
16. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: *Proceedings of Recent Advances in Intrusion Detection (RAID)* (September 2006)
17. Wang, L., Singhal, A., Jajodia, S.: Measuring network security using attack graphs. In: *Third Workshop on Quality of Protection (QoP)* (2007)

## A MulVAL Logical Attack Graph



**Fig. 7.** Full logical attack graph, as generated by MulVAL

# Show Me How You See: Lessons from Studying Computer Forensics Experts for Visualization

T.J. Jankun-Kelly<sup>1</sup>, Josh Franck<sup>2</sup>, David Wilson<sup>1</sup>, Jeffery Carver<sup>3</sup>,  
David Dampier<sup>1</sup>, and J. Edward Swan II<sup>1</sup>

<sup>1</sup> Department of Computer Science and Engineering, Mississippi State University  
{tjk,dw152,dampier,swan}@cse.msstate.edu

<sup>2</sup> Department of Psychology, Mississippi State University  
jaf210@msstate.edu

<sup>3</sup> Department of Computer Science, University of Alabama  
carver@cs.ua.edu

**Abstract.** As the first part of a Analyze-Visualize-Validate cycle, we have initiated a domain analysis of email computer forensics to determine where visualization may be beneficial. To this end, we worked with police detectives and other forensics professionals. However, the process of designing and executing such a study with real-world experts has been a non-trivial task. This paper presents our efforts in this area and the lessons learned as guidance for other practitioners.

## 1 Introduction

While violent crimes such as armed robbery and murder are decreasing in the U.S., computer crime is growing world-wide [1,2,3]. The growth of the Internet has contributed to an increase in cyber crimes such as child pornography, gambling, money laundering, financial scams, extortion, and sabotage [3,4,5]. Besides their using a computer in the commission of a crime, computer criminals share another similarity: The chances of their being caught and successfully prosecuted are relatively small [1]. In one example, a sheriff's department investigator working exclusively on computer crimes full-time for five years made only five arrests, none of which led to convictions [6]. Thus, tools to assist computer forensics practitioners are becoming increasingly important.

Several commercial and open-source tools exist to assist forensic detectives. While these systems automate some tasks and facilitate others, we believe there is room for visualization to be of assistance. Initial efforts have been made in this area [7]. However, the work process of computer forensic detectives and the interaction with their tools has not been thoroughly studied; we do not want to use visualization to solve problems which are irrelevant or done better by existing tools. We have therefore initiated a domain analysis of computer forensic personnel before creating any visualization. In this paper, we discuss the iterative process we went through when working with these experts, some preliminary results, and the lessons learned from this effort. We hope this contribution will serve as a case study for future efforts in security visualization in a similar vein to other efforts in visualization and software engineering [8,9].



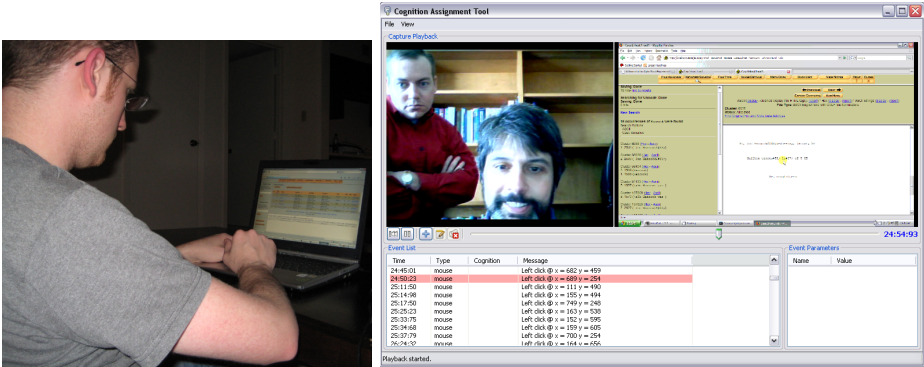
## 2 Case Study: Webmail Forensics Domain Analysis

Computer crime takes on many forms; a study of forensic analysis of all such crimes is beyond our scope. Thus, we focused on one aspect of computer forensics: Webmail and Internet history analysis in fraud cases. The ability to generate numerous web email accounts and the difficulty of putting together a coherent timeline of webmail usage motivated our investigation. In addition, finding corroborating evidence from other files on a hard drive (e.g., financial records) was of interest. Existing forensic tools such as the Forensics Toolkit [10] and the Autopsy Forensic Browser [11,12] provide interfaces to search files and unallocated sectors for relevant information, but do not necessarily provide guidance on what to search or how best to find evidence. The effectiveness of these tools is thus highly practitioner dependent. Therefore, we studied the workflow of such users at different levels of expertise to discover how visualization could be of benefit.

### 2.1 Study Protocol and Design

Our study's design evolved over several iterations as we interacted with our experts, although the eventual visualization goal was kept fixed. We initially considered a full verbal think-aloud protocol design [13,14]. Such analysis is one in which the recorded, verbalized thoughts of subjects are broken into the smallest information-bearing segments and coded categorically (e.g., Search, Navigate). These coded segments can then be examined for frequency of individual coding types, for recurring segments of coding types, and for tracking the evolution of the subjects' cognitions over the course of the task. Most importantly, by connecting the time stamp of coded segments to the specific subtasks being performed during that period, these same connections can be drawn on a subtask-by-subtask basis. Unfortunately, this process not only requires a great deal of time and effort to perform, but generally produces an almost toxic level of data. Also, it is the most complex and data-rich analysis method available to examine this particular task. As the exact relationship between the data gathered on the subject-side and the changes made on the visualization-side has yet to be established, this method was deemed excessive for an initial study.

For our expert trails, we settled on a more open-ended contextual task analysis utilizing a greater range of data sources. During a contextual task analysis, the goal is to observe users within their normal work context, as opposed to an artificial setting such as a laboratory [15,16,17]. For this purpose, a laptop with the forensics software (Autopsy), test cases, and observational software (screen/mouse capture and a web camera) was provided to subjects (Figure 1 left.). Subjects were instructed to "do what they do normally," and were provided (unobtrusively) with a notepad and pen with which to take notes, should they desire. There was also an embedded note-taking system within the analysis suite used. These two note-taking sources were intended as the primary source of data, and were to be analyzed in a fashion similar to the verbal protocols described above, though at a higher (and less cognitively complete) level. The screen/mouse capture and audio/video data provided by the laptop were used



**Fig. 1.** Experimental setup for our task analysis. *Left:* Experimental rig. A version of Autopsy [11] modified for capturing user events was the forensic software used. *Right:* Analysis tool we developed to process expert trials. Webcam input is in the upper-left, screen capture video in the upper right, mouse events on the lower left, and observer notes and analysis on the lower right.

to complement these notes. The screen/mouse capture data was intended to be examined for instances where activity indicated confusion, or where recurring patterns of input required an excessive amount of movement in terms of screen distance. The video and audio data were intended to be examined to inform the design of our subsequent visualizations; we use a custom developed analysis framework that syncs the video, screen capture, and mouse event display for this purpose (Figure 1 right).

While we wished to observe experts analyzing real cases, the legal and time concerns involved, especially for ongoing criminal investigations, forced us to create test datasets. Based upon our discussion with our in-house forensics instructor (who also assists officers in real cases), we created two webmail fraud cases that mimicked attributes of similar cases. We interjected fraud related emails into email streams collected from active mailing list traffic to which our test accounts were subscribed, and intermixed web-page views related to the fraud with visits to high traffic sites such as Google, Yahoo!, and Wikipedia. The rest of the hard drive images contained a standard Windows XP installation. Both our test datasets were used in the task analysis as described next.

## 2.2 Study Execution

Once the task analysis study was designed, we set about soliciting subjects. Initially, detectives taking courses at the Mississippi State Forensics Training Center were considered; however, most did not possess sufficient knowledge to provide any meaningful “expert” data. Consequently, we solicited known forensics detectives throughout the state of Mississippi and neighboring areas through email and phone calls. This recruitment process was lengthy, requiring on average 30 conversations per subject over a month’s time to establish a date for observation.

Over the Spring 2008 semester, five experts were recruited, three of which completed the study. For each subject, a similar process was followed: The intent of the study was described, the structure of the study (e.g., the purpose of the rig, the advise to take notes, etc.) was discussed, basic details of the cases were given, consent forms were provided, and then the study itself was performed. Of the five recruited, two subjects refused to sign the consent form and thus were excluded from the study; their reasons for refusal are discussed in Section 3. Each study was conducted at the officer's place of work, either a police station or a prison. Subjects were observed for up to two hours or until they felt that had made as much progress they could on the two cases. Officers were thanked for their time, and then the anonymized data was processed for initial analysis.

### 2.3 Study Post Mortem

Several observations from our subject data can be made. First, subjects did not make extensive use of the note taking capability of the forensics software or use the provided note paper regardless of expertise level. This frustrated our efforts to perform coding based upon these notes, leaving only the audio, video, and logging data. Second, the audio and video streams are quite noisy, possessing multiple interruptions of the task and non-task related questions by the experts. While useful information was mined from these streams, it was more labor intensive than initially planned. Data from logging proved more rich, and we have begun to code sequences of events using our analysis tool—i.e., identifying sequences of mouse clicks that correspond to search activity. Metrics such as click counts per subtask will be used to identify areas where the task performance could be improved and serve as candidates for visualization. Initial results indicate the searching for terms and their relationships across documents are possible areas of improvement; formal and rigorous results from this analysis are beyond the scope of this paper.

## 3 Lessons Learned

In addition to the preliminary results, we analyzed the process of performing the study. Given the length of time taken to elicit our three completed expert trials, we felt improvements could be made. Herein, we present the lessons we learned from this effort.

**Keep the Goal in Mind.** The goal of our study is to observe how visualization may improve forensics. At several stages of our design, this eventual goal changed the nature of the study. As discussed, we initially considered a more thorough and intensive verbal analysis protocol. While this would be appropriate in the context of cognitive science, where the the low-level details of how a subject thinks is vital, more lightweight methods are sufficient for our purposes. In addition, screen and mouse capture was added as some metrics for determining the complexity of a task cannot be measured without such logs. These metrics can then be compared to the same tasks performed using our eventual visualization solutions.

**Working with Experts is Time Consuming.** While user studies in visualization are generally time consuming [9], expert populations require significant additional effort. Student populations for university-based studies are quite large, especially where Psychology programs provide subject pools as part of their curriculum. Experts, however, have to contend with their normal work assignments, which prolongs the process of recruitment and observation. Our forensic experts perform other tasks in addition to their forensic duties, complicating matters. Persistent effort (over 50 emails and phone calls were required for one subject, with 30 on average) is required. These factors confound the recruitment of experts, as demonstrated by the recruitment of software professionals for case studies [18]. We estimate that it took us two to three times longer to perform our initial study than it would have if we used only local, non-expert subjects.

**Go to the Experts.** Though we had significant difficulty establishing contact with subjects, going to the experts provided valuable. First, it strengthens the relationship with the expert as it shows our willingness to work with them. Second, observations about the expert's work environment (such as the distractions during the study) are pertinent to understanding the user. These observations can then be used when creating profiles of the experts for later work, such as using created personas for initial evaluation [19].

**Clearly Communicate Expectations.** While this is good advice for any study, it is doubly important for experts. Our experts had no experience with human subject studies, so the goals and procedures were unclear to each. Part of the reason recruitment was protracted was due to anxiety over the nature of the activity. One participant was concerned that the work would be used as part of their job performance evaluation (a false impression); another was unfamiliar with webmail cases, having dealt primarily with child pornography. With our later subjects, we were more clear with our expectations, and, as a result, the study went smoother.

**Provide Consent Forms Early.** Though this ties in with the previous lesson, it deserves special mention. Though we communicated to all our potential subjects that they would be recorded and their interactions with the software logged, two of our recruited experts declined to participate when the consent forms for the study were presented—this was after a 100 mile drive to meet them. As a consent form is a binding agreement between the investigators and the experts, care must be taken in explaining the factors involved. In the case of the withdrawn experts, the consent form was rejected due to concerns about the study's data being subpoenaed at a later date as evidence of the expert's potential lack of proficiency; as required by Mississippi State Institutional Review Board policy (and stated on the consent form), this data would have to be turned over in such a circumstance. If we had provided the consent form during our initial contact with the experts, this issue would have been discovered sooner and other measures taken.

**Be Prepared to Develop Your own Tools.** During the design phase of our task study, we searched for software to assist in coding and analyzing the coded results. Our results were disappointing, and we found no off-the-shelf software

that would fit our needs. After queries to our empirical software engineering and cognitive science colleagues, we concluded that most such studies were typically conducted via spreadsheet software and labor intensive manual collating and coding. For our more lightweight approach (video, audio, and logging), we did not find any tools that made the analysis straightforward either. Thus, we ended up creating our own software for coordinating the video, audio, and logging events and for aggregating said events (Figure 11 right). In addition, we used several open source programs to assist in gathering the data in the first place, though some commercial software exists for this purpose.

**But use the Tools the Experts Use.** An early decision of our group was to use the open source Autopsy software as our computer forensics platform; being open source, we could modify it to gather the logging data we required more readily than proprietary software. However, this proved to be a significant stumbling block with our experts, as they were in large familiar with the Forensic Toolkit (FTK). The lack of comfort with Autopsy uniformly caused extra training to be required before the experts could perform the study. For further iterations of our study, we plan to use FTK and have already instrumented it for this purpose.

## 4 Summary

Working with experts is required when an accurate understanding of their work practices is needed. We performed such a domain analysis to determine where visualization may benefit computer forensic practitioners. This study faced several unexpected hurdles which we have described as guidelines for visualization researchers interested in doing similar studies.

Though there were significant difficulties, working with experts was worth the effort. Our data has provided us with some initial avenues to pursue for visualization, and, more importantly, given us a better picture of how computer forensics is actually performed. We are currently redesigning our study to incorporate the atomic tasks we have identified such that novice users (i.e., university students) can perform them; this new design is informed by our interaction with our experts. Finally, our pool of experts will be utilized to validate our visualization designs when they are complete. Such validation would prove more difficult without the groundwork of our initial study.

## Acknowledgments

We gratefully acknowledge the support of the numerous law enforcement detectives which either participated in the study, worked with us on its design, or answered questions from us. We also thank Kendall Blaylock and Gary Cantrell of the MSSState Forensics Training Center for their assistance in recruiting subjects. The work is funded by a National Science Foundation CyberTrust grant #CNS-0627407.

## References

1. Householder, A., Houle, K., Dougherty, C.: Computer attack trends challenge internet security. *IEEE Computer* 35(4), 5–7 (2002)
2. Noblett, M., Pollit, M., Presley, L.: Recovering and examining computer forensic evidence. *Forensic Science Communications* 2(4) (2000)
3. Wolfe, H.: Computer forensics. *Computers and Security* 22(1), 26–28 (2003)
4. Bequai, A.: Syndicated crime and international terrorism. *Computers and Security* 21(4), 333–337 (2002)
5. Kessler, G., Schirling, M.: Computer forensics: Cracking the books, cracking the case. *Information Security*, 68–81 (2002)
6. Thompson, R.: Chasing after 'petty' computer crime. *IEEE Potentials* 18(1), 20–22 (1999)
7. Teelink, S., Erbacher, R.F.: Improving the computer forensic analysis process through visualization. *Communications of the ACM* 49(2), 71–75 (2006)
8. Host, M., Runeson, P.: Checklists for software engineering case study research. In: *International Symposium on Empirical Software Engineering and Measurement*, pp. 479–481 (2007)
9. Kosara, R., Healey, C.G., Interrante, V., Laidlaw, D.H., Ware, C.: User studies: Why, how, and when? *IEEE Computer Graphics and Applications* 23(4), 20–25 (2003)
10. AccessData: (Forensic toolkit 2.0) (Last checked May 2008), <http://www.accessdata.com/Products/ftk2test.aspx>
11. Carrier, B.: (Autopsy forensic browser) (Last checked May 2008), <http://www.sleuthkit.org/autopsy/>
12. Carrier, B.: *Computer Forensics Basics*. In: *Know Your Enemy*, ch. 11, 2nd edn., Addison Wesley, Reading (2004)
13. Singer, J., Lethbridge, T.: Methods for studying maintenance activities. In: *Proceedings of the Workshop for Empirical Studies of Software Maintenance*, pp. 105–110 (1996)
14. VanSomeren, M.W., Bernard, Y.F., Sandberg, J.A.C.: *The Think Aloud Method: A Practical Guide to Modeling Cognitive Processes*. Academic Press, London (1994)
15. Hackos, J.T., Redish, J.C.: *User and Task Analysis for Interface Design*. John Wiley & Sons, Inc., New York (1998)
16. Hix, D., Hartson, H.R.: *Developing User Interfaces: Ensuring Usability through Product & Process*. John Wiley & Sons, Inc., New York (1993)
17. Mayhew, D.: *The Usability Engineering Lifecycle: a Practitioner's Handbook for User Interface Design*. Morgan Kaufmann Publishers, San Francisco (1999)
18. Sjoberg, D.I.K., Anda, B., Arishold, E., Dyba, T., Jorgensen, M., Karahasanovic, A., Koren, E.F., Vokac, M.: Conducting realistic experiments in software engineering. In: *First International Symposium on Empirical Software Engineering*, pp. 17–26 (2002)
19. Stoll, J., McColgin, D., Gregory, M., Crow, V., Edwards, W.K.: Exploiting the user: Adapting personas for use in security visualization design. In: *Proceedings of the Fourth Workshop on Visualization for Computer Security* (2007)

# A Task Centered Framework for Computer Security Data Visualization

Xiaoyuan Suo, Ying Zhu, and Scott Owen

Department of Computer Science  
Georgia State University

xsuo@student.gsu.edu, yzhu@cs.gsu.edu, sowen@gsu.edu

**Abstract.** Most of the existing computer security visualization programs are data centered. However, some studies have shown that task centered visualization is perhaps more effective. To test this hypothesis, we have developed a new framework of visualization and apply it to computer security visualization. This framework provides a new way for users to interact with data set and potentially will provide new insights into how visualization can be better constructed to serve users' specific tasks.

**Keywords:** visualization, task, computer security.

## 1 Introduction

A fundamental question for visualization design is what makes visualizations effective? There have been different answers to this question. Some researchers take a more data-centric view and suggest that effectiveness depends on the accurate interpretation of presented data [1-3], or a matching between data structure and visual structure [4, 5]. However, a number of psychological studies have also shown that the effectiveness of visualization is task specific [6, 7].

In this paper, we propose a new task centered framework of visualization and apply it to computer security visualization. In our framework, a visualization system is optimized for specific tasks by mapping the task related parameters to the visual elements that have high accuracy, utility, and efficiency ratings.

Before visualizations are created, users specify tasks and their associated parameters. This process is essentially a task complexity analysis. Knowing the data parameters associated with a task helps users consciously control the complexity of the tasks and correlate task complexity and visualization complexity. This new framework provides a different way for users to interact with data set and potentially will provide new insights into how visualization can be better constructed to serve users' specific tasks.

## 2 Related Work

Many visualization designs have been proposed for computer security analysis. Noted examples include TNV [8], IDS RainStorm [9], PortVis [10], etc. Most of these designs, however, are prefabricated visualizations that cannot be easily reconfigured by

users for different tasks. An implicit assumption is that users can use interaction techniques to customize data visualization for different tasks. While interaction is essential for making visualization usable, two important issues need to be addressed. First, for most existing visualization systems it is often not clear what specific tasks they are designed for. As a result, users may use the visualization for unintended tasks. Second, most existing visualization systems provide only low level interaction techniques, such as zooming, panning, that are restrained by the predefined visualization structure. They may be suitable for problem solving process with relatively stable procedure and task structure. However, many complex problem solving process are not so well defined. In many cases, problem solving is a process of searching in the solution space. This means that users may constantly testing different hypotheses and apply different strategies. The task structure may keep changing during the problem solving process. A new set of higher level interaction techniques are needed to support this dynamic problem solving method.

Some visualization systems, such as RUMINT [11], do provide a more configurable interface that allow users to assign parameters to different coordinate axes, or choose different types of diagrams. Outside the field of computer security visualization, Tableau Software is noted for its highly flexible and configurable interface that allows users to quickly construct different data visualizations. Another example is Many Eyes [12], a web site that allows different users to construct different visualizations of the same data set. Although these are powerful user interface techniques, users are still operating at the visualization level. But most end users would prefer to operate at a higher level of thinking – ask questions, test hypotheses, etc. For end users, constructing and configuring visualization is a secondary activity to their primary tasks. Again, we need a higher level interaction technique to help end users operate at the level of tasks.

The research presented in this paper is an attempt to address this issue. The central component of the proposed visualization framework is a task tree that is dynamically linked to data visualizations and data tables. Users operate by constructing and maintaining a task tree. A frame of data visualization is created automatically (or semi-automatically) for each task on the task tree, with the support of a visualization engine.

### 3 Overview of Task Centered Visualization Framework

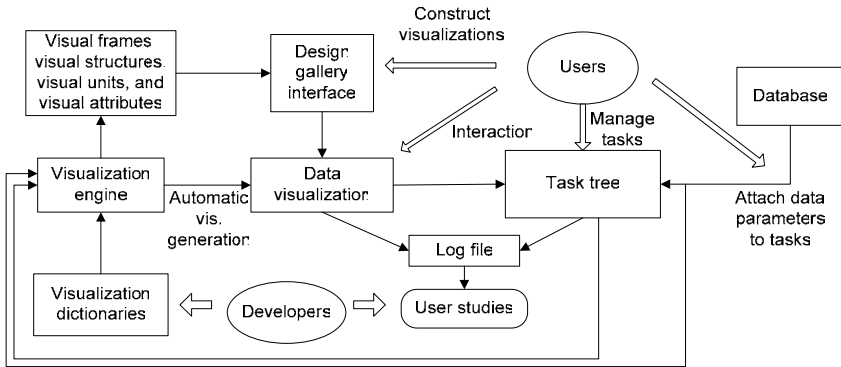
We propose a Task-centered Visualization Design Architecture (TVDA). Figure 1 shows the main components of TVDA.

To construct a visualization, users start with a visual frame and then drag and drop visual structures into the view. They are assisted by a design-gallery style interface [13, 14] that contains multiple visual structures provided by a visualization engine.

For domain experts, a typical visual problem solving process takes the following steps:

- Open the data files or connect to the databases.
- Divide the work into multiple tasks. Create a hierarchical task tree.
- Associate data parameters with each task.





**Fig. 1.** Overview of the Task-centered Visualization Architecture

- For each task, construct a data-visualization. A visualization engine will automatically recommend multiple design choices, which are presented in a design-gallery style interface. The designs are selected and ranked based on their accuracy, utility, and efficiency scores in the visualization dictionaries.
- Explore the data visualization through interaction techniques.

Tree is an appropriate data structure for organizing and storing problem solving activities [15-17]. Each node on the task tree represents a specific task. For each task, users shall explicitly identify the type of this task, based on the task classification conducted in step 3.

For each task, users are required to explicitly identify the data parameters that are needed to perform the task. More specifically, these are the parameters that have to be kept in the working memory simultaneously in order to carry out the task. Based on the Relational Complexity Theory [18, 19], the complexity of the task is determined by the number of these parameters. The proposed visualization tool allows users to open data files, select parameters, and attach them to a task.

A task tree also has other benefits. First, the task tree itself can be seen as a visualization of the problem solving process, reducing the cognitive load by externalizing the task structure that would otherwise be stored in the working memory. Second, a task tree is essentially a visual language for describing a specific problem solving strategy and expertise [20, 21], which can be shared or reused.

User controlled visualization construction is necessary for several reasons. First, complex problem solving is a dynamic process. In search for a solution, users need to test different hypotheses or different strategies. This means the task structure may be constantly changing, and a good visualization tool should allow users to dynamically reorganize visualizations to accommodate this change – because the effectiveness of visualization is task specific. Second, studies have shown that the effectiveness of visualizations depends on users' background and knowledge. Visualization is also a learned skill – as users become more experienced, their behavior for reading and constructing visualization may change [22]. Prefabricated visualizations combined with low level interactions – such as zooming, panning, and level-of-detail – are insufficient to address the individual differences. Third, self-constructed visualizations may

alert	classification	date	time	source	dest.	type
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	13:35:41	0.3.6C...	0.11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	13:37:16	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	13:37:16	0.F.3D...	0.4...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	13:45:41	0.3.6C...	0.11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	13:47:16	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	13:47:16	0.F.3D...	0.4...	0x800
SNMP public access udp	Attempted Information Leak	09/19	13:57:16	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	13:57:16	0.F.3D...	0.4...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:07:16	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:07:16	0.F.3D...	0.4...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:17:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:17:22	0.F.3D...	0.4...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	14:25:41	0.3.6C...	0.11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:27:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:27:22	0.F.3D...	0.4...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	14:35:41	0.3.6C...	0.11...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:37:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:37:22	0.F.3D...	0.4...	0x800
ICMP PING CyberKit 2.2 Windows	Misc activity	09/19	14:45:41	0.3.6C...	0.6...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:47:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:47:22	0.F.3D...	0.4...	0x800
SCAN UPnP service discover att.	Detection of a Network Sc...	09/19	14:50:29	0.2.3F...	1.0...	0x800
SCAN UPnP service discover att.	Detection of a Network Sc...	09/19	14:50:32	0.2.3F...	1.0...	0x800
SCAN UPnP service discover att.	Detection of a Network Sc...	09/19	14:50:35	0.2.3F...	1.0...	0x800
SNMP public access udp	Attempted Information Leak	09/19	14:57:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	14:57:22	0.F.3D...	0.4...	0x800
SNMP public access udp	Attempted Information Leak	09/19	15:17:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	15:17:22	0.F.3D...	0.4...	0x800
SNMP public access udp	Attempted Information Leak	09/19	15:27:22	0.F.3D...	0.4...	0x800
SNMP request udp	Attempted Information Leak	09/19	15:27:22	0.F.3D...	0.4...	0x800

Fig. 2. Data table

assist problem solving in ways different from prefabricated visualizations [6, 23]. Over time, these benefits will outweigh the initial learning curve.

In addition to the traditional interaction techniques (e.g. zooming, panning, level-of-detail [24]), the TVDA also allows users to be able to merge two or more visual frames. This technique is designed to reduce the cognitive load of visual integration and inference by externalizing the mental transformations [25-28].

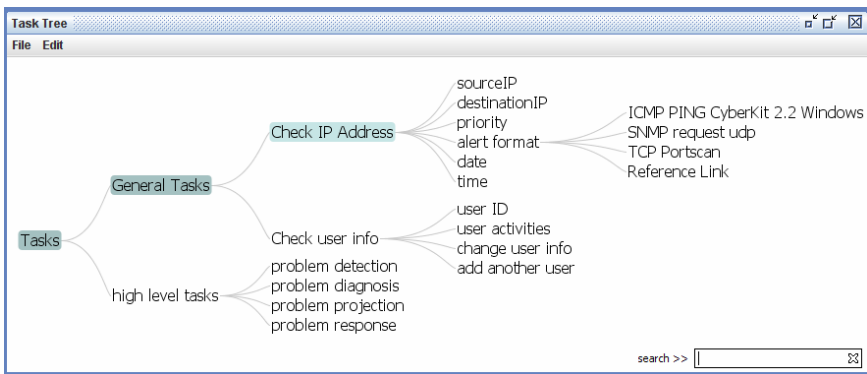


Fig. 3(a). Task tree

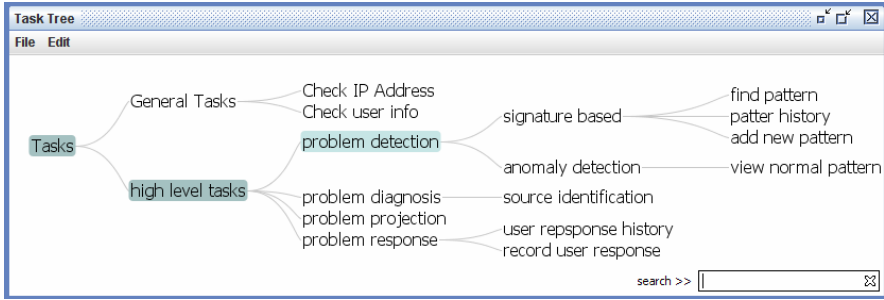


Fig. 3(b). High level task list

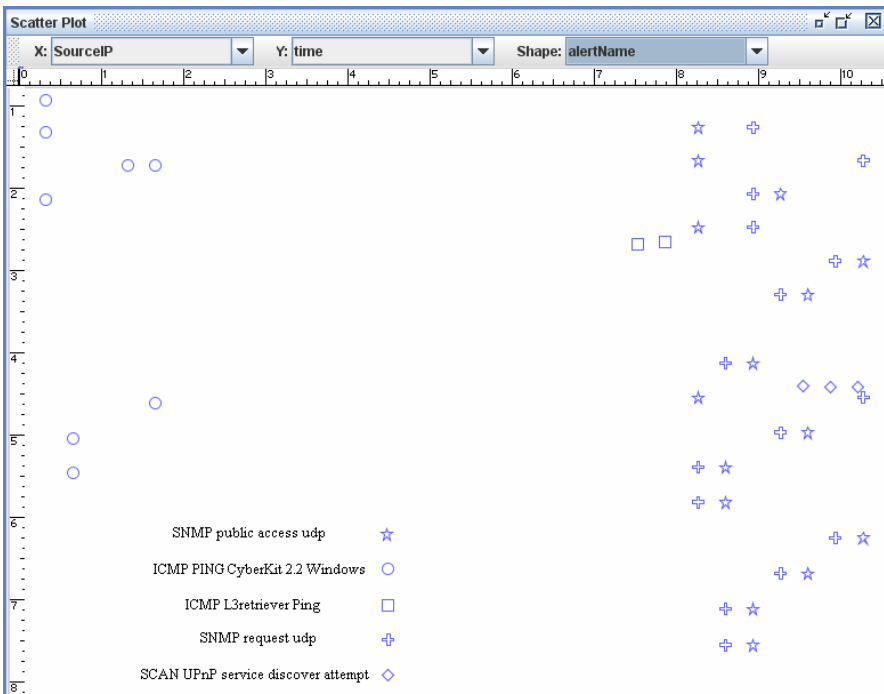


Fig. 4. Scatter plot. Different shapes represent different alert types. User can choose different parameters (Source IP, time, date, priority, alert name) for the X and Y axes.

## 4 Implementation

We have built a prototype based on the proposed framework. This prototype is implemented with Java and uses the *prefuse* [29] – an open source interactive information visualization library.

Current system has three main windows: data table, task tree, and visualization.

1. Data table (Figure 2):  
When the data is loaded, the system will process the raw data (Snort etc.), extract the necessary information, and then display the relevant data in a table. The data table is intended for security experts to exam raw data and also to drag and drop parameters into the task tree.
2. Task tree (Figure 3):  
Tasks will be displayed in a tree format, in which each node represents a task. A task can be divided into sub-tasks. Each task is associated with a number of parameters, as explained earlier. Low level tasks are also available using menu bars on top of the main visualization (zooming, panning, etc.).
3. Visualization (Figure 4):  
When a task is selected in the task tree, the desired visual interface is automatically generated for that task. In this visualization, each different shape represents a different parameter. Figure 4 shows the visualization generated for the task “Check IP Address” and “problem detection”. The different shapes on scatter plot represented different information extracted during a short period of time.

## 5 Conclusion and Future Work

We propose a task centered visualization design framework, in which tasks are explicitly identified and organized and visualizations are constructed for specific tasks and their related data parameters. The center piece of this framework is a task tree which dynamically links the raw data with automatically generated visualization. The task tree serves as a high level interaction technique that allows users to conduct problem solving naturally at the task level, while still giving end users flexible control over the visualization construction.

Much work needs to be done to realize the full potential of the proposed framework. Our future work includes developing a design gallery style visualization interface that allows users to compare and select from multiple visualizations that are automatically generated. A significant challenge is to develop a visualization engine that helps automatically generate visualizations given a task and its related parameters. The key is to codify the many design rules from the visualization research literature and to develop a systematic method to evaluate and optimize the visualization. Our previous work on visualization complexity analysis [30] can be used as the basis for the evaluation and optimization. Finally, we will develop an evaluation plan to test the effectiveness of the proposed framework, working with domain experts in the field of computer security.

## References

- [1] Mackinlay, J.: Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics* 5, 110–141 (1986)
- [2] Cleveland, W.S., McGill, R.: Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association* 79, 531–554 (1984)

- [3] Cleveland, W.S., McGill, R.: Graphical Perception and Graphical Methods for Analyzing Scientific Data. *Science* 229, 828–833 (1985)
- [4] Dastani, M.: The Role of Visual Perception in DataVisualization. *Journal of Visual Languages and Computing* 13, 601–622 (2002)
- [5] Wattenberg, M., Fisher, D.: Analyzing perceptual organization in information graphics. *Information Visualization* 3, 123–133 (2004)
- [6] Cox, R.: Representation construction, externalised cognition and individual differences. *Learning and Instruction* 9, 343–363 (1999)
- [7] Freedman, E.G., Shah, P.: Toward a Model of Knowledge-Based Graph Comprehension. In: Hegarty, M., Meyer, B., Narayanan, N.H. (eds.) *Diagrams 2002*. LNCS (LNAI), vol. 2317, pp. 59–141. Springer, Heidelberg (2002)
- [8] Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Preserving the Big Picture: Visual Network Traffic Analysis with TNV. In: *Workshop on Visualization for Computer Security*, Minneapolis, MN, USA, pp. 47–54 (2005)
- [9] Abdullah, K., Lee, C., Conti, G., Copeland, J.A., Stasko, J.: IDS RainStorm: Visualizing IDS Alarms. In: *IEEE Symposium on Information Visualization's Workshop on Visualization for Computer Security (VizSEC)* (2005)
- [10] McPherson, J., Ma, K.-L., Krystosk, P., Bartoletti, T., Christensen, M.: PortVis: a tool for port-based detection of security events. In: *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, ACM Press, Washington (2004)
- [11] Conti, G.: *Security Data Visualization: Graphical Techniques for Network Analysis*. No Starch Press (2007)
- [12] Viegas, F.B., Wattenberg, M., Ham, F.v., Kriss, J., McKeon, M.: Many Eyes: A Site for Visualization at Internet Scale. In: *Proceedings of the IEEE Symposium on Information Visualization* (2007)
- [13] Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T.: Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In: *Proceedings of ACM SIGGRAPH Conference* (1997)
- [14] Terry, M.: *Set-Based User Interface*, in PhD Thesis, School of Computing, Georgia Institute of Technology, Atlanta, Georgia (2005)
- [15] Bratko, I.: *PROLOG Programming for Artificial Intelligence*, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1990)
- [16] Pain, H., Bundy, A.: What stories should we tell novice PROLOG programmers? In: *Artificial intelligence programming environments*, pp. 119–130. John Wiley & Sons, New York (1987)
- [17] Simmons, R., Apfelbaum, D.: A Task Description Language for Robot Control. In: *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada (1998)
- [18] Halford, G.S., Baker, R., McCredden, J.E., Bain, J.D.: How Many Variables Can Humans Process? *Psychological Science* 16, 70–76 (2005)
- [19] Halford, G.S., Wilson, W.H., Phillips, S.: Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Behavioral and Brain Sciences* 21, 803–865 (1998)
- [20] Casner, S., Bonar, J.: Using the expert's diagram as a specification of expertise. In: *Proceedings of IEEE Symposium on Visual Languages* (1988)
- [21] Davies, J., Goel, A.K.: Transfer of problem-solving strategy using Covlan. *Journal of Visual Languages and Computing* 18, 149–164 (2007)

- [22] Petre, M., Green, T.R.G.: Learning to Read Graphics: Some Evidence that 'Seeing' an Information Display is an Acquired Skill. *Journal of Visual Languages and Computing* 4, 55–70 (1993)
- [23] Cox, R., Brna, P.: Supporting the use of external representation in problem solving: the need for flexible learning environments. *Journal of Artificial Intelligence in Education* 6, 239–302 (1995)
- [24] Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: *Proceedings of the IEEE Conference on Visual Languages*. IEEE, Los Alamitos (1996)
- [25] Ratwani, R.M., Trafton, J.G.: Making Graphical Inferences: A Hierarchical Framework. In: *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci)* (2004)
- [26] Ratwani, R.M., Trafton, J.G., Boehm-Davis, D.A.: Thinking Graphically: Extracting Local and Global Information. In: *Proceedings of the Annual Meeting of Cognitive Science Society* (2003)
- [27] Trafton, J.G., Kirschenbaum, S.S., Tsui, T.L., Miyamoto, R.T., Ballas, J.A., Raymond, P.D.: Turning pictures into numbers: extracting and generating information from complex visualizations. *International Journal of Human-Computer Studies* 53, 827–850 (2000)
- [28] Trafton, J.G., Trickett, S.B.: A New Model of Graph and Visualization Usage. In: *Proceedings of the Annual Meeting of Cognitive Science Society* (2001)
- [29] Heer, J., Card, S.K., Landay, J.A.: Prefuse: A Toolkit for Interactive Information Visualization. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)* (2005)
- [30] Suo, X., Zhu, Y., Owen, G.S.: Measuring the Complexity of Visualization Design. In: *Proceedings of the 2007 Workshop on Visualization for Computer Security (VizSEC)* (2007)

# BGPeeP: An IP-Space Centered View for Internet Routing Data

James Shearer<sup>1</sup>, Kwan-Liu Ma<sup>1</sup>, and Toby Kohlenberg<sup>2</sup>

<sup>1</sup> Visualization and Interaction Design Innovation lab  
University of California, Davis, CA 95616  
{jjshearer,klma}@ucdavis.edu  
<http://vidi.cs.ucdavis.edu>

<sup>2</sup> Intel Corporation  
toby.kohlenberg@intel.com

**Abstract.** We present BGPeeP, a tool for visualizing Border Gateway Protocol traffic at a detailed level, using a novel depiction of IP-space. This new visualization renders the network prefixes involved with such traffic using a method that leverages the peculiarities of BGP traffic to gain insight and highlight potential router misconfigurations. BGPeeP utilizes a simple interface and several methods of interaction to allow users to quickly focus on the data of interest. Our tool highlights aspects of BGP data which have received less attention in previous visualization applications, in order to help form a more complete picture of this vital part of the Internet communications infrastructure.

## 1 Introduction

The Border Gateway Protocol (BGP) is the top-level routing protocol currently utilized to maintain a constantly connected topology for the global Internet. Every day, many thousands of border routers under the ownership of Autonomous Systems (ASes) constantly converse, exchanging information about their own IP-space ownership, distant network reach-ability, and broken peer connections. This ongoing router “chatter” generates a huge amount of multi-variate data, and at any given time governs the current state of the amorphous, global routing table. This ephemeral data exists exclusively in the rarely-glimpsed world of the BGP speakers - diminutive routers that sit at the topological edge of our networks.

Understanding BGP data is critical given its foundational nature regarding the Internet. This protocol is so deeply entrenched in the communications infrastructure that updating and upgrading is extremely difficult, even though a secure, authenticated version of the protocol is necessary. Router misconfigurations and purposeful attacks can render dark whole swaths of the Internet in a very short period of time. Unsurprisingly, much research exists - both in terms of detection and analysis - to cope with these dangers.

We present BGPeeP, a new tool for visualizing BGP update messages using a novel visual encoding of IP-space. Our tool complements the many existing

tools for viewing routing data in that it provides a unique picture of the data at lowest-level, rather than focusing on overall AS connectivity. We provide intuitive methods of interaction so that network operators can quickly isolate the data of interest and produce a useful picture for aiding BGP problem diagnosis.

## 2 Related Work

Given its import and data-intensive nature, it is no surprise that BGP has received given a thorough treatment from the visualization community. Several recent works focus on providing a high-level view of the ever-changing topology amongst autonomous systems, the best known of which is BGPlay [1]. With this work, Colitti et al. allow network operators to monitor or observe the reachability of a specified prefix from the perspective of a given border router. Colitti’s colleagues at Roma Tre University extended BGPlay to include the AS “importance” hierarchy in the visualization, drawing inspiration from topographical cartography [2]. Wong, et al described a method [3] of clustering voluminous BGP data - called stemming - in order to present a clear, useful visualization of routing from the perspective on one AS, and demonstrate how their technique can help diagnose BGP anomalies. The described system, like BGPlay, uses animation to show how the routing situation changes over time, allowing network operators to quickly “see” the story rather than crawling through thousands, perhaps millions, of update messages. The LinkRank visualization [4] developed by Lad, Masset, and Zhang, provides a higher level view of AS connectivity in that it simultaneously visualizes the *number* of routes lost or gained between many different ASes. It can help show how the overall topology changes when the link between two ASes fails.

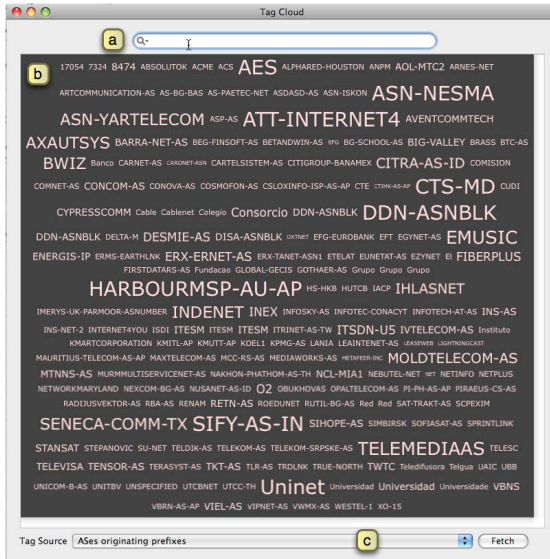
The above cited works all focus on presenting a high-level view of BGP update traffic, aiming to free the network operator from the river of data flowing silently among the many border routers. Other tools deal with the data at a somewhat lower level. Teoh, et al. provide a suite of visualization tools [5] for examining individual update messages with the aim of problem diagnosis and root-cause analysis. They also provide a technique for clustering and picturing these messages as single, categorizable events.

Some visualization applications focus on specific types of BGP misconfiguration or attack. For example, Teoh et al. describe techniques for visualizing Change of Origin AS events [6] [7]. These tools include an encoding of the overall IP-space using a quad-tree. More recently, they have extended their previous work on classification of update messages into BGP events, and provide new tools for showing a more global picture of BGP activity with an application suite called BGPEye [8]. This tool provides several different visualizations of BGP activity from the perspective of one collection point. The different perspectives given by this tools show not only the overall BGP activity, but also how it affects the network connectivity at the data collection point.

Most existing tools focus either on the high-level picture, or specific aspects of the low-level data. BGPeep instead provides a general tool for visually examining







**Fig. 2.** The AS tag cloud displays (b) the ASes matching the search criteria, specified using the selection popup (c). Tag sizes are proportional to the relative number of update messages sent in the selected time by the associated AS. Users can filter and search the presented AS subset using the search field (a), select ASes for display by clicking the tag, or obtain additional AS information using the detail viewer.

### 3.1 The Data

BGPee currently supports ASCII dumps in MRT ‘machine’ format, such as those provided by the University of Oregon RouteViews project [9]. These dumps are collections of all BGP update messages sent by peer ASes to border routers participating in the RouteViews project. Each record in a given dump is either a BGP update or withdrawal message, with contents as specified in IETF RFC 4271 [10]. For the purposes of our discussion, it is sufficient to know that a message contains at least the follow parts:

**Peer AS.** This is the topologically neighboring border router which sent the update message to the collection router.

**Announcement/Withdrawal.** An announcement message is when an AS claims a certain portion of IP-space is reachable through it, and a withdrawal is when it revokes this claim.

**A Prefix.** The portion of IP-space for which this message applies. Essentially, this is an IP address with a set number of bits marked as fixed. The remainder - the variable bits - represent the range of address the prefix covers. For example, 192.168.1.0/24 represents the range of addresses starting at 192.168.1.0 with the first 24 bits fixed. The remaining 8 bits are variable, covering 256 unique IP addresses. IETF RFC 4632 [11] contains a detailed description of CIDR addressing.

**AS Path.** For announcement messages, the AS path is simply an ordered list of ASes that lead to a given prefix. The last AS on the path to a given network prefix is called the *origin AS*. That AS is said to have *originated* that prefix.

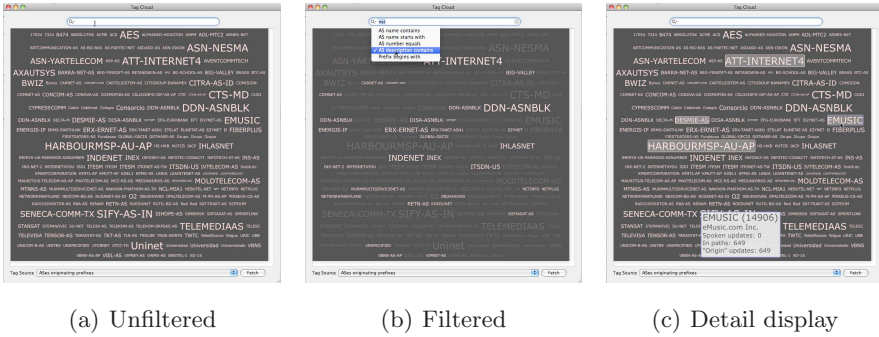
### 3.2 AS Tag Cloud

Tag clouds are an interaction technique that has been popularized recently on the world wide web. A tag cloud is a list of visual elements, typically words, that are assigned different sizes based on some metric. A user can click directly on a tag to instigate some action in the program or browser, often navigating to some other application content. Though little academic research describes tag clouds, Rivadeneira, et al. [12] present recent work which provides an overview and describes evaluation strategies.

The AS tag cloud (Figure 2) is the primary interface element for querying the BGP update data set. Users can select a query to run against the data using the selection popup marked ‘c.’ These queries are written in a SQL-like language, and are modifiable by the application designer. BGPeep currently supports queries for returning all ASes which originate prefixes in the current data set, for returning all peer ASes which present updates or withdrawals, and for returning all ASes mentioned in an announced AS path. These queries can be temporally restricted using the timeline, discussed below in Section 3.4.

The query populates the cloud with AS tags, listed alphabetically by AS name, with the sizes assigned based on the relative number of matching updates associated with the given AS. For example, if the user queried ‘ASes originating prefixes,’ then the tag sizes would be based on the number of prefixes the AS originated in the selected time range. As such, the most active ASes immediately stand out against the background noise of the less active systems. Users can select up to four ASes to display in the prefix viewer at a given time by clicking the appropriate tags. Right-clicking a tag calls up a AS detail view, as depicted in Figure 3(c). This panel, which contains some simple statistics regarding the selected AS for the given dataset, floats above the cloud when requested, and is invisible otherwise. This provides a mechanism for unobtrusively providing information that is not typically useful, but that the user might need to infrequently access.

The tag sizing is meant to help users in their initial encounter with a new data set in order to identify the most active ASes. But for certain tasks, the tag cloud sizing might not be useful. For example, if the user is searching for a particular AS, or activity concerning a particular prefix, then the assigned sizes might simply be distracting. For that reason, BGPeep provides a variety of methods for filtering the cloud. The search field, marked ‘a’ in Figure 2, allows the user to select from a list of pre-defined searches such as ‘AS name contains...’, and ‘AS announces prefix...’. When the user enters text, all tags *not* matching the criteria are visually darkened, while matching tags retain their visual impact (Figure 3(b)). As such, a user can quickly locate the AS of interest and select it for display in the other application views.



**Fig. 3.** Filtering allows BGPeep users to more easily find the ASes of interest. Figure 3(a) shows a normal, unfiltered tag cloud. In Figure 3(b), the cloud is filtered showing those ASes with a long description containing the string ‘net’. Figure 3(c) shows several selected ASes and one displayed using the AS detail panel, which users can show or hide by right-clicking on a tag.

### 3.3 Prefix Visualization

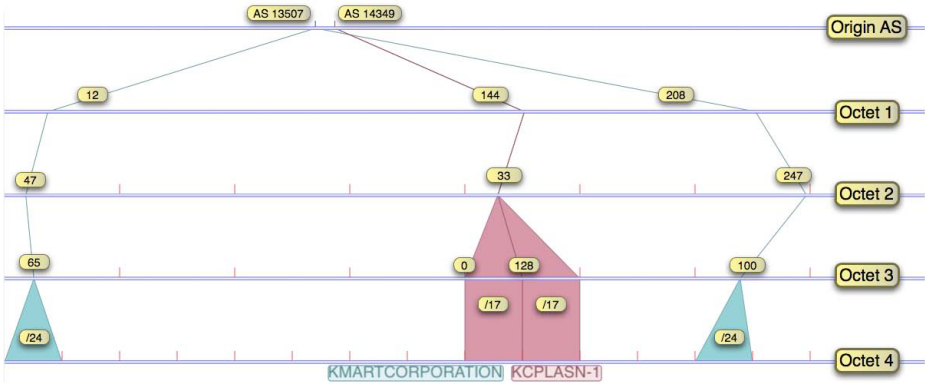
The prefix visualization in the main visual element of BGPeep, and provides a novel view of IP-space in a BGP-centric fashion. The view contains five axis, and selectable, colored labels naming the currently visualized ASes. Figure 4 provides a labeled version of the prefix visualization showing only a few updates. The visualization components, rendering technique, and interaction methods are described below.

**The Axes.** The topmost axis in the display represents the AS associated with the update message, either as peer AS or originating AS, and has values ranging from the lowest AS number in the data set to the highest. For example, if the ‘lowest’ AS mentioned in the data set was 100, and the highest was 41000, then the midpoint of the first axis would correspond to AS number 20000.

The subsequent axes correspond to the various octets of the prefix’s CIDR-style IP address range. For example, for the prefix 192.168.1.0/24, 192 would be plotted on the second axis, 168 on the third, 1 on the fourth, and 0/24 on the fifth. The axis for the second, third, and fourth octets are subdivided to prevent display over-plotting. Using the method described below, a shape is drawn through these axes which shows the viewer which AS(es) announced or withdrew the prefix and what portion of IP-space it covers.

**Update Rendering.** Each prefix is rendered as a shape that passes through all five axis. There are three situations that contribute to the overall shape of the prefix:

**AS to Octet 1.** This portion of the shape is a line connecting the AS associated with the update to the location of the first octet, where the Octet axis is valued 0-255 from left to right.

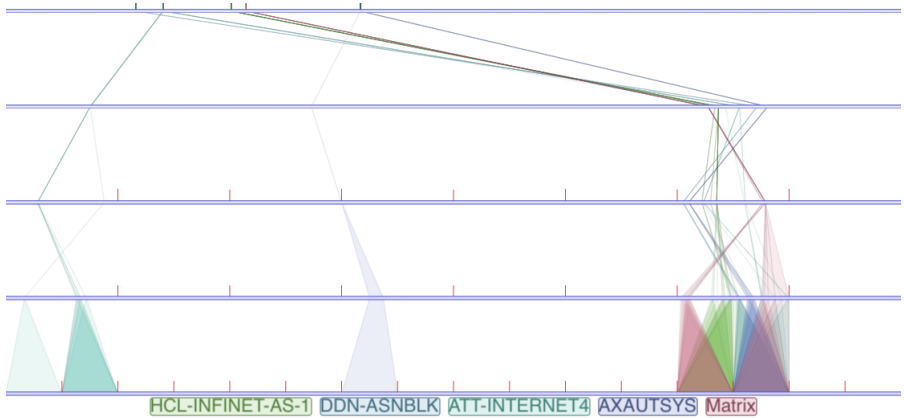


**Fig. 4.** The prefix visualization, labeled for explanation, showing four prefix announcements from two ASes. The top axes denotes the AS associated with the update message, and the subsequent four axes represent the four IP address octets. Each axis is subdivided as detailed in section 3.3 to avoid extensive display over-plotting and allow easier visual decoding of the prefix. Variable portions of the CIDR address are drawn as polygons that extend across the addresses in the range. Note that in this particular image, we also see a potential inefficiency in the announcements for AS 14349. Instead of announcing two /17 prefixes, it could instead announce one /16 and handle the de-aggregation internally.

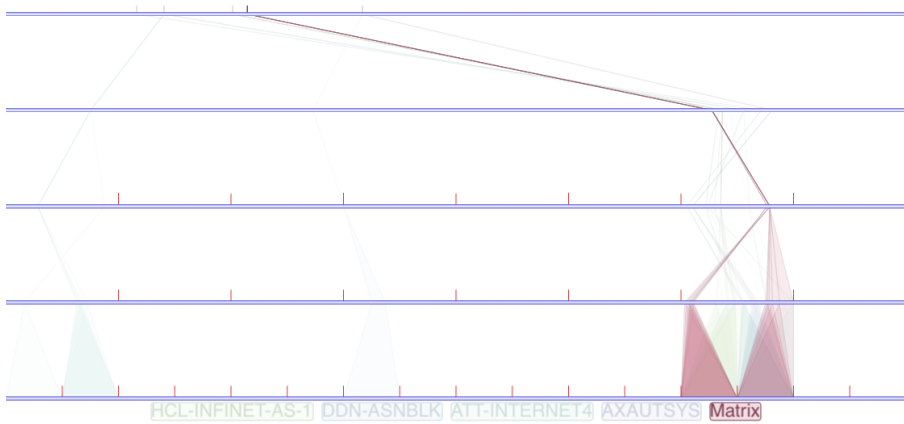
**Octet to Octet.** Octet 2, and later axes are subdivided into a set number of sections, demarcated with small hash marks. Each section on the axis corresponds to the entire 0-255 octet range for prefixes whose first octet falls above it. This is evident from Figure 4. Prefix 12.47.65.0/24 flows down the left side of the display due to this subdivision. This allows many more prefixes to be shown simultaneously without overlapping.

**Ranges.** Almost all BGP updates involve a range of addresses, usually 256 unique IPs or more in size. Ranges are depicted using a triangle which covers all the addresses contained in the update. Because most updates are /24 or greater in size, the shape between axes 4 and 5 are usually a triangle or a rectangle. This depicting of ranges means that larger ranges are fuller, larger shapes. Also, prefixes announcing less than 256 addresses - usually indicative of a misconfiguration - clearly show up as skinny shapes between the final two axes.

All updates for the selected ASes are rendered simultaneously in the display to allow intra- and inter-AS comparison. Each prefix is rendered with a user-modifiable base opacity. This allows the user to see - at a glance - very chatty ASes. If the base opacity is set very low, yet the prefix visualizer shows a relatively solid-colored prefix, then this prefix was announced many times in the selected time period. The user can then examine the specific timing details in the data view table.

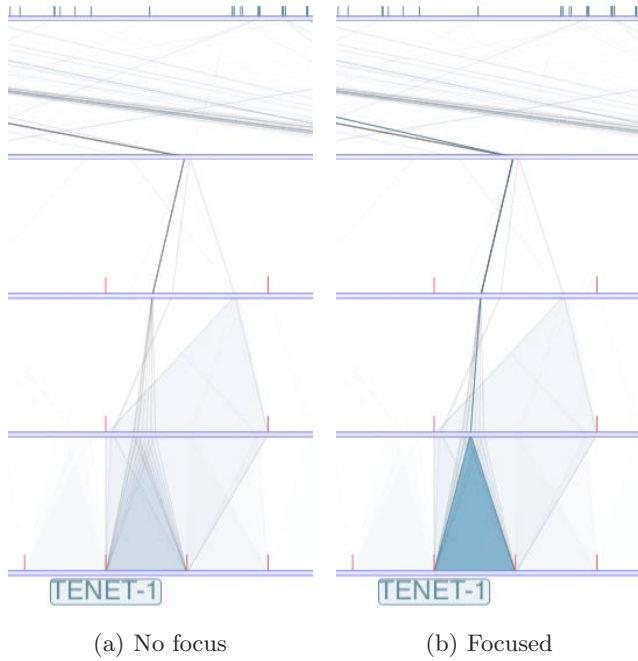


(a) No emphasis

(b) AS *Matrix* emphasized

**Fig. 5.** AS emphasis can help isolate a particular AS for closer examination, or compare/contrast two similar ASes. In the top image, three displayed ASes announce prefixes in the same octet range, leading to some cluttering on subsequent axes. In the bottom image, the unselected ASes are assigned 10% opacity, so that the selected AS is highlighted, but all context is not lost.

**AS Emphasis.** Very active ASes can sometimes present many update announcements, which makes comparison with other ASes difficult. As a remedy, BGPeeP allows the user to select a specific AS for *emphasis*. When selected, the rendered prefixes for that AS retain their opacity, while all other updates and their corresponding labels are rendered with significantly reduced opacity. The user can select another AS for emphasis either by selecting it from the selection popup in the data view, or by clicking its label in the prefix visualizer. Figure 5



**Fig. 6.** Individual prefix focus visually isolates the selected prefixes in a potentially crowded display. The selected prefixes are rendered with full opacity, regardless of the set base opacity, and are rendered on top of the other, non-focused prefixes. This allows the user to highlight the prefix of interest without losing the context of the selected AS' overall announcement activity.

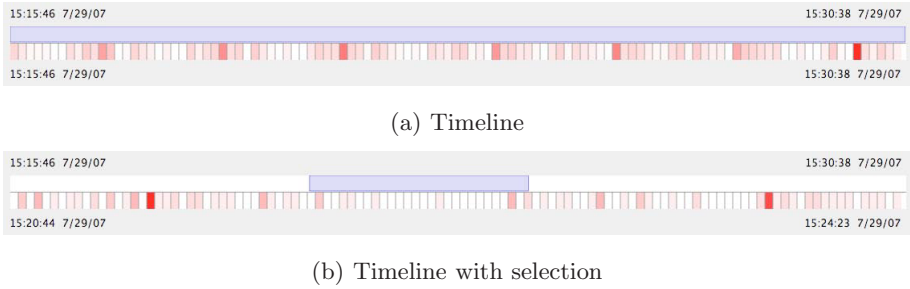
shows AS emphasis in action. Note that the other ASes are not entirely removed - simply faded - so as to retain context for comparison.

**Prefix Focusing.** Similarly, the user might see a specific prefix in the data view that she wants to highlight in a visually busy mass of updates. In this case, the user can enable prefix focusing by clicking the associated check box in the data view and then selecting one or more prefixes in the table. When focused, a prefix is rendered last - and therefore on top of the other prefixes - with full opacity.

### 3.4 Timeline

The timeline allows the BGPeep user to temporally restrict queries in the other views, including the initial tag cloud query. When the application starts, the entire time range for the loaded data set is pre-selected, as depicted in Figure 7(a). By clicking and dragging in the upper portion of the timeline, the user can intuitively select only a portion of the time range for display, Figure 7(b).

The bottom half of the timeline is the update frequency visualization, which provides the user with a quick overview of update 'hot spots' for the selected



**Fig. 7.** The timeline view allows BGPeep users to temporally restrict both the rendered messages and the initial AS cloud search. The top portion of the timeline is a simple rectangle which represents the portion of the loaded data set currently selected. The bottom portion divides the currently selected time range into equally-sized rectangles and colors each region based on the number of updates in the corresponding time slice. The more saturated the color, the greater the number of updates.

AS. This portion of the display is sub-divided into equally sized rectangles. Each rectangle corresponds to an equal slice of time in the already selected time range. BGPeep colors each rectangle based on the *relative* number of associated updates that occur in that time. The more saturated the color, the greater the number of updates in that time slice, relative to the whole selected range. By alternating the selection between two ASes, the user can compare and contrast when the main activity occurred for each system, which might be important for cases such as that detailed in Section [4.2](#).

### 3.5 Data View

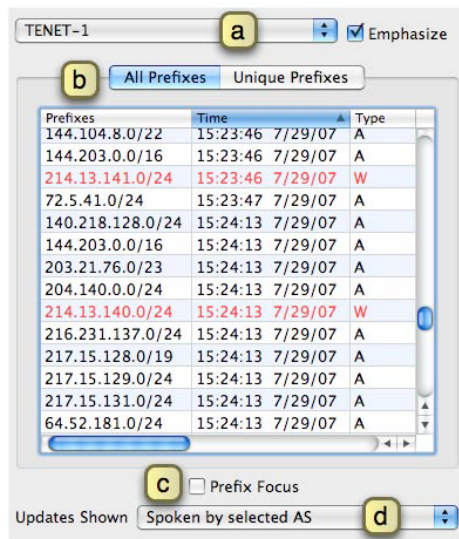
The data view provides controls for filtering the displayed update messages in the prefix visualizer, and also provides a more traditional, table-based view of the update messages. The various interface elements are described in Figure [8](#). The topmost selection popup, labeled ‘a,’ allows the user to choose which AS’ data is shown in the table, marked ‘b’.

As previously noted, the data view contains interface elements for highlighting specific ASes or prefixes.

The table view contains two distinct presentations of the update messages for the currently selected AS. In the first, *all* update messages - including duplicates - for the selected AS are listed in the table. For each update, the time, the type, the prefix, the AS path, and other data are given. Users can sort on any column in order to arrange the data in the most convenient manner. If an update is a withdrawal, the text for that update is colored red. This allows the user to quickly spot flapping routes, as described in section [4.1](#).

The second table view shows only unique prefixes - filtering out repeat announcements - which can be more useful for initial data exploration. Often an AS will make repeated announcements regarding the same prefix, with the same information. A user can combine all of these updates into a single table line which





**Fig. 8.** The data view provides controls for visually isolating specific ASes (a), specific prefixes (c), examining low-level details of individual update messages (b), and varying the type of visualized update messages (d). Note that the table view assigns a distinct color to the text of withdrawal messages, which helps to identify cases of route flapping.

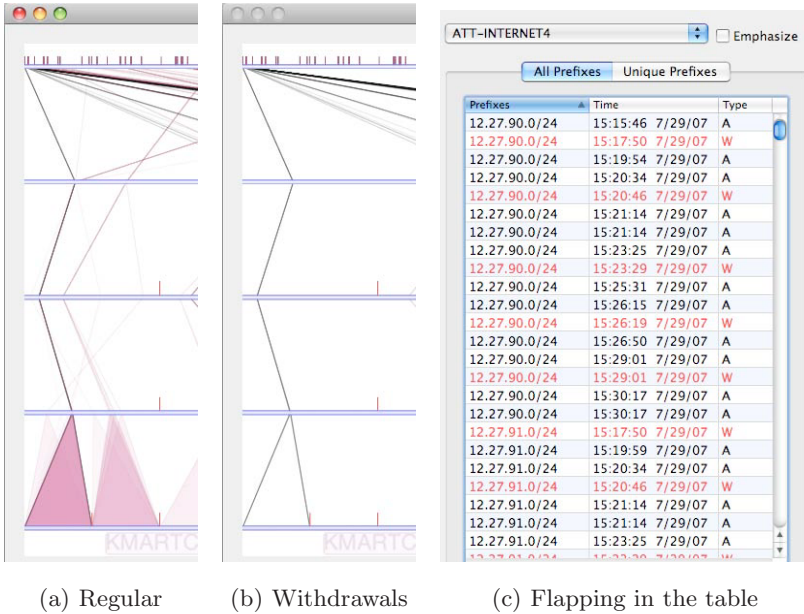
has only the prefix and the count of times it was announced. This is useful if the user wants to quickly focus each individual prefix announced by a particular AS in order to get a sense of the overall IP-space ownership claimed.

## 4 Results

The primary benefit of BGPeep is the ease by which it allows a user to quickly navigate and visualize the traffic observed at a particular router. However, there are some particular cases of router misconfiguration or mischief for which BGPeep can provide a unique perspective.

### 4.1 Route Flapping

Route flapping occurs when an AS repeatedly announces and then withdraws a specific network prefix. While flapping usually has little effect on the overall topology of the Internet, it can generate excessive network traffic and can unnecessarily add to the workload of computationally constrained routers. BGPeep provides features to help identify such cases. First, it renders withdrawal messages with a fixed opacity, black outline as detailed in Figure 9. The result is that moderately-to-heavily flapping routes appear very different from those that are simply announced. When the user adjusts the base rendering opacity to zero, the withdrawn routes - even those withdrawn only once - leave behind a visible



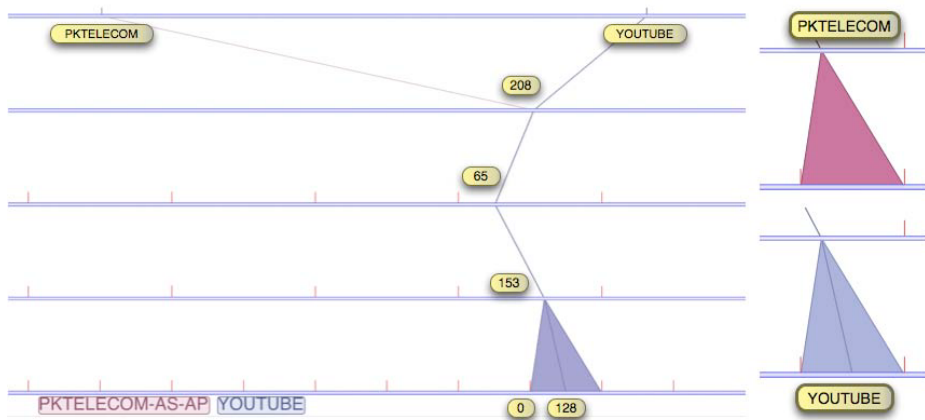
**Fig. 9.** Withdrawal messages are rendered slightly differently from prefix announcements. Announcement borders have the same hue and base opacity as the fill color for the given AS. Withdrawal messages, however, are drawn with a black border at a fixed opacity. As a result, a flapping route will have a ‘pencil-sketch’ look (left) when rendered with non-zero base opacity, and will leave behind a visual residue (center) when the base opacity is set low. Combined with the distinct coloring and sort options available in the data view table (right), it is easy to identify flapping routes.

outline. He or she can then use the data view table to see specific details of the offending update messages by sorting on time and prefix. Withdrawal messages are printed with red text, and the alternating red-black text makes flapping easy to identify.

## 4.2 Prefix Highjacking

Prefix Highjacking occurs when AS A mistakenly or purposefully announces itself as the owner of AS B’s network prefix. If the path announced by A is shorter than that in the current global routing table, or the newly announced prefix is more specific, then many systems will mistakenly route packets truly destined for B to A.

A high-profile case of prefix hijacking occurred on February 24, 2008 when Pakistan Telecom announced itself the origin AS of 208.65.153.0/24, an IP range owned by YouTube. The announcement was meant to block YouTube from within Pakistan, but somehow it mistakenly leaked out to neighboring ASes. As a result, the announcement propagated throughout the Internet and many hosts could no longer properly route packets to YouTube. Eventually, YouTube announced the



**Fig. 10.** This shows the prefix hijacking of YouTube’s 208.65.153.0/24 by Pakistan Telecom on February 24th, 2008. Note how the AS lines ‘funnel’ into the same prefix. Also note the later de-aggregation purposefully announced by YouTube to combat the hijacking.

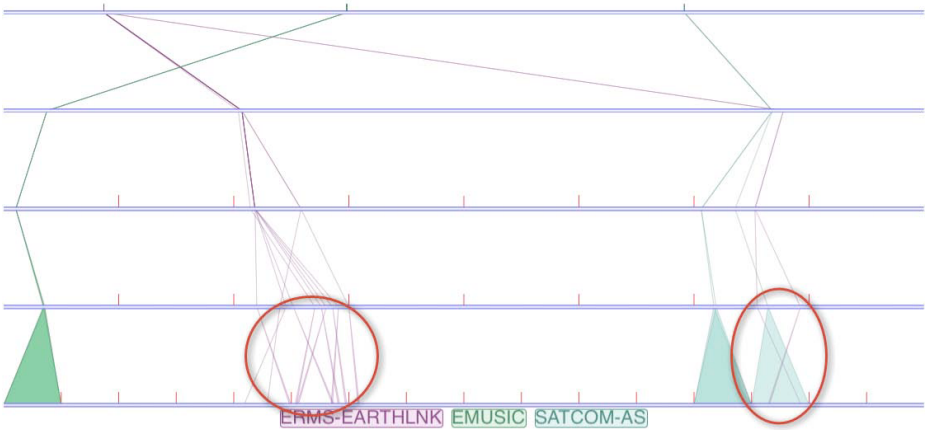
same IP-space as two smaller /25 prefixes, which repaired connectivity for many hosts. The RIPE news archive contains a detailed analysis of this event [13].

Figure 10 shows this event, as rendered in the BGPee prefix visualizer. The image clearly shows that both Pakistan Telecom and YouTube claim ownership of the same prefix, and that YouTube later announced two consecutive, smaller prefixes. In order to obtain this image, we loaded data from the RouteViews archive for February 24th into BGPee and searched for ASes involved with prefixes beginning with ‘208.65.153.0’ bounded by the time range of the event. The timeline frequency visualization provided temporal context to the selected announcements.

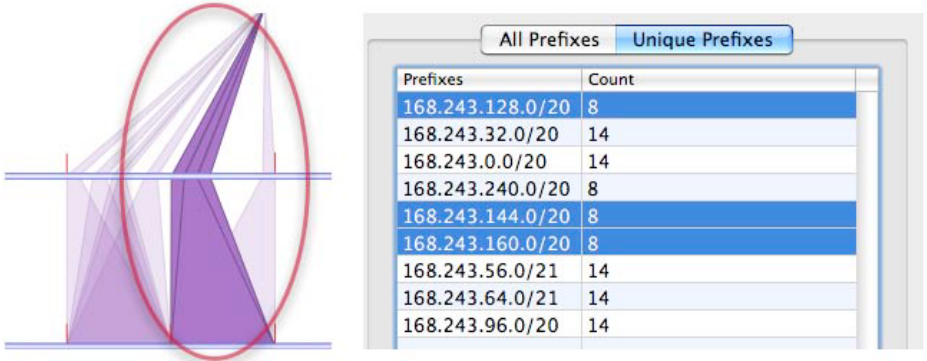
### 4.3 Inefficient Announcements

Inefficient prefix announcements, like flapping, can have a negative effect on overall routing performance by generating unnecessary traffic, and by increasing the size of the global routing table. For example, most ASes do not announce or propagate announcements for prefixes smaller than 256 hosts, since almost all ASes deal in chunks of IP-space of size /24 or greater. BGPee’s prefix visualization clearly shows suspiciously small announcements, as depicted in Figure 11. Here the suspicious updates are obvious because they deviate from the typical visual pattern of triangles between the bottom two axes.

Figure 12 shows another example of how BGPee can highlight potentially bad route announcements. Here an AS has announced three prefixes which are consecutive in IP-space. Consulting the data view, we saw the for all three, the announcement details, including the AS Path, were identical. It is likely that these three prefixes could be collapsed into one announcement.



**Fig. 11.** In this figure, we see that ERMS-EARTHLINK has announced many sub-/24 prefixes, including a few individual IP addresses. Because of the deviation from the normal visual pattern, these ‘spikes’ on the last axis visually jump out at the user. This could be of particular use in an animated, real-time monitoring application of BGPeep.



**Fig. 12.** An example of possibly inefficient route announcements. The focused routes, circled in red, announce consecutive ranges of IP address with the same AS path. Most likely, the originating AS could collapse these announcements into one consecutive IP range and perform needed de-aggregation internally. This helps keep the global routing table small, which is important given the limited computing power most BGP speakers possess. These kinds of striated patterns are easy to see using BGPeep.

## 5 Future Work

This work presents the core visual elements and interaction design for BGPeep. We’re currently working on extending our tool to incorporate animation to show the announcements over time. With this feature, the user could place the tool into

a ‘monitoring’ or ‘playback’ mode and watch as live prefixes or a selected data subset arrive at their AS’ border routers. As each update arrives, it would briefly flash with full opacity, and then slowly fade away. In such an implementation, flapping routes, overly chatty neighbors, and inefficient announcements would be easy to spot. This could be combined with an overlay of the particular ASes own prefix space, so that hijacking and de-aggregation would be immediately evident to a watchful eye.

## 6 Conclusion

We have presented BGPeep, an interactive system for visualizing BGP update messages at a lower level than most existing applications. Using BGPeep, a network operator can interactively explore the update traffic as seen by her border routers, and better understand the traffic generated by peering ASes. In addition, our unique encoding of IP-space affords the user a fresh perspective on such data sets, and can clearly show IP-space de-aggregation, prefix hijacking, and route flapping. We believe BGPeep to be a useful addition to the already powerful arsenal of visualization tools available for contending with the data avalanche BGP presents.

## Acknowledgements

This research was supported in part by Intel Corporation, the U.S. National Science Foundation through grants CCF-0634913, IIS-0552334, CNS-0551727, and OCI-0325934, and the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FC02-06ER25777.

Special thanks to the University of Oregon Route Views Project for providing the data used in the development of BGPeep.

## References

1. Colitti, L., Di Battista, G., Mariani, F., Patrignani, M., Pizzonia, M.: Visualizing interdomain routing with *bgplay*. *Journal of Graph Algorithms and Applications* 9, 117–148 (2005); Special Issue on the 2003 Symposium on Graph Drawing, GD 2003
2. Cortese, P., Di Battista, G., Moneta, A., Patrignani, M., Pizzonia, M.: Topographic visualization of prefix propagation in the internet. *IEEE Transactions on Visualization and Computer Graphics* 12(5), 725–732 (2006)
3. Wong, T., Jacobson, V., Alaettinoglu, C.: Internet routing anomaly detection and visualization. In: *International Conference on Dependable Systems and Networks, 2005. DSN 2005. Proceedings*, 28 June-1 July 2005, pp. 172–181 (2005)
4. Lad, M., Massey, D., Zhang, L.: Visualizing internet routing changes. *IEEE Transactions on Visualization and Computer Graphics* 12(6), 1450–1460 (2006)
5. Teoh, S.T., Ma, K.L., Wu, S.F.: A visual exploration process for the analysis of internet routing data. In: *VIS 2003: Proceedings of the 14th IEEE Visualization 2003 (VIS 2003)*, Washington, DC, USA, p. 69. IEEE Computer Society, Los Alamitos (2003)

6. Teoh, S.T., Ma, K.L., Wu, S.F., Zhao, X.: Case study: interactive visualization for internet security. In: VIS 2002: Proceedings of the conference on Visualization 2002, Washington, DC, USA, pp. 505–508. IEEE Computer Society, Los Alamitos (2002)
7. Teoh, S.T., Ma, K.L., Wu, S., Jankun-Kelly, T.: Detecting flaws and intruders with visual data analysis. *Computer Graphics and Applications* 24(5), 27–35 (2004)
8. Teoh, S.T., Ranjan, S., Nucci, A., Chuah, C.N.: Bgp eye: a new visualization tool for real-time detection and analysis of bgp anomalies. In: VizSEC 2006: Proceedings of the 3rd international workshop on Visualization for computer security, pp. 81–90. ACM, New York (2006)
9. University of Oregon RouteViews Project, <http://www.routeviews.org>
10. Rekhter, Y., Li, T., Hares, S.: A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard) (2006)
11. Fuller, V., Li, T.: Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC 4632 (Best Current Practice) (2006)
12. Rivadeneira, A.W., Gruen, D.M., Muller, M.J., Millen, D.R.: Getting our head in the clouds: toward evaluation studies of tagclouds. In: CHI 2007: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 995–998. ACM, New York (2007)
13. YouTube Hijacking: RIPE Analysis, <http://www.ripe.net/news/study-youtube-hijacking.html>

# Large-Scale Network Monitoring for Visual Analysis of Attacks

Fabian Fischer, Florian Mansmann, Daniel A. Keim, Stephan Pietzko,  
and Marcel Waldvogel

University of Konstanz, Computer and Information Science,  
78457 Konstanz, Germany

{Fabian.Fischer, Florian.Mansmann, Daniel.Keim,  
Stephan.Pietzko, Marcel.Waldvogel}@uni-konstanz.de  
<http://infovis.uni-konstanz.de>

**Abstract.** The importance of the Internet and our dependency on computer networks are steadily growing, which results in high costs and substantial consequences in case of successful intrusions, stolen data, and interrupted services. At the same time, a trend towards massive attacks against the network infrastructure is noticeable. Therefore, monitoring large networks has become an important field in practice and research. Through monitoring systems, attacks can be detected and analyzed to gain knowledge of how to better protect the network in the future. In the scope of this paper, we present a system to analyze NetFlow data using a relational database system. NetFlow records are linked with alerts from an intrusion detection system to enable efficient exploration of suspicious activity within the monitored network. Within the system, the monitored network is mapped to a TreeMap visualization, the attackers are arranged at the borders and linked using splines parameterized with prefix information. In a series of case studies, we demonstrate how the tool can be used to judge the relevance of alerts, to reveal massive distributed attacks, and to analyze service usage within a network.

**Keywords:** visual network monitoring, visualization for network security, large-scale netflow analysis.

## 1 Introduction

The increase of network attacks in terms of coverage, intensity, and aggressiveness is one of the most difficult challenges for network administrators today. A major part of these developments is to be attributed to so-called *Botnets*, which play a critical role in large-scale attacks [1]. In particular, more and more attacks focus on corporate and governmental networks with the goal of stealing confidential information or blackmailing companies whose business model depends on uninterrupted availability of their business and services. These facts point out the need for software to effectively and efficiently analyze network traffic both in real-time and for forensic purposes. The latter analysis can be especially useful for discovering compromised hosts within the local network.

Since the analysis of flow data in current analysis systems is only supported to some extent, we propose a novel analysis system called *NFlowVis* with the goal of enabling quick visual insights into communication patterns. The system is capable of storing NetFlow data of large systems, linking these flows to alerts from intrusion detection systems or public warnings, and to visualize flows between external and internal hosts. Using a TreeMap visualization, we depict the local network infrastructure emphasizing high traffic subnets. On top of this visualization, Splines in selected colors are utilized to connect the external host with the local communication partners, thereby revealing insight into communication patterns of malicious and legitimate network traffic.

To protect the privacy of our network users, we anonymized all IP addresses used while maintaining the grouping according to the higher level prefixes. Therefore, conclusions about the usage of a particular hosts, either internal or external ones, cannot be drawn from the displayed figures.

The remainder of this paper is structured as follows: Section 2 discusses related work, Section 3 introduces our *NFlowVis* application, Section 4 presents real-world case studies, Section 5 discusses the tool's applicability and scalability, and Section 6 summarizes our contributions.

## 2 Related Work

Visualization for computer security is a relatively young research field. While substantial research has been conducted in the field in the last few years, for brevity this section will focus on visual network traffic monitoring and discuss the roots of the used visualization concepts.

In the Open Source community, there are two popular tools: *NfSen* [2] and *Stager* [3]. Both tools comprise web frontends to display aggregated information about previously captured netflows. In the backend, database management systems enable efficient access to detailed information and efficient generation of aggregated reports. For visual analysis, both systems use line charts for displaying temporal overviews of network system load. While *Stager* only stores highly aggregated data, *NfSen* reverts back to the original flow data for detailed analysis.

Since network monitoring is particularly important for the health of the commercial network infrastructure, there exist a multitude of commercial systems. In contrast to the previously discussed tool, commercial systems such as *IBM Aurora* [1], *NetQoS Reporter Analyzer* [2], *Caligare Flow Inspector* [3], and *Arbor Peakflow* [4] often include methods for intrusion detection in which generated alerts can be examined through interactive reports. However, the used statistical charts and diagrams only scale to a limited number of alerts or highly aggregated information.

Visualization approaches in network monitoring aim at supporting the system administrator in the exploration of network traffic by means of interactive visual

<sup>1</sup> <http://www.zurich.ibm.com/aurora>

<sup>2</sup> <http://netqos.com/solutions/reporteranalyzer>

<sup>3</sup> <http://www.caligare.com/netflow>

<sup>4</sup> <http://www.arbornetworks.com>



displays. *NVisionIP* [4], for example, enables visual pattern recognition and drill-down functionalities to inspect suspicious machines. *TNV* [5] is a network traffic visualization tool focusing on temporal aspects by means of a time versus internal host matrix, which details traffic flows for each host and links the external communication partners on the side. The home-centric network view of *VISUAL* [6] is probably closest to our proposed visualization since a matrix showing all internal hosts in the center is linked to external communication partners using straight connecting lines.

In contrast to this work, we made two major conceptual changes:

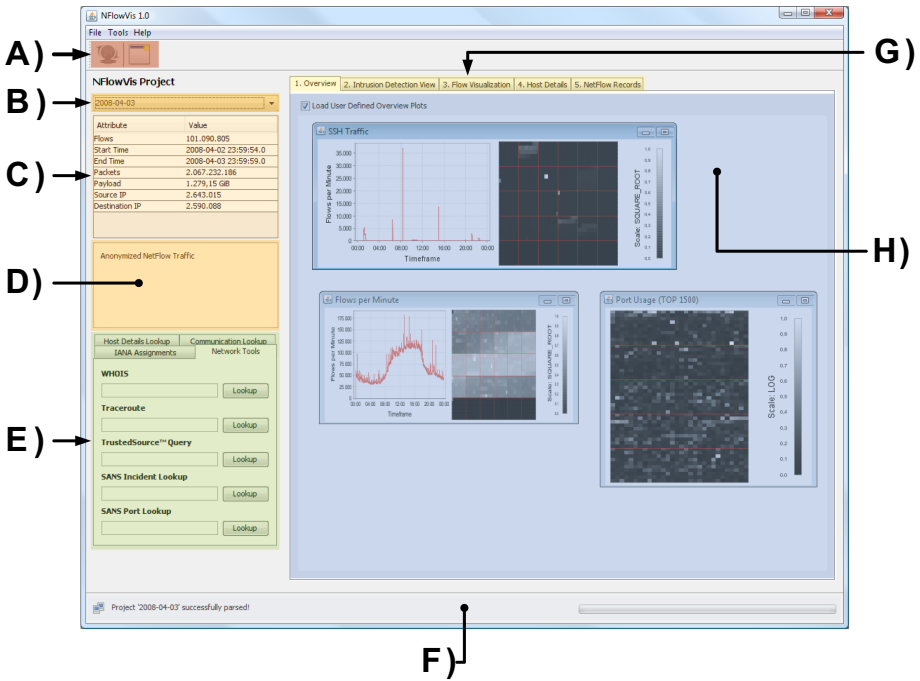
- a) Instead of using a matrix view for the internal hosts, we employ a *TreeMap* [7] visualization, which hierarchically maps the monitored network infrastructure to prefixes of various granularity. Unlike in our previous work [8], high-load entities are thereby enlarged.
- b) Rather than using straight lines to link the communication partners, we employ *Hierarchical Edge Bundles* [9] to visually group related flows, and thereby avoid visual clutter.

While we visualized flows using Hierarchical Edge Bundles with both start and end point within a *TreeMap* visualization in an earlier work [10], the work presented in this paper explicitly focuses on a home-centric network view, which represents the local IP prefixes or addresses in a *TreeMap* and places the external hosts at its border.

### 3 Visual Analysis of Attacks

Keeping the general workflow of a network analyst in mind, we developed *NFlowVis* to interpret the relevance of network security alerts. The system supports this full workflow through its five analysis views with a general network *overview*, an integrated *intrusion detection view*, the *flow visualization* of attackers' connections, a detailed *host view*, and the full *NetFlow records* of the specified communications as the most detailed view. In the graphical user interface these views are represented through several tabs to emphasize the drill-down and filtering process. Fig. 1 describes the design of *NFlowVis*: the connection settings to the database server and project creation wizard (A), project selection (B), key data of the selected project (C), textual description (D), fast access to external tools and internal queries to retrieve host details (E), current status and progress of operations (F), and the previously mentioned data exploration views ordered according to the levels of details (G).

Within the *overview* tab, the system provides several user-defined plots (H). With the help of these graphs the analyst is able get a rough overview of the actual network situation and utilization detailing the aggregated traffic and port usage within the whole network. To visualize these time series we use line charts and grouped line-wise pixel arrangements. The use of both visualizations combines the advantages of the well known line charts and the pixel visualization, which provides identification of every single minute and enables recognition of



**Fig. 1.** User interface of the NFlowVis system showing the annotated main view (left)

recurring patterns. The overview also provides an interactive port activity map to identify the most active ports.

The *intrusion detection view* links IDS alerts or public warning lists with the full NetFlow records and displays the textual data in a colored table. By calculating some statistics concerning the influence of the attacking hosts on the whole network, the analyst gets a more realistic view of the relevance of the alerts. For further investigation of a number of hosts, it is possible to select the attackers and to visualize their connections to explore their influence. Besides the integration of external IDS alerts and warning lists, this view also provides a template editor to define database queries, which can directly access the flow data. We included a variety of different predefined templates, such as grabbing all SSH traffic or other suspicious activities.

Within the *flow visualization* view, we map the monitored network to a TreeMap visualization in the center of the display and arrange the previously selected attackers at the borders. The TreeMap comprises all hosts related to the attacking hosts during the chosen timeframe, which can be defined in the project creation wizard. Flows between the attackers and the local hosts or prefixes are displayed through Splines, whose control points are the center points of the network prefixes of various levels and the attackers on the outside. The size of the TreeMap rectangles (weight), their background color, and the Spline

width can be set to arbitrary attributes of the aggregated flow data, e.g., flow count, transferred packets, or bytes.

In the default configuration the Spline color correlates with the attacker's IP prefix, which better shows the behavior of attackers with similar prefixes supporting the analyst in gaining insight into the distribution of the attacking IP addresses. Alternatively, random colors can be chosen.

The position of the attackers is calculated based on a *k-Medoid* clustering algorithm [11], which identifies all attackers and clusters them based on similar destination hosts. Therefore, it is possible to arrange hosts with similar victims close to each other to minimize overlaps. Another positive effect is the meaningful grouping of collaborating attackers in the same cluster.

For further analysis of single hosts under attack, the analyst is able to use the *host view* detailing histograms, a port activity map, and an aggregated overview of all attackers related to the chosen host. Likewise, the original NetFlow records can be further analyzed by drilling-down and extracting the corresponding data in the *NetFlow records* view.

## 4 Case Studies

In this section, we will demonstrate, how our *NFlowVis* tool can be applied to analyze potentially successful attacks, how massive distributed SSH attacks can be displayed, and how the tool can facilitate deeper understanding of service usage within the administrated IP network.

It is widely known that brute-force and dictionary SSH attacks are on the rise, as documented by huge numbers of explicit scans on port 22. The network security officer is primarily interested in those hosts, which do not have any prevention systems and do not block or throttle incoming login attacks. Since the attacker is able to make unlimited login attempts on such servers, the probability of successful logins is drastically increased. Besides, the attacker might automatically check the host's functionality after a successful login. By increasing a threshold slider in the visualization view, it is possible to smoothly hide all splines not exceeding that threshold (see Fig. 2) in order to identify hosts with high traffic to the attacker.

It is also possible to identify previously unknown attack scenarios. We were not aware of a massive distributed SSH attack in May 2008 before conducting a visual analysis of that day. The slow and low-volume attack pattern successfully avoided a) detection by intrusion prevention systems since it did not exceed common threshold limits and b) blocking of attacker IPs on the target machines. Using *NFlowVis*, we were able to identify a massive distributed SSH attack originating from a Botnet as shown in Fig. 3. The SSH connections originated from several hundred hosts, lasted over two days, and targeted about 50 specific university servers summing up to about 20 000 connections.

Besides the analysis of attacks, *NFlowVis* can also be used to gain insight into legitimate network traffic, such as understanding of normal service usage or identification of abnormal network behavior. Fig. 4 details such a usage scenario

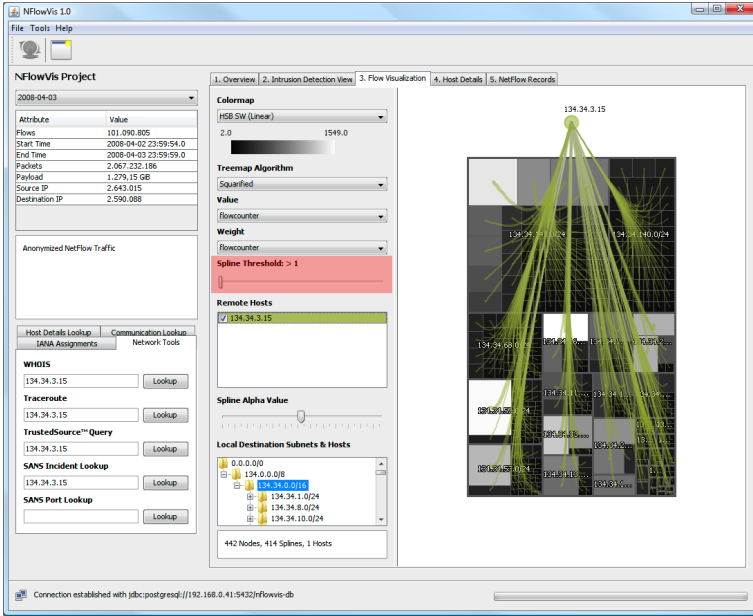


Fig. 2. Identification of possibly compromised hosts using threshold adjustment (red)

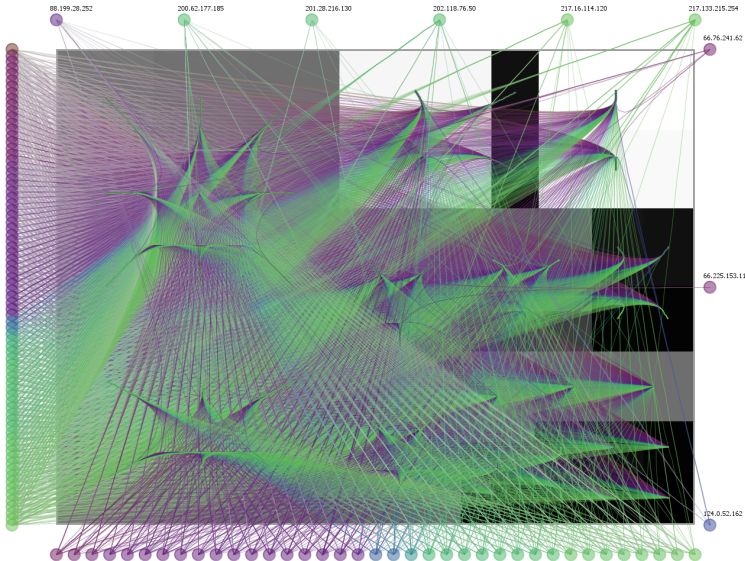
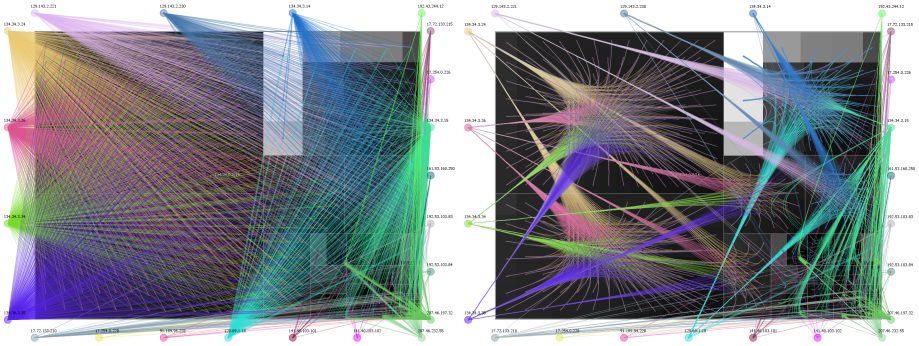


Fig. 3. Massive distributed SSH attack conducted by a Botnet with 120 Zombie computers against hosts of the university network on May 11, 2008



**Fig. 4.** Connections of local hosts to time servers visualized with straight lines (left) and Hierarchical Edge Bundles (right)

in which the security officer visualizes flows to time servers. In this case, the IP addresses of the external time servers can reveal valuable information, for example, the distribution of operating systems on the machines within the local network since Macs have a tendency to connect to Apple’s time servers and Windows machines preferably connect to Microsoft’s time servers. Similar analysis can be conducted with IRC or DNS services, which might even be more relevant for network security.

## 5 Discussion

In contrast to previous approaches to visualizing traffic between internal and external network hosts, the combination of advanced visualization techniques with a clustering algorithm provides a scalable overview of the flows as demonstrated in Fig. 4. In particular, using the Hierarchical Edge Bundles [9] after clustering the external hosts based on the common internal connection hosts allowed us to identify distributed attacks and insightful traffic patterns as demonstrated in the previous section. In addition to that, the home-centric TreeMap visualization of the network is visually scalable due to the possibility to apply it at different granularity levels of the prefixes.

## 6 Conclusions

In the scope of this paper, we presented the *NFlowVis* system to analyze intrusion detection and flow data. The user interface of the system follows a drill-down metaphor, guiding the analyst from an abstract overview of the overall network activity to aggregated views of IDS data and thorough analysis of attackers, their network traffic, and the victim hosts. In particular, this paper focused on a flow visualization technique combining a TreeMap visualization, a clustering algorithm, and Hierarchical Edge Bundles to group flows in a meaningful way. Three

small case studies demonstrated the tool’s applicability for exploring potentially successful attacks, for detection of slow and low-volume distributed attacks, and for analysis of service usage within our network.

In the future, we plan to extend our visualization technique to consider temporal aspects of attacks in more details using interactive specification of time intervals or a small multiples visualization.

## Acknowledgment

This work has been funded by the BW-FIT research project “Information at your Fingertips: Interactive Visualization for Gigapixel Displays”. We thank the anonymous reviewers of the VizSec 2008 for their valuable comments.

## References

1. McPherson, D., Labovitz, C., Hollyman, M.: Worldwide infrastructure security report, volume III. Technical report, Arbor Networks (September 2007)
2. NfSen - Netflow Sensor: A graphical web based front end for the nfdump netflow tools (2007), <http://nfsen.sourceforge.net/>
3. Oslebo, A.: Stager A Web Based Application for Presenting Network Statistics. In: Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP, pp. 1–15 (2006)
4. Lakkaraju, K., Bearavolu, R., Slagell, A., Yurcik, W., North, S.: Closing-the-Loop in NVisionIP: Integrating Discovery and Search in Security Visualizations. In: IEEE Workshops on Visualization for Computer Security, 26 October 2005, p. 9 (2005)
5. Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Preserving the Big Picture: Visual Network Traffic Analysis with TNV. In: VIZSEC 2005: Proceedings of the IEEE Workshops on Visualization for Computer Security, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2005)
6. Ball, R., Fink, G., North, C.: Home-centric visualization of network traffic for security administration. In: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 55–64 (2004)
7. Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.* 11(1), 92–99 (1992)
8. Mansmann, F., Keim, D.A., North, S.C., Rexroad, B., Sheleheda, D.: Visual Analysis of Network Traffic for Resource Planning, Interactive Monitoring, and Interpretation of Security Threats. *IEEE Transactions on Visualization and Computer Graphics* 13(6), 1105–1112 (2007)
9. Holten, D.: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Trans. Vis. Comput. Graph.* 12(5), 741–748 (2006)
10. Mansmann, F., Fischer, F., Keim, D., North, S.: Visualizing large-scale IP traffic flows. In: Proceedings of 12th International Workshop Vision, Modeling, and Visualization (2007)
11. Kaufman, L., Rousseeuw, P.: Finding groups in data. An introduction to cluster analysis. In: Applied Probability and Statistics. Wiley Series in Probability and Mathematical Statistics. Wiley, New York (1990)

# Visualizing Real-Time Network Resource Usage

Ryan Blue, Cody Dunne, Adam Fuchs, Kyle King, and Aaron Schulman\*

Department of Computer Science  
University of Maryland, College Park  
{blue, cdunne, afuchs, kking, schulman}@cs.umd.edu

**Abstract.** We present NetGrok, a tool for visualizing computer network usage in real-time. NetGrok combines well-known information visualization techniques—overview, zoom & filter, details on demand—with network graph and treemap visualizations. NetGrok integrates these tools with a shared data store that can read PCAP-formatted network traces, capture traces from a live interface, and filter the data set dynamically by bandwidth, number of connections, and time. We performed an expert user case study that demonstrates the benefits of applying these techniques to static and real-time streaming packet data. Our user study shows NetGrok serves as an “excellent real-time diagnostic,” enabling fast understanding of network resource usage and rapid anomaly detection.

**Keywords:** Real-Time, Network Administration, Force-Directed, Treemap.

## 1 Introduction

Network administrators typically look for patterns in textual router logs in real-time. Such patterns include: spotting attackers, validating routing configuration, and monitoring for unfair resource usage. Often, these can be difficult to catch by scanning logs or scripting, and it seems natural that visualization methods could apply. Unfortunately, the task of finding network traffic patterns has long been unable to benefit from information visualization; networks are difficult to visually represent, and few visualization methods have been developed to handle the real-time nature and sheer scale of network data. While forensics applications can benefit from static, historical pictures of the network, network monitoring functions require a view of the network that is always up-to-date.

In this paper, we present *NetGrok*, a tool that studies the application of powerful visualizations—force directed network graphs [7] and treemaps [13]—to problems faced by many network administrators. NetGrok allows network administrators to view network traffic at a glance, and to interact with the visualizations in novel ways that allow them to discover phenomena such as network host scanning. NetGrok’s primary goal and fundamental technical challenge is to bring network visualization techniques into the realm of real-time, streaming data. To this end, we extend both the network graph and treemap to handle

---

\* Aaron Schulman was supported by NSF-0643443 (CAREER).

real-time as well as static data. We also present an extension to treemaps, which allows the user to see “edges”—network connections, in our setting—without obscuring the treemap.

We describe the relevant work done in network visualization in section 2, our visualization approach in section 3, NetGrok’s user interface in section 4, the back-end infrastructure in section 5, a short case study in section 6, future work in section 7, and our conclusions in section 8. The NetGrok source code and a demonstration video can be found at <http://www.cs.umd.edu/projects/netgrok>.

## 2 Related Work

In developing a network traffic visualization tool, a fundamental design question is the following: At what network layer should the traffic be shown? Teoh *et al.* introduced novel ways of representing network routing data [15] for analyzing faults and anomalies, mainly in physical topologies. NetGrok focuses instead on visualizing and interacting with the logical structure of networks at the IP layer: what IP addresses are interacting, ignoring pass-throughs and infrastructure carrying the IP packets.

Cheswick *et al.* presented one of the first large scale, static network visualizations: the first force-directed map of the Internet [4]. This map shows all connections between Internet routers, which unfortunately obscures the topology of the network. Cheswick *et al.* also color coded the map by IP address; the first three octets of the IP are red, green, and blue color values. Although this visualization is useful for understanding a network’s structure as a snapshot, it is not interactive. For larger networks, where many hosts may look visually similar, analyzing such network graphs yields little discovery, unless, of course, the user has an intimate understanding of the underlying data set.

Girardin proposed visualizing static network data through the use of self-organizing maps for attack detection [8]. Girardin’s work mapped multi-dimensional data onto a 2D map using an artificial neural network. Unfortunately, the layout of the hosts varies with each run, forcing users to re-acquaint themselves spatially with the network. Moreover, the algorithm is computationally intensive and designed for static data, making it challenging to use in real-time.

Herman and Melan presented a survey of the known techniques for visualizing networks [11]. Their work thoroughly characterizes the state of network visualization at the end of the twentieth century. We adapted two of the techniques mentioned in this paper to be used by NetGrok: the popular force directed network graph [7], and treemaps [13].

Like the work of Ball *et al.*, NetGrok focuses on viewing the network from a home-centric perspective by segregating local and external hosts [1]. Ball *et al.* provide stable internal and external host layouts through the use of grid patterns derived from the hosts’ IP addresses. This has the drawback of sacrificing significant screen space to show hosts that do not exist. NetGrok achieves efficient space usage by using a force directed approach for the internal hosts, with a hashed layout for external hosts. Host clustering reduces visual instability for



internal nodes, and increases the information content embedded in the external node layout. The host anchoring feature also supports visual consistency.

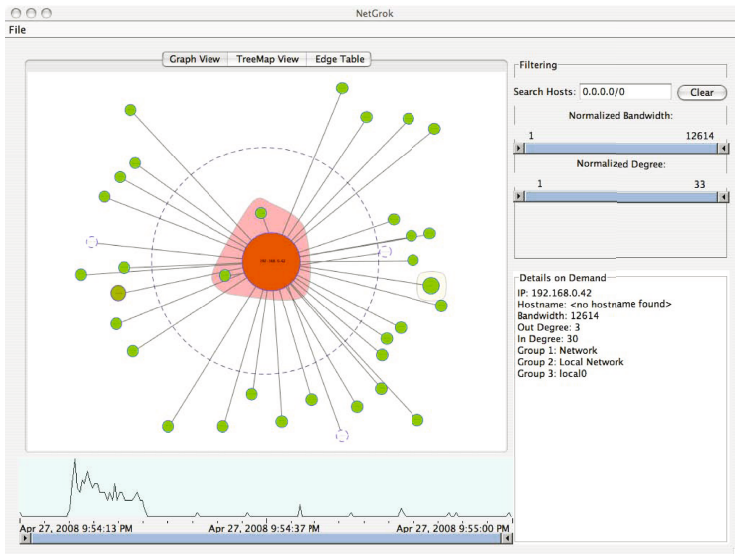
There are a handful of free and commercially available network administration-focused visualization tools. These include: The Multi Router Traffic Grapher (MRTG) by Tobi Oetiker (available as a free download), QRadar from Q1 Labs (commercially available), and several others. These tools provide traditional statistics visualizations, including: pie charts, line graphs and histograms.

### 3 Visualizations

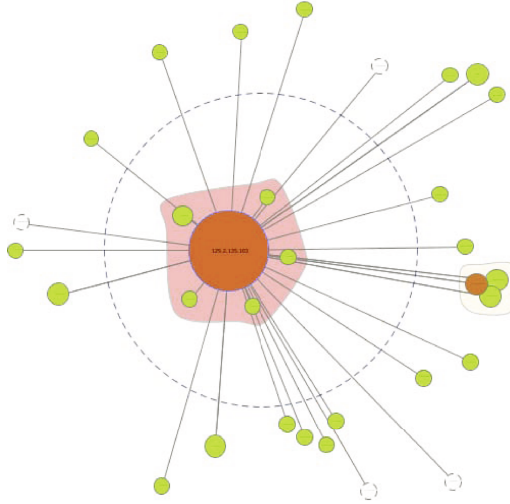
NetGrok's main features are two visualizations: a network graph and a treemap. Both of these visualizations capture: 1. IP hosts, 2. the hosts' bandwidth usage, and 3. links between hosts. We implemented both visualizations using the Prefuse visualization library [10]. These visualizations are found in the upper left portion of the overall interface that can be seen in Figure 1.

#### 3.1 Network Graph

We now discuss the features of NetGrok's network graph and how they aid in finding patterns in network traffic, and developing familiarity with a particular network.



**Fig. 1.** NetGrok's visual elements include a main visualization (upper left), a time-line histogram (lower left), a filter and search panel (upper right), and a details on demand window (lower right)



**Fig. 2.** NetGrok's Network Graph Visualization

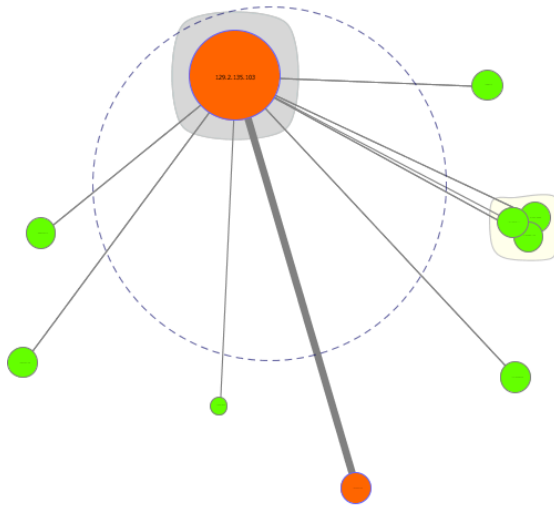
**Grouping Hosts.** As shown in Figure 2, NetGrok's network graph groups nodes in two ways: through their placement in or out of a large dashed ring, and through convex hulls.

Traffic collected on a typical Internet-connected network will likely have far more foreign hosts represented than local hosts. To prevent local network hosts from becoming lost among an overwhelming number of foreign hosts, the network graph contains a dashed ring. This ring serves as a boundary with local hosts contained within the ring and foreign hosts outside, providing a home-centric view of the network [1].

All hosts belonging to user-defined groups are encompassed by a uniquely colored, semi-transparent convex hull. This technique allows users to visually identify the logical groupings of hosts in the network and manipulate entire groupings of hosts simultaneously. The google.com group can be seen on the right side of Figure 2.

**Navigation.** An important feature of NetGrok is its ability to allow users to navigate the vast amount of real-time network traffic in the visualizations. In the network graph, this is done mainly through zooming and panning; both of which are useful for exploring groups of hosts. A user can, for example, zoom in on a group to see all members of the group simultaneously. A user may zoom in on an item by double-clicking on it, or by scrolling with the scroll wheel. Double-clicking the item a second time resets the zoom to encompass the entire graph. The zoom can also be reset by right-clicking anywhere on the graph and selecting the "Reset Zoom" option.

As internal hosts are laid out using the force directed algorithm, the hosts will move when new hosts are added. To overcome this problem, users arrange the



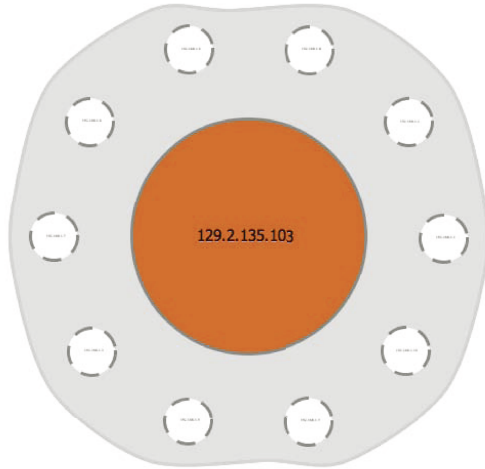
**Fig. 3.** This graph shows group, host, and edge characteristics. The large node in the center is a local host in the Wireless group. A download between the two dark nodes is evident in the image. The group on the right contains hosts in the google.com domain.

internal hosts as they wish by fixing hosts or groups to their desired location on the graph. This also allows the user to arrange the internal hosts to their pleasure.

**Host and Connection Characteristics.** Every IP host on the network is represented in the graph as a colored node labeled with the host's IP address. The label's size is proportional to the size of the host, so that it does not occupy more space on the graph than the host itself. A host's size is proportional to the number of unique hosts it communicates with. For example, in Figure 2, the large node within the dashed ring has by far the most number of connections.

Hosts are colored by their bandwidth usage. Colors range from green to red, with green hosts utilizing the least bandwidth and red hosts utilizing the most. NetGrok uses binned host colors. Host colors are assigned relative to the most bandwidth consuming host in the network. Similarly, there are binned sizes for hosts, assigned relative to a host's degree. Figure 3 shows host and edge characteristics.

We define *zero-byte hosts*, shown in Figure 4, as the hosts that have received IP traffic, but have not sent any IP traffic. Although these hosts might not have sent any data, it is possible that they do not exist at all. For example, a ping sweep is a scanning technique for finding active IP addresses by sending out ping packets to a range of IP addresses, and waiting for a response from active hosts. Since many of these addresses are not active, a ping sweep of a local subnet might yield far more zero-byte hosts than responses. Figure 4 shows how a ping sweep of a local network might look in NetGrok. Since there is no visual way to



**Fig. 4.** An example of what a ping sweep looks like in the network graph. The large, dark circle in the center is performing a scan and the smaller white circles are zero byte hosts.

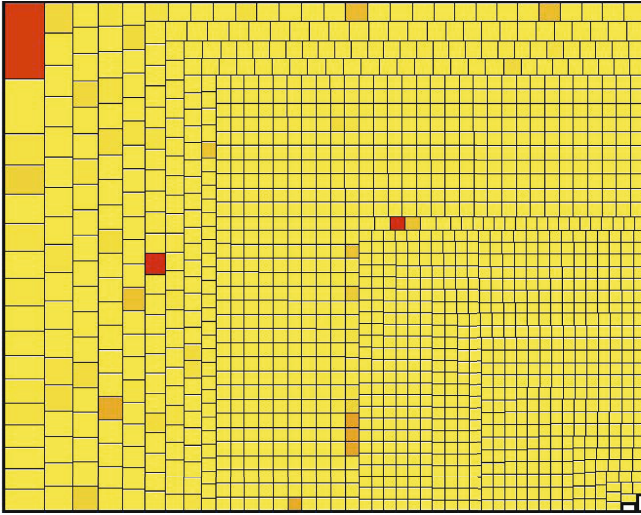
distinguish between low-degree hosts and zero-byte hosts, the visualization treats them as a special case, coloring them white and giving them a dashed border.

Every connection between unique hosts is represented as an edge in the network graph. Because the sheer number of edges in the graph can obscure the visualization, only the edges to or from the host under the mouse are shown. There are options to always show edges to or from certain nodes, and to show all edges on the entire graph. Edges thickness depends on the number of bytes transferred between two hosts. To aid in differentiation of edge thickness only three values were chosen.

**Discovering Abnormalities.** Developing a familiarity with the graph of a network helps to reduce the work required to discover network abnormalities. A user that has become familiar with a network graph in which, for example, a large red backup server appears every night, is likely to notice when the server fails to appear. The ability to easily recognize network hosts is an important factor in the developing of graph familiarity. Host recognizability is based on a host's size, color, and, most importantly, position on the graph. The more consistent these characteristics are, the more recognizable the host will be to users.

Since local network hosts have well-defined grouping, established network roles, and a dedicated area at the center of the graph, they are far more recognizable than foreign hosts for which users have far less information. To increase the recognizability of foreign hosts and the spatial stability of the graph, foreign hosts are always placed in the same location on every graph.

The foreign host layout algorithm hashes the host's IP, creates rectangular coordinates from the halves of the hash, and then converts them into polar



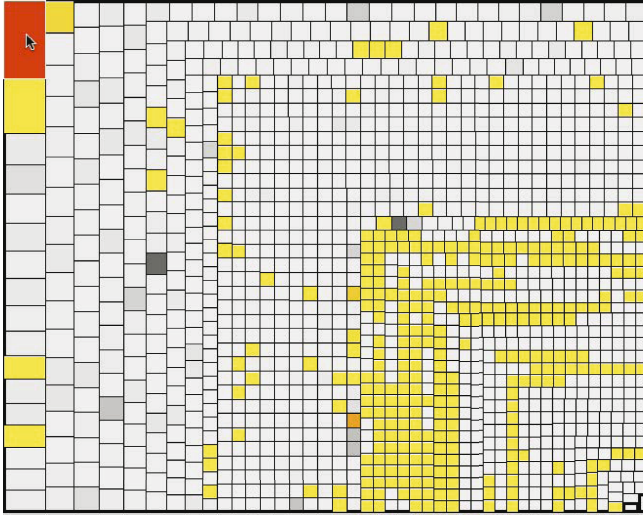
**Fig. 5.** A treemap visualization generated from the OSDI dataset. The dataset was anonymized so groupings are not possible.

coordinates. The resulting coordinate is plotted in an exponential polar plot,  $(e^{\rho-1}, \theta)$  as the new coordinates where  $\rho$  and  $\theta$  are set from parts of the hashed IP [6]. This layout pushes the points away from the center while keeping them from being bunched up at any particular radius. This algorithm guarantees that the same foreign host will always occupy the same position on any network graph. Users can override this position by fixing hosts to a desired position on the graph.

### 3.2 Treemap

To augment the network graph NetGrok also offers a treemap visualization [13]. Just like the network graph, the treemap shows: IP hosts, their bandwidth usage and connections, as well as groups and links between nodes. Treemaps complement network graphs as they can handle considerably more nodes, without occlusion, than the network graph, and they layout nodes using all of the available space. However, unlike the network graph the layout algorithm does not consistently place a host in the same location. Figure 5 shows a treemap with many hosts, generated from the OSDI 2006 dataset [3].

**Host Characteristics.** The treemap is structured using the squarified treemap algorithm [2]. The tree is organized with hosts as leaf nodes and groups as internal nodes. The size of a host in the treemap indicates the number of connections to it, while color denotes the host's bandwidth usage, relative to the other hosts.



**Fig. 6.** Showing links in a treemap

To allow for display of large networks, hosts do not contain any labels. Instead, mousing over a host reveals this information in the details on demand window.

**Showing links in a treemap.** A treemap can coherently show more hosts than the network graph, but a basic treemap does not show links between nodes. Fekete *et al.* showed that links between nodes in a treemap can be overlaid on top of the treemap [5]. Unfortunately, placing links on top of nodes can occlude the nodes underlying the links. When there is high link density, even the links themselves can be occluded. We take advantage NetGrok’s interactive nature to show links in the treemap.

Figure 6 shows how we take advantage of mouse roll-over and color to quickly view a host’s connections. Hovering over a host activates the link browsing view of the treemap. The hosts that are not connected to the selected host will change from color to black and white. Only the hosts that are connected to the hovered over host will be in color. This technique allows the users to distinguish between connected and unconnected hosts, while still being able to see the network structure. When using a treemap, users first look for nodes that are larger and a darker color. We expect users will want to inspect the connections for these hosts.

## 4 Interface

In this section, we present the other elements of NetGrok’s user interface that assist network administrators in exploring their real-time or static network data.

NetGrok provides a Swing-based GUI written in Java using the prefuse visualization toolkit [10]. The NetGrok user interface (Figure 1) provides a consistent

set of controls to manipulate the data model. In the current implementation, these controls provide three methods to filter the hosts that are displayed in the visualizations.

IP prefix filtering and bandwidth and degree sliders only filter on hosts, while the time-line tool filters both hosts and links. There are many additional ways to filter links and host, and the NetGrok framework is easily extensible for adding such filters in the future.

#### 4.1 IP Prefix Filtering

IP filtering is based on classless inter-domain routing (CIDR) prefixes<sup>1</sup>. The user is presented with a text box in the filter pane, shown in Figure 7, that they can use to input a CIDR prefix to filter by. IP filtering is helpful for focusing on resource usage in contiguous IP blocks.

#### 4.2 Bandwidth and Degree Filtering

Bandwidth and degree filtering are simple and effective methods for tailoring the data displayed in the visualizations to answer particular questions. In network traces, host behavior can be very different at opposite ends of the bandwidth and degree spectra. Large classes of hosts can be quickly eliminated from the view by using NetGrok's bandwidth and degree filtering. Hosts are filtered by specifying bandwidth and degree ranges using sliders in the filter pane. The sliders allow users to interactively filter the hosts that are displayed by selecting a range or eliminating extremes in host bandwidth and degree. These controls can be seen in Figure 7. Filters affect only the visualization of the data. Their effects are seen immediately and are completely reversible.

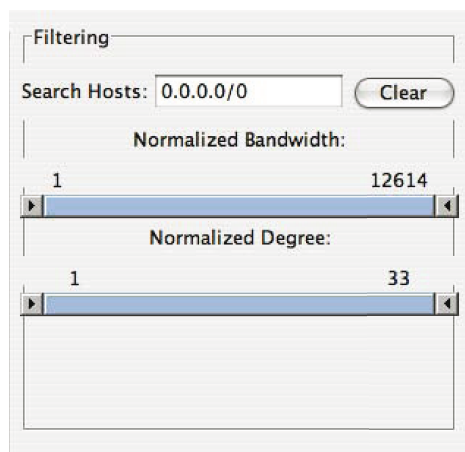
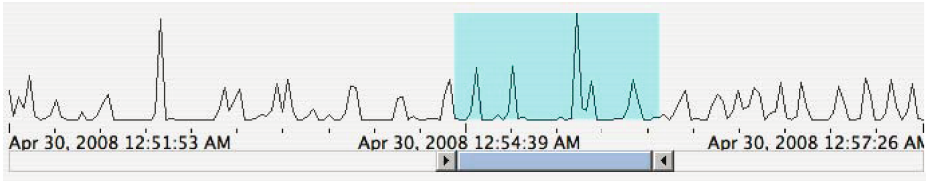


Fig. 7. NetGrok's filter panel

<sup>1</sup> For example, 172.26.0.0/16 represents IP addresses between 172.26.0.0 and 172.26.255.255.



**Fig. 8.** NetGrok’s time-line histogram. This histogram shows a history of hundreds of packets collected between 12:51:53 and 12:57:26 on April 30, 2008. The highlighted region indicates a zoomed-in region of time that is used as a filter for the other NetGrok visualizations. This region can be moved, broadened, or narrowed by using the double-block slider shown.

### 4.3 Time-Line Histogram Filtering

NetGrok provides a time-line histogram and time-range slider filtering option, as well. This filter can be seen in Figure 8. The time-line histogram filter has two functions: it displays an overview of the collected packets over time, and it allows users to zoom in to a particular time frame of interest. This histogram automatically updates as data flows into the program, displaying the entire time range of known packets. Like all of the visualization aspects of NetGrok, the time-line histogram required significant design effort to deal with displaying real-time streaming data.

**Histogram Update Algorithm.** Constructing an efficient algorithm to update the histogram online is technically challenging. We require the histogram to cover a time range that includes all of the packets in the visualization. However, covering too much time diminishes the efficiency and accuracy of the histogram. We also want the histogram to maintain a roughly constant number of buckets in which to aggregate packet counts. When NetGrok starts, the time range that will need to eventually be covered is unknown. As new packets arrive, that time range grows into the future to include the new packet. For accuracy throughout the execution of NetGrok, the range of the histogram must dynamically match the range of packet times seen. Balancing efficiency and accuracy required the creation of a histogram updating algorithm.

NetGrok never removes packets from its data set, hence one can assume that the histogram only needs to grow, and never to shrink. The histogram can therefore effectively adapt to a growing time range by using a pair of grow operations: one to grow back in time, and one to grow forwards in time. The forward growth algorithm is as follows: When a packet is received that has a time greater than the histogram covers, NetGrok finds an integral factor by which to grow the width of the histogram so that it will cover the new point. For example, if we have a histogram that covers 10:00 to 11:00, and we see a new packet from 11:30, we will grow the histogram forwards in time by a factor of 2, and the new histogram will cover the time 10:00 to 12:00. The new histogram has the same number of buckets as the old histogram. During the growth, all of the



buckets of the previous histogram are dumped into the appropriate buckets of the new histogram. With an expansion factor of  $n$ , we will see  $n$  buckets from the previous histogram combined to form a single bucket in the first portion of the new histogram. In this translation, we see no loss of accuracy on the bucket boundaries. NetGrok uses a similar algorithm to grow the histogram backwards in time, as it may need to do when loading historical data from a file.

Because of the dynamic updating algorithm, the histogram may cover a time period wider than the time range of collected packets. This can be up to a factor of 4 in the worst case, but generally is no more than a factor of 2. NetGrok does not display this entire range, but instead only the range of the packets seen. Only the part of the histogram from the first seen packet to the last seen packet is drawn. The number of buckets in the histogram must be sized accordingly, so enough precision is given at a zoom factor of four, and the curve is still relatively smooth when showing every bucket. When the grow operations are called, users see the view zoom out slightly to include the new range, and the histogram smoothes slightly because the bucket width increases.

**Filtering With the Timeline Histogram.** The time-line includes a double-block range slider that allows users to select a temporal region of interest, and only see traffic from that time period. This resembles the technique used by Girardin *et al.* [9], as well as many other projects since then. When users move the range slider, the highlighted period of the histogram updates to reflect the selected time period, and the other visualizations filter out data that is not relevant to that time period.

The time range slider has two modes: forensic mode and real-time mode. If a time period that does not include the latest packets is selected, the time slider will be in forensic mode. In forensic mode, NetGrok does not change the selected period when new packets arrive. When observing a historical time period for forensic reasons, users most likely do not want that time period changing. This can lead to the highlighted time period walking away from the position of the double-block slider. When users move the slider again, the highlighted time period will again line up with the slider. If the selected range includes the latest packets, the time slider will be in real-time mode. In real-time mode, the selected time window is updated when new packets arrive to include new packets that are further in the future.

## 5 Infrastructure

### 5.1 Back-End Data Model

NetGrok uses the `prefuse.data.Graph` object as its primary data store [10], and this is augmented by several auxiliary data sets. The program stores simplified versions of collected packets, and a `prefuse.data.Tree` structure holds the hierarchical set of groups. The data store also keeps a collection of basic statistics for normalizing visualizations. Lastly, the NetGrok data store has basic support for

brushing and linking with shared filter and selection resources. These data structures take advantage of prefuse data models, where possible, for performance. The data store manages the data and exposes the containing structures to the visualization components.

For dynamic updates, NetGrok components can register to be notified when new packets arrive and when graph attributes are modified. This is particularly important for atomic real-time updating of data with multiple visualization threads. This also allows NetGrok's back-end data model to support batch updating, where newly added data may not be accurate until the entire batch is loaded. Large data sets and high bandwidth flows rely on batch updating for efficiency.

## 5.2 Network Packet Collection

NetGrok provides a variety of options for importing data. First, a user can directly connect to a local network interface and stream packets from that interface. For a network administrator, local interfaces may not have access to wide enough coverage of the network traffic. Thus, NetGrok also supports a remote sensor system, where a set of remote network sniffers forward batches of simplified packets to NetGrok, in real-time. In addition to live capture options, NetGrok supports reading the standard PCAP packet capture file format. It also has support for reading and saving data to a more compact format that disregards packet data not used in the visualization.

## 5.3 Group Configuration

The groups.ini file defines the hierarchical groups used in NetGrok's visualizations. This file allows the user to specify which IP addresses and ranges are associated with each group (see Figure 9). The file is divided into two main groups: local and foreign. Each line contains the group name followed by any number of CIDR prefixes to represent networks. Initial group location can be specified by two additional x and y parameters, as shown for the Slashdot foreign group with location (100,200). If the location isn't specified, it is automatically set to the default root polar projection location for the first CIDR prefix as described in Section 3.1.

```
[local]
UMD CP=128.8.0.0/16
local 0=192.168.0.0/24
local 1=192.168.1.0/24

[foreign]
Slashdot=66.35.250.55/24=100=200
Google=216.239.0.0/16,64.233.0.0/16,64.68.0.0/16
```

Fig. 9. Sample groups.ini file

## 6 Evaluation

To evaluate NetGrok, we performed a single subject case study. Our subject, Brad Plecs, is the network administrator for the University of Maryland Computer Science Department. Mr. Plecs’s network monitoring is “usually in response to a right-now network-is-broken situation,” but he is interested in network visualization tools to aid with attack detection. His normal usage of network monitoring tools is to find “interesting” hosts, meaning IP addresses that are using a “disproportionate” share of bandwidth or number of connections. Mr. Plecs’s current tools are a combination of `grep`, `tcpdump`, and “occasionally,” `Ethereal/Wireshark`.

### 6.1 Hands-on Sessions

The study consisted of a short demonstration of NetGrok and two hands-on sessions in which the subject used NetGrok to analyze PCAP packet traces, and a live network. In the first session the subject used NetGrok to analyze PCAP traces of the wireless network at the 2006 OSDI Conference (available from CRAWDDAD [3]). The purpose of this session was to familiarize the subject with NetGrok by finding interesting features in the data. This session went quickly because the interesting features—disproportionate bandwidth and connections—were easy to locate with both visualizations’ use of color gradients and node size. “Zero-byte” hosts also received a positive response.

Mr. Plecs gave two criticisms after the first session. First, that NetGrok needs to incorporate transport layer information, specifically, TCP and UDP port numbers: “[NetGrok] lets you find IP addresses easily, but next you want to see what they’re doing.”<sup>2</sup> His second criticism was that there should be a division between incoming and outgoing bandwidth, to make investigating zero-bytes hosts easier, and for more fine-tuned filtering.

For the second session, Mr. Plecs gathered live traffic from a segment of the UMD Computer Science network which hosts the department’s PlanetLab [12] nodes. As soon as NetGrok started showing traffic in the network view, the subject said the external host placement animation caused confusion. The hosts appear in the center of the display then they slingshot to their final positions in the external network area. When large internal hosts are near the start location, it visually implies communication between the new external host and the existing internal host.

Mr. Plecs quickly found the time slider useful to restrict what was displayed. Initially, he used it to remove the distraction of hundreds of hosts entering the display every second. He responded that—along with the histogram—the feature was useful, allowing users to identify and investigate events that “could easily be missed.” However, because the time slider is so closely tied with the histogram display, Mr. Plecs initially attempted to use the edges of the histogram’s high-

---

<sup>2</sup> TCP and UDP information greatly increases the complexity of network tools. This data, while extremely useful, was simply beyond the scope of the initial project.

lighted time section to modify the filter. He also liked the filter sliders because he could remove the vast number of uninteresting hosts on the live network.

The subject appreciated NetGrok’s representation of the internal network (see Figure 2). Within seconds, he understood the network segment and went on to investigate the external hosts. We think that this is a successful demonstration of applying semantic substrates—in this case, groups—to visualizing computer networks. The subject responded positively to the groups and requested a dynamic grouping ability, so that users can maintain a watch-list or hide hosts on demand and keep the display visually organized without restarting the application and editing the groups configuration. He also noted that having the ability to fix a node’s position helps, but is tedious for tasks like ignoring a large group.

## 6.2 Suggestions for Improvement

Investigating the network through the treemap view was difficult because the treemap does not currently respond to the filtering or time overview and selection. Further, Mr. Plecs indicated that the treemap was unintuitive—obvious patterns in the treemap were difficult to understand, and once understood, indicated uninteresting features.

After the live session, Mr. Plecs said that the “biggest thing” is incorporating transport layer data to give users more information. At the IP layer, NetGrok helps users to discover interesting hosts, but cannot help users investigate exactly why those hosts are interesting.

Mr. Plecs had two other criticisms about the network graph. First, that the animation used to place hosts in the external network is distracting. Initially hosts have a set position in the display, but then they are force-directed to avoid overlapping; this works well with a moderate number of hosts, but when too many hosts are on the graph, it causes constant shaking or jiggling. Mr. Plecs commented that it’s fine for initial placement, but hosts should not continue moving, especially when the traffic displayed is a static time-slice of live capture or packet trace.

Second, Mr. Plecs wanted a more structured layout for the network graph. Hosts are placed via hashing outside the internal network circle so that all of the space on the screen is used. This, however, discards the structure of IP space and isolates hosts that are potentially associated. Consequently, if a certain IP subnet is sending malicious traffic (and it is not specifically grouped), the traffic will visually appear to come from random places in the external network.

## 6.3 Questionnaire

In addition to working with Mr. Plecs’s during the sessions, we provided him with a short questionnaire about NetGrok. For each task discussed in our initial interview, Mr. Plecs was asked to gauge the speed and accuracy of NetGrok when compared to his standard tools. The questionnaire presented a five-option

Likert scale ranging from “much worse” to “much better” to record responses<sup>3</sup>. Mr. Plecs rated NetGrok “much better” for all of the tasks using both PCAP and live data sources.

The questionnaire also included Likert-scaled questions regarding the usability of both the visualizations and user interface components. The scale for these questions ranged from “bad” to “acceptable” to “good.”<sup>4</sup> Mr. Plecs rated the graph visualization and user controls (filtering) “good” in all areas, including organization, the ability to identify features or details, and filtering. Mr. Plecs rated the treemap visualization mostly “acceptable.”

The last part of the questionnaire asked Mr. Plecs to rate NetGrok’s ability to aid in detecting problems and monitoring networks. Mr. Plecs rated NetGrok “fair” on monitoring because “[monitoring tools] should be able to run automatically and notify you when [they] detect something,” but rated NetGrok “good” in the areas of problem detection. He noted: “NetGrok is excellent as a real-time diagnostic.”

## 7 Future Work

One future direction for NetGrok is to rapidly prototype additional real-time network visualizations. NetGrok has a modular construction, so new visualizations can be added without repeating the back-end development effort. The current version displays two primary visualizations, but there are countless others we can potentially implement.

NetGrok’s filtering capabilities offer a good first solution to displaying large datasets, but these capabilities need additional work. For NetGrok to truly support large scale data sets, it needs to contain an abstract overview capability that displays the gist of the data set without showing every node. Many network visualization tools support graph clustering algorithms and cluster-based views of the network. However, extending these clustering and visualization techniques to real-time data flows is not trivial. NetGrok has the foundations for a real-time clustered visualization, which can be seen in the group visualization. However, more effort is needed to develop this into a complete, large scale overview capability.

The network graph was created under the assumption that there would be a manageable number of hosts in the local network and a force-directed algorithm would convey more information about the relationships between local hosts. However, the force direction destroys the spatial stability, which disorients users. In the future a hashed layout algorithm, like the foreign host layout, should be applied to local hosts.

The current network view visualization consistently lays out non-local hosts in the same location. However, that location is essentially randomly generated. We think that there is room for improvement by using semantic substrates [14]

<sup>3</sup> The actual choices for the first section of the questionnaire were: “much worse,” “worse,” “equivalent,” “better,” and “much better.”

<sup>4</sup> The actual choices for the second section of the questionnaire were: “bad,” “poor,” “acceptable,” “fair,” and “good.”

to meaningfully arrange hosts. The challenge of using a semantic substrate for IP address layouts lies in grouping addresses from the same domain, while illustrating one or two attributes of those addresses. Although we expect this is possible, we did not find an appropriate set of attributes to generate a useful semantic substrate to integrate into our visualization, and we must leave this as future work.

## 8 Conclusion

The visualizations, filtering, and details on demand, provide network administrators with the capability to identify network scans, and hosts using a disproportionate amount of network resources. The case study with our expert subject shows that NetGrok's approach holds promise. The study also confirms that NetGrok's network graph and treemap assist in rapid identification of high bandwidth and degree hosts, as well as ping sweeps. The study gave us significant insight into which features need to be enhanced to create a production quality tool for network administration visualization.

## Acknowledgments

We thank Brad Plecs, our case study subject, for lending his time and expertise to help evaluate NetGrok. We also thank the reviewers and Dave Levin for their thoughtful comments.

## References

1. Ball, R., Fink, G.A., North, C.: Home-centric visualization of network traffic for security administration. In: *VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pp. 55–64. ACM, New York (2004)
2. Bruls, M., Huizing, K., van Wijk, J.: Squarified treemaps. In: *Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG 2000)*, pp. 33–42. IEEE Press, Los Alamitos (2000)
3. Chandra, R., Mahajan, R., Padmanabhan, V., Zhang, M.: CRAWDAD data set microsoft/osdi2006 (v. 2007-05-23) (May 2007), <http://crawdad.cs.dartmouth.edu/microsoft/osdi2006>
4. Cheswick, B., Burch, H., Branigan, S.: Mapping and visualizing the internet. In: *ATEC 2000: Proceedings of the annual conference on USENIX Annual Technical Conference*, San Diego, California, p. 1. USENIX Association (2000)
5. Fekete, J.-D., Wang, D., Dang, N., Aris, A., Plaisant, C.: Overlaying graph links on treemaps. In: *Information Visualization Symposium Poster Compendium*, pp. 82–83. IEEE, Los Alamitos (2003)
6. Fink, G., North, C.: Root polar layout of internet address data for security administration. In: *IEEE Workshop on Visualization for Computer Security, 2005 (VizSEC 2005)*, 26 October 2005, pp. 55–64 (2005)

7. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Software - Practice and Experience* 21(11), 1129–1164 (1991)
8. Girardin, L.: An eye on network intruder-administrator shootouts. In: *Proceedings of the Workshop on Intrusion Detection and Network Monitoring (ID 1999)*, Berkeley, CA, USA, pp. 19–28. USENIX Association (1999)
9. Girardin, L., Brodbeck, D.: A visual approach for monitoring logs. In: *LISA 1998: Proceedings of the 12th USENIX conference on System administration*, Berkeley, CA, USA, pp. 299–308. USENIX Association (1998)
10. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: *CHI 2005: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430. ACM, New York (2005)
11. Herman, I., Melancon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics* 06(1), 24–43 (2000)
12. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.* 33(1), 59–64 (2003)
13. Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.* 11(1), 92–99 (1992)
14. Shneiderman, B., Aris, A.: Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics* 12(5), 733–740 (2006)
15. Teoh, S.T., Ma, K.-L., Wu, S.: A visual exploration process for the analysis of internet routing data. In: *Visualization, 2003. VIS 2003, 24-24 October 2003*, pp. 523–530. IEEE, Los Alamitos (2003)

# Wireless Cyber Assets Discovery Visualization

Kenneth Prole, John R. Goodall, Anita D. D'Amico, and Jason K. Kopylec

Secure Decisions division of Applied Visions, Inc.

6 Bayview Avenue, Northport, NY 11768

{kennyp, johng, anitad, jasonk}@securedecisions.avi.com

**Abstract.** As wireless networking has become near ubiquitous, the ability to discover, identify, and locate mobile cyber assets over time is becoming increasingly important to information security auditors, penetration testers, and network administrators. We describe a new prototype called MeerCAT (Mobile Cyber Asset Tracks) for visualizing wireless assets, including their location, security attributes, and relationships. This paper highlights our latest iteration of our prototype for visual analysis of wireless asset data, including user requirements and the various coordinated visualizations.

**Keywords:** Visual analytics, wireless discovery, wireless security, coordinated views, geographic visualization, information visualization, wardriving.

## 1 Introduction

Wireless networks (WiFi, 802.11 protocols) are becoming increasingly more prevalent. Wireless network devices and cards are cheap; most laptops and even desktops are coming with wireless network interface cards preinstalled. While traditional wired networks can be secured from external attackers through firewalls, intrusion prevention systems, and the like, wireless networks open an entirely new kind of security threat for network administrators. Even organizations that do not have a wireless infrastructure are susceptible to wireless attacks. Unwitting end-users can open to outsiders an otherwise secure internal network by simply turning on their wireless cards while connected to the wired network, providing a bridge for malicious outsiders to access the wired network. Attackers can breach wireless networks to steal bandwidth, capture sensitive data, or attack and gain control of computers on both wireless and wired networks. Wireless security vulnerabilities have been gaining media attention. For example, the Wall Street Journal reported that the worst reported security breach of credit card data, which resulted in at least 45 million stolen credit and debit card numbers from TJX's retail stores, stemmed from wardriving and weak wireless encryption keys [1]. These types of occurrences are leading toward the enforcement of various compliance standards, such as PCI DSS [3], the mandated security program created by Visa and MasterCard for their merchants and service providers to safeguard credit cardholder information.

To combat this new threat, security professionals have turned towards tools that attempt to discover, identify, and locate wireless transmitters. This can help them pinpoint rogue access points – wireless transmitters that act as a bridge between the



wireless and wired networks – that are setup by attackers to sniff wireless traffic and hijack legitimate users’ wireless communications. In other cases, wireless users inadvertently plug in access points without any security measures enabled, leaving them completely open to attack and misuse. Attackers can easily identify access points with weak or no encryption.

There are several commonly used free tools for wireless discovery, such as NetStumbler [4] and Kismet [5]. NetStumbler is an active wireless discovery tool for Windows. Kismet is a passive wireless sniffer for Linux and Unix; it is often used for wardriving, and can save GPS location-based data in addition to information related to each of the wireless transmitters it detects. Because it listens for all wireless communication passively, there are some kinds of traffic – such as access points that do not broadcast their name to the world, an increasingly common setting, which Kismet can capture.

Security professionals have some limited visual tools (e.g. GPSMAP, which is included with Kismet) for presenting the results of a single wardrive, but there are no widely adopted visual analysis tools for performing the analysis of many wireless discovery sessions. Without these tools, security professionals report difficulty in detecting changes in the wireless threats over time or geographic region.

Our prototype system, called MeerCAT (Mobile Cyber Asset Tracks), is designed to provide a visual analytic tool for analysis of wireless discovery data. It visualizes wireless transmitter locations, their security attributes, and the relationships among transmitters. We currently use Kismet as our data source, but intend on extending our prototype to visualize NetStumbler data as well.

Our goal is to support the analytic process of information security auditors, penetration testers, and network administrators after performing a wardrive or site survey. To do so, we have incorporated both information and geographic visualizations into a visual analytics system that security practitioners can use for post hoc, interactive analysis of wireless discovery data.

## 2 Related Work

Hurley [6] identifies and describes two of the most popular tools for visualizing wireless discovery results, GPSMAP and StumbVerter. GPSMAP provides various features, including travel path and interpolated signal ranges. GPSMAP is a command line tool and does not provide interactive analysis of the data collected.

StumbVerter [7] is a wireless visualization tool that relies on Microsoft’s MapPoint mapping library. It plots wireless transmitters on a street map using size and color to denote signal strength and encryption mode. It lacks signal range mapping and it does not appear to provide imagery data.

Other popular wireless visualization tools include KNSGEM [8] and Kismet Earth [9], which convert discovery log files into 3D plots in Google Earth. One of the limitations of this approach is that the visualization is constrained to the Google Earth framework; it cannot be embedded in a custom application that adds additional visual displays.

A collaborative effort led by University of Kansas, performed a three-year study tracking statistics on wireless market’s growth, vendor saturation, and security attributes [11]. Using ESRI’s ArcGIS and gathered data, they generated various wireless visualization

maps, which show signal propagation of an access point and its potential security risks. Dartmouth College has also performed extensive research on the various existing wireless visualization and also presented techniques for generating wireless visualization coverage map [12].

Lacking in existing wireless security visualizations is the ability to perform comparisons over time, visual interactivity (most were static images), and difficulty in accessing background imagery. Most importantly, these tools lack visual analytic capabilities for parsing through the copious amount of data to find the most interesting information.

### 3 User Requirements

We interviewed several potential commercial and military users to determine the requirements for a wireless visualization system. This group included information security auditors, penetration testers, and network administrators. This section highlights the results of these interviews.

Although wardriving does not provide the fidelity of a Wireless Intrusion Detection System (WIDS), many users find it is the best solution for performing ad hoc security audits and for covering large areas, such as military bases or college campuses, due to the low cost and ease of setup. A WIDS normally requires a large number of costly sensors to be installed throughout the monitoring area to attain full coverage.

Many of the users we met with perform periodic security audits, from daily to quarterly. During these audits they are looking for rogue, misconfigured, or suspicious devices, such as probing transmitters or ad hoc networks. *Probe networks* represent clients trying to join a network; this could also indicate an active probe performing reconnaissance of the wireless area. *Ad hoc networks* are peer-to-peer networks in which computers can discover and communicate without involving a central access point; there are inherent security issues with ad hoc networks [14].

The first step for many users to assess their current wireless security state is to perform a baseline wireless discovery. During this process, devices are compared with a list of known devices and configurations. Many organizations set security configuration policies in which devices are checked against, such as encryption requirements or SSID naming convention.

Unknown devices are analyzed to determine if they are a friendly neighbor or a rogue device. These rogue devices are further analyzed, normally starting with determining the most likely location. The location area of uncertainty will be needed if in-depth network monitoring is to be done. Having easily accessible detailed geographic imagery was also deemed important for the users we interviewed. This is required when trying to pinpoint the location of devices for remediation.

For known devices, analysts want to understand the signal leakage associated with the transmitters. Ideally, analysts would like the network range not to protrude beyond their building or campus perimeter, therefore reducing their security risk. Analysts also look for overlapping channels from neighboring devices, which may interfere with the availability and reliability of the networks.

When performing follow-up wardrives, analysts frequently performed the arduous task of manually comparing device configurations against their baseline data. Analysts currently have no tools to visually analyze the data collected, especially when it comes to comparing changes over time. Having the ability to quickly filter and group discovery data was also valued, yet lacking in existing tools.

A driving force to these security audits is the various compliance standards being mandated by government (DoD Directive Number 8100.2) [15], healthcare (HIPAA) [16], retailers (PCI DSS), and various other markets. To accommodate these new requirements, reporting is becoming an important requirement in wireless auditing, not just for compliance reporting, but also as a way to report to management and collaborate with others.

We found that many users find great value in having a visualization to describe the current state of devices of interest to others. Having a simple picture makes the intricacies of wireless network security easy to describe to less technical savvy people. These reports are general desired in both texture and visual representation and in various formats, including PDF, PowerPoint, Word, and e-mail.

As we iteratively design and implement our system, we continue to work with our identified user groups to elicit ongoing feedback that is incorporated in future iterations.

## 4 Coordinated Views for Wireless Security Analysis

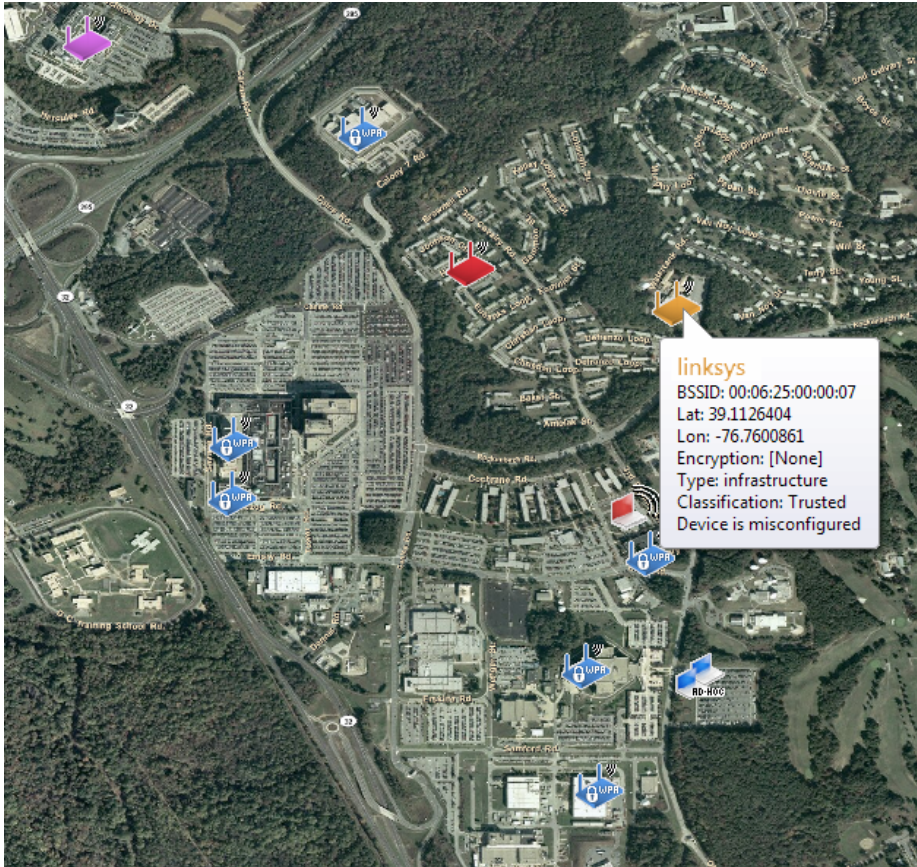
The primary visualization in MeerCAT is the two dimensional geographic visualization as shown in Fig. 1. The background imagery is provided by ESRI's ArcGIS Online repository [17], which provides 1 meter or better aerial imagery for the contiguous United States and satellite imagery for the world at 500-meter and 15-meter resolutions, as well as detailed street map data.

This screenshot shows the 11 devices that were detected during a wardrive. The device icon shows whether it is an access point or a client computer. Although color is configurable, in this case it is used to indicate classification of devices: blue for trusted, red for rogue or probe, purple for friendly, and orange for misconfigured. Encrypted devices show a lock symbol and the level of encryption is shown on the icon itself, either WPA (strong encryption) or WEP (weak encryption).

Some interesting items can be quickly spotted looking at the 2D geographic view. We can see many trusted assets (in blue), one ad hoc network (two computers side-by-side), one probing client (red laptop), and one rogue access point (red). The misconfigured device in orange with an SSID of "linksys" is indicating that a known device was found to be in a configuration other than what was expected. In this case the security policy states this device should have WPA encryption enabled, but the device was detected with no encryption. This device can be annotated and flagged as something to watch during follow-up wireless surveys.

MeerCAT contains multiple views as shown in Fig. 2. These views are linked together with the same data to provide an interactive visual analytic environment; highlighting or filtering in one view is reflected in the others. The following highlights each of these views:

- *Device Tree* (Fig. 2a): A hierarchy of the detection runs (an individual wardrive) and the wireless transmitters and clients discovered during each run.



**Fig. 1.** 2D geographic visualization

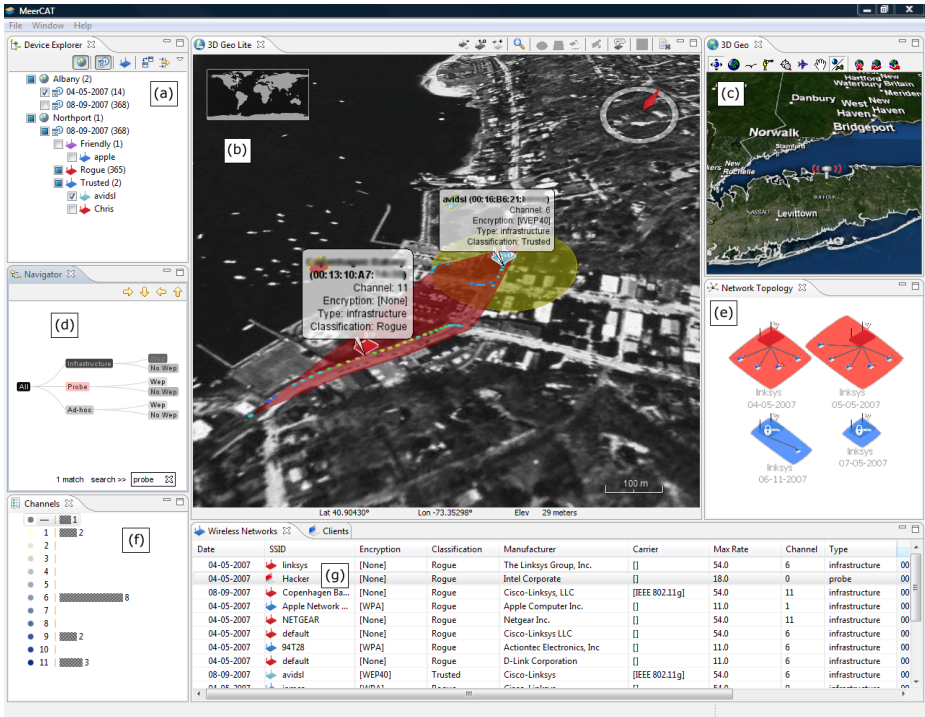
Devices can be sorted, filtered, and grouped in various ways to help analyst quickly find the information they are looking for.

- *Geographic Visualizations* (Fig. 2b, c): Two and three-dimensional geographic fly-through visualizations showing satellite imagery locating the discovered wireless transmitters. These views allow for displaying tooltips, popup captions, signal ranges, drive path, and attached clients.
- *Device Visualization* (Fig. 2d): A tree visualization organizing the discovered wireless transmitters according to their type, encryption, and connected clients, colored according to the relative number of packets collected in that branch of the tree.
- *Network Visualization* (Fig. 2e): A graph visualization showing the discovered access points and clients that are connected to them. This view uses small multiples [18] to show a given network's change over time as shown. In the figure, the same device is shown in four different wardrives (shown by the dates under the device name, linksys); this device changed between wireless surveys, from unencrypted (red without lock) to encrypted (blue with lock).

This view can also be useful for quickly identifying networks with many clients connected to it.

- *Channel Visualization* (Fig. 2f): A histogram showing selected transmitters channel distribution. This provides a color legend when using geographic range displays for analyzing signal propagation.
- *Detail Tables* (Fig. 2g): A tabular display showing the details of a selected wireless transmitter and clients, allowing for sorting on columns of interest.

One of the use cases we have designed for is the need to see changes over time. When performing follow-up detection runs, analysts can use MeerCAT’s various features to perform this type of temporal analysis. These include the ability to filter the device list to only show items that have changed one or more attributes, such as encryption, SSID, channel, type, or if the location moved a certain number of feet. The table view allows analysts to iterate through an individual device’s history, causing the other views to update accordingly. The small multiples network visualization can quickly show how networks evolve and change between wardrives.



**Fig. 2.** MeerCAT wireless cyber asset discovery visualization prototype. (a) Device Tree list of detected devices; (b, c) Geographic Visualizations showing location of devices; (d) Device Visualization showing transmitters by type and encryption; (e) Network Visualization showing connections between transmitters; (f) Channel Visualization showing channel distributions; and (g) Details Table showing details of networks and clients.

## 5 Implementation

MeerCAT is implemented in Java using the Eclipse Rich Client Platform (RCP) [19] and Standard Widget Toolkit (SWT) [20] to provide cross-platform support with a native look-and-feel. A cross-platform solution was required as the users we interviewed depended on both Windows and Linux for their data collection and analysis.

We are currently using ESRI's ArcGIS [21] and NASA's WorldWind [22] for geographic visualizations and the open-source prefuse toolkit [23] for information visualizations. For data processing, we are currently using Oracle's TopLink Essentials [24] implementation of the Java Persistence API (JPA) and H2 [25] as our embedded database repository.

## 6 Conclusion and Future Work

The ability to discover, identify, and locate wireless transmitters is an increasingly crucial aspect of information security. To facilitate the analysis of wireless discovery data, we have developed a prototype visual analytic tool to enable security practitioners to easily understand the attributes, relationships, and locations of wireless transmitters. This prototype is intended to demonstrate the utility of the system and garner early feedback from security practitioners.

We plan to incorporate visualizations that depict communication patterns, derived from packet capture data collected by network discovery tools. We will also plan on incorporating reporting features, the ability to display wired network topology in addition to wireless, and in-building (floor plan) visualizations. Finally, we will be bringing our prototype to security practitioners to solicit feedback that will be incorporated into successive iterations.

## Acknowledgments

This research and development effort is supported by DARPA Strategic Technologies Office through a Small Business Innovative Research grant, under contract number W31P4Q-07-C-0022.

## References

1. Hole, K., Dyrnes, E., Thorsheim, P.: Securing Wi-Fi Networks. *Computer* 38(7), 28–34 (2005)
2. Pereira, J.: Breaking The Code: How Credit-Card Data Went Out Wireless Door. *Wall Street Journal*, 5/4/07 Issue (2007)
3. PCI Security Standards Council (Accessed 1 June 2008), <https://www.pcisecuritystandards.org>
4. NetStumbler (Accessed 1 June 2008), <http://www.netstumbler.com>
5. Kismet (Accessed 1 June 2008), <http://www.kismetwireless.net>
6. Hurley, C., Thornton, F., Rogers, R., Connelly, D., Baker, B.: *WarDriving & Wireless Penetration Testing*, pp. 219–246. Syngress Publishing, Inc. (2007)

7. StumbVerter (Accessed 1 June 2008),  
<http://www.sonar-security.com/sv.html>
8. KNSGEM (Accessed 1 June 2008), <http://www.rjpi.com/knsgem.htm>
9. Kismet Earth (Accessed 1 June 2008),  
<http://www.niquille.com/kismet-earth>
10. Bittau, A.: WiFi Exposed, Crossroads, vol. 11(1), p. 3. ACM Press, New York (2004)
11. Wireless Network Visualization Project (Accessed 1 June 2008),  
<http://www.ittc.ku.edu/wlan>
12. Lentz, C.: 802.11b Wireless Network Visualization and Radiowave Propagation Modeling, Dartmouth College Technical Report TR2003-451 (2003)
13. Connelly, C., Liu, Y., Bulwinkle, D., Miller, A., Bobbitt, I.: A Toolkit for Automatically Constructing Outdoor Radio Maps. In: International Conference on Information Technology: Coding and Computing (ITCC 2005), vol. II, pp. 248–253 (2005)
14. Zhou, L., Zygmunt, H.: Securing Ad Hoc Networks, IEEE Networks Special Issue on Network Security, Cornell University, Ithaca (1999)
15. Department of Defense Directive Number 8100.2 (Accessed 1 June 2008),  
<http://www.dtic.mil/dticasd/sbir/sbir041/srch/n076.pdf>
16. HIPAA Security Standard (Accessed 1 June 2008),  
<http://www.cms.hhs.gov/SecurityStandard>
17. ArcGIS Online (Accessed 1 June 2008),  
<http://www.esri.com/software/arcgis/arcgisonline>
18. Tufte, E.: Envisioning Information, pp. 67–79. Graphics Press (1990)
19. Eclipse Rich Client Platform (Accessed 1 June 2008), <http://eclipse.org/rcp>
20. The Standard Widget Toolkit (SWT) (Accessed 1 June 2008),  
<http://eclipse.org/swt>
21. ArcGIS (Accessed 1 June 2008), <http://www.esri.com/software/arcgis>
22. NASA World Wind (Accessed 1 June 2008), <http://worldwind.arc.nasa.gov>
23. Heer, J., Card, S.K., Landay, J.A.: Prefuse: A Toolkit For Interactive Information Visualization. In: ACM Conference on Human Factors in Computing Systems (CHI), pp. 421–430. ACM Press, New York (2005)
24. Oracle TopLink Essentials JPA (Accessed 1 June 2008), <http://www.oracle.com/technology/products/ias/toplink/jpa/index.html>
25. H2 Database Engine (Accessed 1 June 2008), <http://www.h2database.com>

# NetFlow Data Visualization Based on Graphs

Pavel Minarik<sup>1</sup> and Tomas Dymacek<sup>2</sup>

<sup>1</sup> Institute of Computer Science, Masaryk University  
Botanická 68a, 602 00 Brno, Czech Republic  
pavel.minarik@mail.muni.cz

<sup>2</sup> Mycroft Mind, Inc.  
Lidická 28, 602 00 Brno, Czech Republic  
dym@mycroftmind.com

**Abstract.** We present an innovative approach to NetFlow data processing and visualization developed at Masaryk University in Brno. Our visualization method based on graphs bridges the gap between highly aggregated information visualization represented by charts and too much detailed information represented by the log files. In our visualization method the graph nodes stand for network devices and oriented edges represent communication between these devices. We also present the utilization of external data sources (DNS, port names, etc.), which helps to present NetFlow data in more intuitive way. Hence this approach is very natural one for both network administrators and non-specialists. Based on these methods a proof-of-concept tool called *NetFlow Visualizer* has been developed and is now offered as an plug-in for the NetFlow probes.

**Keywords:** visualization, visual analytics, NetFlow data, graphs.

## 1 Introduction

The usage of NetFlow data [1] in the network monitoring domain is growing. Available tools for NetFlow data processing such as *NFSen* [2] are equipped with large scale visualization represented by charts (see figure 1) on the one hand and very detailed visualization represented by lists of NetFlow data log files on the other (see figure 2).

We believe that these methods are not sufficient for the analysis of network traffic and that there is a gap between these two methods. Charts may give the analyst the whole picture of the situation in the network. Listing log files gives the analyst all the details. However, with thousands of logged connections, it is extremely complicated to process this data, particularly when the analyst does not know exactly what he/she is looking for.

We are therefore introducing a visualization method of NetFlow data based on graphs (see figure 3) which can supply chart-based visualizations. Our method focuses on network devices (graph nodes) and communication between these devices (oriented edges) aggregated on different levels. This scalable level of detail (level of aggregation) is suitable for the analysis of network traffic where



the analyst is able to see the whole picture of the situation on the network and is able to focus on every single data transfer (flow) at once.

It is also very important to do as much mechanical work for the analyst as possible. The other concept addresses the utilization of external data sources. The key idea is to provide additional information (domain names, port information, etc.) to help the analyst during the process of data analysis. We should note there that this is the piece of information which the analyst seeks manually when working with the NetFlow log files.

## 2 Related Work

There are several approaches which use graph-based visualization in the network monitoring domain.

*Interactive Network Active-traffic Visualization (INAV)* is a monitoring solution for use in real-time network environments. It monitors the traffic currently active between nodes [3].

*Netview* is a graph-based network visualization tool that displays an animated realtime view of the network. Traffic noticed by the *netview-server* is classified and aggregated and in turn animated by the *netview-client* [4].

*tcpdump – a network packet capture library* provides real-time decomposition and graph-based visualization of network traffic [5].

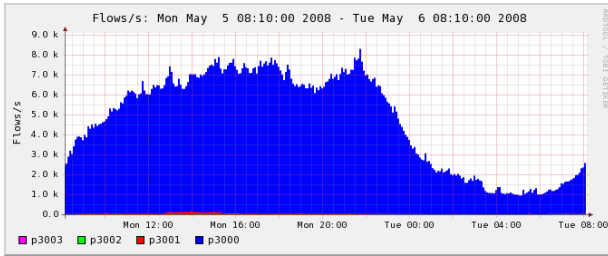


Fig. 1. Chart (based on NetFlow data) visualization example

```

** sfdump -M /data/sfres/profiles-data/Live/p3001 -T -r 2008/05/05/sfdump.200805052010 -a -X srcip,dstip,dstport -o long -50
sfdump filter:
any
Date flow start          Duration Proto      Src IP Addr:Port      Dst IP Addr:Port      Flags Top  Packets      Bytes Flows
2008-05-05 20:09:58.181  1.945  0           17 26 0 ->          5 37 80 .A...  0  18  720  9
2008-05-05 20:09:58.028  0.138  0           35 30 0 ->          5 37 80 .A...  0  2  256  2
2008-05-05 20:09:58.131  1.923  0           209 170 0 ->         5 37 80 .A.P.S.  0  18  1705  3
2008-05-05 20:09:58.282  0.700  0           149 16 0 ->          5 37 80 .A.P.F  0  29  2262  2
2008-05-05 20:09:58.338  0.015  0           5 37 0 ->          7 12 59338 .A..F  0  26  33500  1
2008-05-05 20:09:57.987  1.098  0           5 37 0 ->         209 170 7897 .A.P.F  0  26  30396  3
2008-05-05 20:09:58.594  0.000  0           5 37 0 ->         153 203 28428 .A..F  0  4  160  2
2008-05-05 20:09:58.653  0.000  0           5 37 0 ->         153 203 28638 .A..S.  0  2  96  1
2008-05-05 20:09:58.676  0.000  0           5 37 0 ->         153 203 28640 .A..S.  0  2  96  1
2008-05-05 20:09:58.677  0.037  0           5 37 0 ->         153 203 28641 .A..S.  0  4  938  1
2008-05-05 20:09:57.603  0.145  0           5 37 0 ->         209 170 7704 .A.P..F  0  14  18528  1
2008-05-05 20:09:58.149  0.000  0           5 37 0 ->         153 203 28637 .A..S.  0  2  96  1
2008-05-05 20:09:58.594  1.631  0           5 37 80 .A.P.F  0  21  4097  7
2008-05-05 20:09:58.331  1.902  0           72 210 0 ->          5 37 80 .A...  0  24  2173  4
2008-05-05 20:09:57.586  2.290  0           5 37 0 ->         17 26 4272 .A....  0  88  124960  2
2008-05-05 20:09:59.002  0.000  0           5 37 0 ->         5 101 2514 .A..S.  0  2  120  1
2008-05-05 20:09:59.032  1.165  0           5 37 0 ->         209 170 10586 .A.P.F  0  30  6844  3
2008-05-05 20:09:58.643  0.090  0           5 37 0 ->         153 203 28636 .A..S.  0  6  1104  1
2008-05-05 20:09:55.649  1.730  0           20 19 0 ->          5 37 80 .A.P..  0  10  400  1
2008-05-05 20:09:52.923  0.000  0           5 37 0 ->         5 37 80 .A...  0  2  104  1
2008-05-05 20:09:59.002  1.278  0           5 101 0 ->          5 37 80 .A..S.  0  3  172  2
2008-05-05 20:09:58.513  0.699  0           5 37 0 ->         149 16 52981 .A.P.F  0  34  35224  1
Summary total flows: 90, total bytes: 324311, total packets: 387, avg bps: 352656, avg pps: 52, avg kpps: 938
Time window: 2008-05-05 20:09:51 - 2008-05-05 20:14:57
Total flows processed: 807, records skipped: 0, bytes read: 417036
Type 0:02s (flows/second): 2680797.2  null: 0.037s (flows/second): 211834.5
    
```

Fig. 2. NetFlow log file listing example

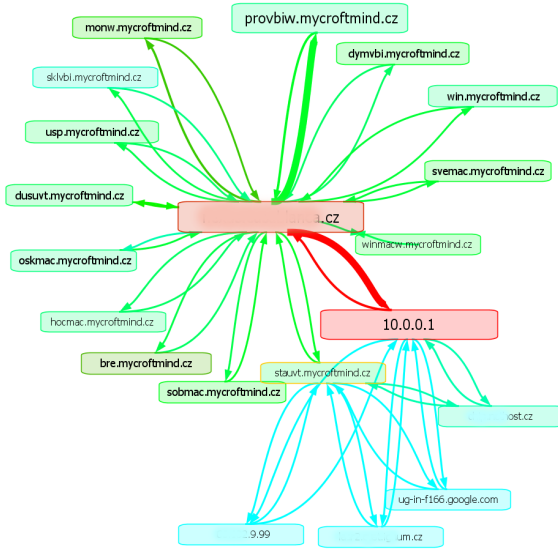


Fig. 3. Visualization of NetFlow data based on graphs example

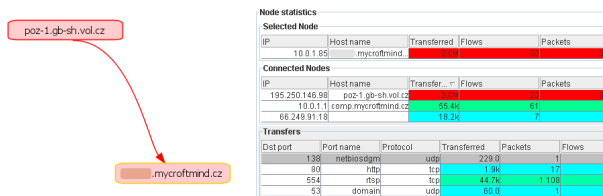
All of these solutions use a similar visualization method but they have a different purpose to our approach. They are used for real-time network traffic monitoring and do not support different time slots comparison and analysis. Therefore only a limited number of traffic attributes is processed. Filtering possibilities are also not sufficient. They do not provide WHOIS information or additional port information.

### 3 Visualization Method Properties

Motivation for our work is the research and development agenda for visual analytics to facilitate advanced analytical insight. This agenda was presented by the *National Visualization and Analytics Center* in the book called *Illuminating the Path* [6].

Our visualization method reflects the recommendations presented in [6]. The main properties of our visualization method may be characterized by the following points (only fundamental functionality related to visualization is presented):

- **Graph-based visualization** (so called dynamic mind map visualization) of the communicating network devices. Visualization method suitable for presenting connections. Edges are oriented according to flow direction.
- **Spreadsheet based visualization** of communication details and statistics.
- **Multiple level of detail** offers communication aggregation between network devices aggregated by protocol, details of the communication aggregated using protocol and destination port or pure NetFlow data visualization.



**Fig. 4.** Example of graph-based visualization complemented with spreadsheet visualization of statistics for selected node

- **Dynamic visualization adjustment** according to actual data. Coloring and sizing of nodes representing network devices and edges representing communication between these devices. Coloring and sizing changes dynamically according to selected attributes of the NetFlow records and their current values (peaks and lows) present in current data. E.g., size of a node corresponds to number of packets transmitted by the node, its color corresponds to amount of transferred data, size of an edge corresponds to number of flows transferred and its color corresponds to the number of packets transferred.
- **User defined visualization adjustment** for selected nodes representing network devices. These user defined devices may be visualized using different node shape or size. Visualization settings are tied to IP addresses.

Presented visualization method is built on *Prefuse* [7] visualization toolkit complemented by standard JAVA components for spreadsheet visualization (see figure 4).

## 4 NetFlow Visualizer

The visualization method presented was implemented by *Mycroft Mind Inc.* [8]. The resulting product is called *NetFlow Visualizer* [9] and is provided with *INVEA-TECH Inc.* [10] *FlowMon probes* [11] as a freeware plug-in. Innovated version of *NetFlow Visualizer* is being developed concurrently.

*NetFlow Visualizer* is the client-part of client-server solution. The server is called *NFSel* and its purpose is to provide NetFlow data. *NFSel* is a part of the *FlowMon probe* and is invisible for users. *NFSel* is a *XML-RPC* server which provides NetFlow data from the NetFlow data collector using a standard tool called *NFDump* [12]. *NFSel* converts this data into a *GraphML* format [13]. Figure 5 is a comprehensive illustration of the architecture.

*NetFlow Visualizer* itself is a client-side Java application which visualizes NetFlow data provided by *NFSel* upon request in *GraphML* format. *NetFlow Visualizer* uses the graph-based visualization method and provides user interface (see figure 6) with additional filtering or search features and external data sources utilization. *NetFlow Visualizer* provides basic filtering possibilities

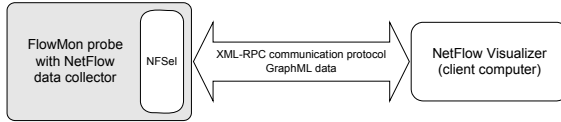


Fig. 5. System architecture overview diagram

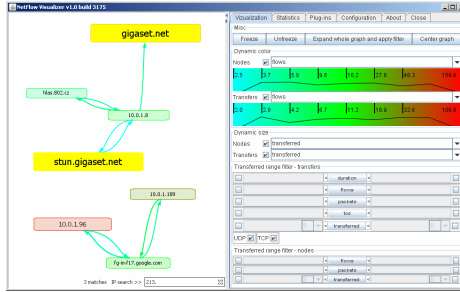


Fig. 6. NetFlow Visualizer tool. Illustration of the visualization properties tab and IP search feature.

utilizing parameters of pure NetFlow data (protocols, amount of transferred bytes, packets, etc.).

Another extension lies in external data sources utilization. The purpose of data sources utilization is to visualize communication on the network more naturally and clearly for the operators. *NetFlow Visualizer* tries to do as much mechanical work for the analyst as possible and so the analyst can focus on his/her work instead of searching for port names or translating IPs into domain names. The following data sources are utilized by *NetFlow Visualizer*:

- **DNS (Domain Name Service)** – Human beings are used to working with names; computers, however, use numeric identifiers. Translating IP addresses into corresponding domain names is crucial especially in large networks. Information about domain name is not present in the NetFlow data. It should therefore be obtained from a proper DNS server online.
- **WHOIS Service** – It is sometimes necessary to search for additional information about a network device, e.g. its location or administrative contact. Direct integration of the WHOIS service saves analyst’s time.
- **Port names** – Even experienced network administrators might not be familiar with uncommon port numbers. The motivation is similar to DNS. Name and description is much more than a number. This data source provides translations from pairs protocol, port into a port name<sup>1</sup> and its description.

<sup>1</sup> A “port name” is just a shortcut for “typical service running on the given port and available via the given transport protocol”.

## 5 Use-Case

In this section we would like to present a simple use-case and compare *NetFlow Visualizer* with the classic approach using *NFDump* [12] and *NFSen* [2] tool. Our use-case will be the exploration of traffic between the top data producers/consumers in the monitored network.

### Use-case procedure using *NFDump*

1. Construct query to obtain top N traffic producers or consumers. Mark the results. Example of corresponding *NFDump* query:  

```
nfdump -M /live/p3000 -T -r nfcapd.200805130405 -n 10 -s ip/bytes
```
2. Construct query to obtain the communication of the devices from the previous step (aggregation using source IP address, destination IP address). Example of corresponding *NFDump* query:  

```
nfdump -M /live/p3000 -T -r nfcapd.200805130405 -a -A srcip,dstip, proto IP XXX.YYY.ZZZ.UUU or IP...
```
3. Construct query to obtain the communication between selected IP addresses (pure data or aggregated by destination port). Example of corresponding *NFDump* query to get pure NetFlow data:  

```
nfdump -M /live/p3000 -T -r nfcapd.200805130405 IP XXX.YYY.ZZZ.UUU or IP...
```

### Use-case procedure using *NFSen*

1. Obtain top N traffic producers or consumers using user interface for top N statistics (see figure 7).
2. Set the filter in user interface for flows listing. Copy IP addresses acquired in the previous step into "Filter" text box. Set aggregation using source IP address, destination IP address and protocol in user interface.
3. Analogous to previous step. Add aggregation using destination port or remove aggregation completely to get pure NetFlow data.

### Use-case procedure using *NetFlow Visualizer*

1. Set time interval and press the "Load data" button, *NFSen* will acquire data from the collector and deliver it to *NetFlow Visualizer*.
2. Set filter "nodes transferred more than" using user interface (see figure 6).
3. Expand with a single mouse click one or all the nodes to obtain the communication of the devices satisfying the filter.



Fig. 7. *NFSen* user interface for top N statistics

The screenshot shows a window titled 'Flow information' containing a table of network flow data. The table has columns for start time, duration, end time, source port, source port name, destination port, destination port name, protocol, transfer, packets, flags, and loss. Below the table are two sections: 'Source port info' and 'Destination port info', each with fields for port, protocol, port name, description, known threats, and additional info.

Start time	duration	end time	Src port	Src port name	Dst port	Dst port name	Protocol	Transf.	Packets	Flags	loss
2007-09-19	0.704	2007-09-19	80	http	56449	56449	TCP	354.4k	252	AP-SF	0
2007-09-19	9.841	2007-09-19	80	http	56445	56445	TCP	163.3k	24	AP-SF	0
2007-09-19	16.682	2007-09-19	80	http	56447	56447	TCP	180.0k	13	AP-SF	0
2007-09-19	15.569	2007-09-19	80	http	56449	56449	TCP	180.0k	13	AP-SF	0
2007-09-19	8.265	2007-09-19	80	http	56449	56449	TCP	11.3k	55	AP-SF	0
2007-09-19	8.126	2007-09-19	80	http	56450	56450	TCP	52.4k	48	AP-SF	0
2007-09-19	2.132	2007-09-19	80	http	56451	56451	TCP	25.9k	21	AP-SF	0
2007-09-19	13.636	2007-09-19	80	http	56452	56452	TCP	24.7k	21	AP-SF	0
2007-09-19	0.0	2007-09-19	80	http	56451	56451	TCP	40.6k	1	A..	0

Source port info		Destination port info	
Port	80	Port	56449
Protocol	tcp	Protocol	tcp
Port name	http	Port name	tcp
Description	World Wide Web HTTP	Description	
Known threats	Back End, Executor, Hooker, RingZero (TCP)	Known threats	
Additional info	<a href="http://seamed.ucsf.edu/projects/80.html">seamed.ucsf.edu/projects/80.html</a> <a href="http://en.wikipedia.org/wiki/80">en.wikipedia.org/wiki/80</a>	Additional info	

Fig. 8. Table of pure NetFlow data (replies of a web server to a client)

- Open with a single mouse click the statistics tab for selected device to obtain the data aggregated by destination port (see figure 4) or open the edge between two devices to obtain the pure NetFlow data in a table (see figure 8) using its context menu.

Let's summarize the use case presented. Using *NFDump* or *NFSen* users have to think out and write their own commands or set up a complicated filter. It is clear that using a tool such as *NetFlow Visualizer* can increase dramatically the productivity of labour of network analysts and make NetFlow data analysis accessible even to non-specialists.

## 6 Conclusion and Future Work

The NetFlow data visualization method based on graphs was created, presented, evaluated and discussed with network and security experts. The main conclusion is that such a method has big potential for complementing existing methods and is very useful for specific use-cases. The fact that this method has been implemented by a commercial company and is being provided to customers as a visualization tool called *NetFlow Visualizer* with NetFlow data probes confirms its usefulness and potential.

The visualization method presented was adapted in the CAMNEP project [14] to provide visualization of anomalous traffic according to detection layer results. Visualization tool *NetFlow Visualizer* was verified at Masaryk University within the framework of the internal security targeted project.

The development of the visualization tool presented and the visualization method itself is not yet finished. One of the biggest challenges is to provide an analyst with the exact portion of information required. Network traffic data are massive and quick insight will not be possible if the analyst is subjected to information overload. It is necessary to provide the analyst with a powerful and intuitive way of defining and expressing his/her actual focus. Another challenge is to permit any number of centers of focus so that users will be able to view details for more than one node at one time. The solution proposed is to combine graph-based and spreadsheet-based visualization in one workspace. The graph nodes will therefore contain spreadsheets with details.

*Acknowledgment.* This material is based partially upon work supported by the European Research Office of the US Army under Contract No. N62558-07-C-0001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the European Research Office of the US Army.

## References

1. Cisco Systems: Cisco IOS NetFlow (2007), <http://www.cisco.com/go/netflow>
2. Haag, P.: NfSen - NetFlow Sensor (2007), <http://nfsen.sourceforge.net>
3. Robinson, N., Scaparra, J.: Interactive Network Active-traffic Visualization (INAV), <http://inav.scaparra.com/docs/whitePapers/INAV.pdf>
4. Cornell University, Department of Computer Science: Netview, <http://netview.gforge.cis.cornell.edu/index.php>
5. Jcap project team: jpcap – a network packet capture library, <http://jpcap.sourceforge.net/>
6. Chinchor, N., Hanrahan, P., Robertson, G., Rose, R.: Illuminating the Path: The Research and Development Agenda for Visual Analytics. National Visualization and Analytics Center (2006)
7. Berkeley Institute of Design: The Prefuse Visualization Toolkit, <http://www.prefuse.org>
8. Mycroft Mind Inc.: Mycroft Mind Inc. Company Profile, <http://www.mycroftmind.com>
9. Mycroft Mind Inc.: NetFlow Visualizer, <http://www.mycroftmind.com/products:nfvis>
10. INVEA-TECH Inc.: INVEA-TECH Inc. Company Profile, <http://www.invea.cz/main/home>
11. Čeleda, P., Kováčik, M., Konří, T., Krmíček, V., Špringl, P., Žádník, M.: FlowMon Probe. Technical Report 31/2006, CESNET, z. s. p. o. (2006), <http://www.cesnet.cz/doc/techzpravy/2006/flowmon-probe>
12. Haag, P.: NFDUMP - NetFlow processing tools (2007), <http://nfdump.sourceforge.net>
13. Graph Drawing Steering Committee: GraphML format, <http://graphml.graphdrawing.org>
14. Agent Technology Group, Gerstner Laboratory, Czech Technical University in Prague and Institute of Computer Science, Masaryk University in Brno: CAM-NEP (Cooperative Adaptive Mechanism for NETwork Protection) project web page, <http://agents.felk.cvut.cz/projects/camnep>

# Backhoe, a Packet Trace and Log Browser

Sergey Bratus, Axel Hansen, Fabio Pellacini, and Anna Shubina

Dartmouth College, NH 03755, USA

**Abstract.** We present *Backhoe*, a tool for browsing packet trace or other event logs that makes it easy to spot “statistical novelties” in the traffic, i.e. changes in the character of frequency distributions of feature values and in mutual relationships between pairs of features. Our visualization uses feature entropy and mutual information displays as either the top-level summary of the dataset or alongside the data. Our tool makes it easy to switch between absolute and conditional metrics, and observe their variations at a glance. We successfully used *Backhoe* for analysis of proprietary protocols.

## 1 Introduction

Analysis of packet traces and event logs derived from packet traces by aggregation or by matching for certain types of events is likely the most frequent task performed by analysts. In particular, *browsing for anomalies*, for points where the traffic changes in character, is a frequent fallback of the analyst or administrator trying to form a hypothesis of what might have happened.

More precisely, users look for groups of records (selected according to some restriction, for example by time intervals) where the distribution of some feature substantially changes (e.g., goes from an almost random uniform distribution to a highly skewed one, or vice versa), or the nature of the mutual relationship between two features substantially changes (e.g., one feature stops being a good predictor for the value of another, or vice versa). Such changes very often have clear security implications; related heuristics are popular in anomaly-detecting intrusion detection systems (IDSes).

Considering that such “novelties” may occur in any one of the usually many features of interest (and in any one of the even more numerous pairwise relationships between these), the users need a tool that lets them examine such relationships at a glance, pivot it easily on selected features, and quickly switch between selections. In this paper, we present *Backhoe*, a proof-of-concept visualization tool for exploring the features’ variability and mutual relationships across slices of data, which builds on the *prefuse* toolkit [5].

Although many tools exist to help with the task of anomaly detection on either static (captured) or dynamically flowing packet traces or event streams, most of these tools use an IDS-style model, trained on “normal” input, for judging records as anomalous and highlight the detected anomalous records in the display. We, on the other hand, build our display itself around information-theoretic metrics and let the user make judgements about the overall character of feature interdependence; thus we are not bound by any particular IDS model.



Our approach is based on information-theoretic metrics (explained later in [3]) and inspired by the multi-strata visuals of *Name Voyager* ([9]). Unlike in *Name Voyager*, however, the mutual positioning of strata in our display of a packet trace or log is determined from the current dataset, and the relative thickness of some layers is strongly determined by others (in fact, understanding such dependencies is likely what the user is after). Accordingly, after the initial layout, the users can manually re-order and otherwise manipulate the strata, as well as switch the entire view into a mode relative to (“conditioned on”) a chosen layer and back.

## 2 Backhoe in Operation

Backhoe’s [4] goal is to help the user make sense of a sequence of packets, or, generally speaking, a sequence of log entries, each multi-field record representing an event. Backhoe relies on some other tool to parse these packets or records into rows of a table, the columns of which correspond to protocol fields or other relevant features. For packets we use *tshark*’s PDML output, for other kinds of records – the output of our *Kerflog* browsing tools [2]. From now on, we will refer to either parsed packets or other parsed log entries as “records” or “sequence elements”. Also, we use “fields” and “features” interchangeably, because at the point when the record (packet) is parsed and represented as a set of key–value pairs, it does not matter much whether the value is extracted directly from the record or computed.

Besides this input sequence, Backhoe needs to know how to partition this record sequence into groups, across which features’ distributions and their statistical relationships could be compared and visualized. In most analysis scenarios where sequence elements are time-stamped, it is natural to divide them by time intervals; a group is made up by all records with the timestamp falling within the interval,  $G_i = \{r : t(r) \in [t_i, t_{i+1}]\}$ . Alternatively, when the user is interested in seeing how much adding new records changes the character of the features’ overall distribution, the groups can be defined as comprised of all records to date,  $G_i = \{r : t(r) \leq t_i\}$ .

However, the splitting of records into groups need not necessarily be done by a timestamp: any features, the values of which could be meaningfully ordered, would do. For example, the user might choose to group packets by size, port numbers, and so forth, or by ranges thereof. In fact, our *Treeview* tool from the above-mentioned Kerf suite attempts to establish the series of features that, when used to recursively group a log dataset, results in groupings that are most convenient for spotting anomalies and classifying “normal” traffic [1].

**Strata mode.** Regardless of how the grouping is accomplished, Backhoe computes a set of features  $\{F_k\}$  for each group and graphs the thickness of stratum  $k$  at point  $i$ , i.e., over the group  $i$  to be  $F_k(G_i)$ . We call the visualization mode

<sup>1</sup> The resulting strata configurations reminded us of cross-sections of exposed rock strata (in the proverbial “mountains of data”), hence the name for the tool that “exposes” these strata.

<sup>2</sup> For a description of our Kerf project, see <http://kerf.cs.dartmouth.edu/>.

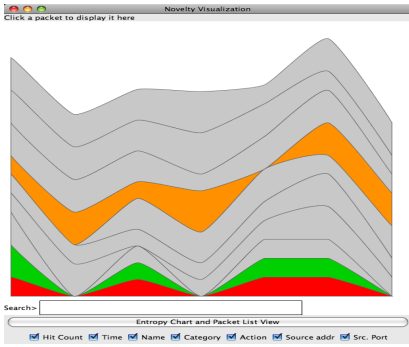


Fig. 1. Strata mode, absolute entropies

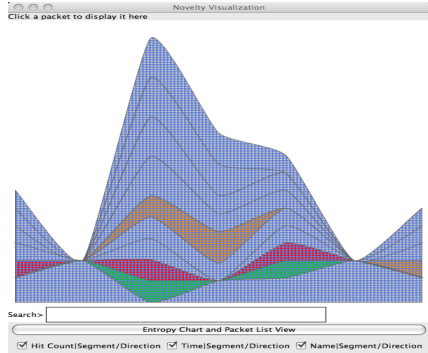


Fig. 2. Strata mode, conditional entropies

in which only this data is displayed *strata mode* (Fig. 10). This strata mode helps the user to zero in on the interesting interval where the relative changes in the thickness of the strata attract the user’s attention.

**Packetfall mode.** The strata mode helps the user choose an initial interval for further exploration. Once the interval is chosen, the user can switch into the mode we call the *Packetfall mode*, inspired by the *Rainfall mode* of the Rumint tool by Greg Conti<sup>3</sup>. In this mode, the packet summaries (or record group summaries for non-packet data) are displayed to the right of the strata diagram, which in turn is rotated (Fig. 3). To find and examine packet summaries of interest, the user can take advantage of the fisheye and search functions, explained below.

**Switching features sets.** The default set  $F_k$  of strata features is  $H(F_k) = \{H(F_k(G_i))\}_{k,i}$ , the entropies of the respective features over the record groups. The rationale for this choice is provided in Section 3. However, the user can switch to other sets with a single keystroke. For example:

- ‘v’ switches the thicknesses of the  $k$ -th stratum to the count of distinct values of the feature  $F_k$  across each group,  $\{\#\{\text{distinct values}(F_k|G_i)\}\}$ , and
- ‘C’ pivots every layer except the selected one  $k_0$  to be the conditional entropy  $H(F_k|F_{k_0}) = \{H(F_k|F_{k_0})|G_i\}_i$ .

We call these “conditional” (Fig. 9, left side) and “distinct values” sets. Strata conditioned on another stratum are painted with a distinctive brush.

We also use custom features calculated from the underlying packets, such as the length of the packet’s payload when compressed as a byte string by a Lempel–Ziv type algorithm (see 4).

**Mixing in conditional strata.** When convenient, the user can mix-and-match conditional layers with absolute ones, choosing the conditioning per layer. This is useful when we know that some feature  $X$  is strongly dependent on  $Y$ , and  $W$  is strongly dependent on  $Z$ , but  $Z$  and  $Y$  are virtually independent, and we want

<sup>3</sup> Available at <http://www.rumint.org/>

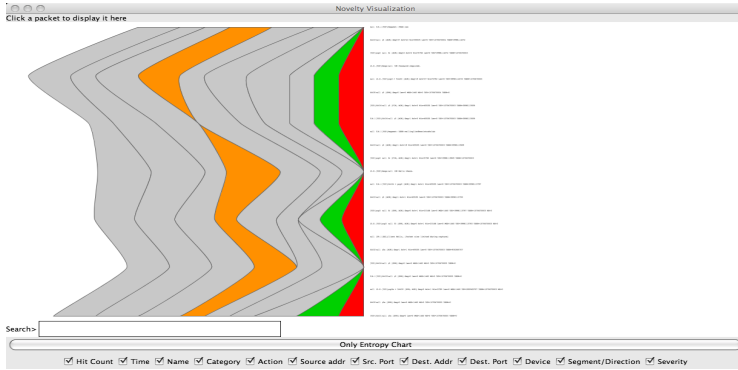


Fig. 3. Packetfall mode, base view

to show this manner of mutual dependencies at a glance and also save screen space. The left side of Fig. 7 provides an example.

With additional Chow–Liu style precomputation (see [2]), the user can choose to see the top  $m$  pairwise correlations (more precisely, the  $m$  feature pairs with the highest mutual information) drawn from the start as conditional layers, showing, at a glance, where these (on average) strongest dependencies are weakened, as an exception to the general trend.<sup>4</sup>

**Strata (re-)ordering.** In the simplest display, the strata are sorted according to their average thickness. The user is given the option of choosing the order and the colors of strata via simple keystroke commands (e.g., ‘d’ lowers the stratum, ‘u’ raises it, ‘f’ toggles colors, ‘h’ hides it altogether). Since reordering operations change the overall layout, their results are animated with fade-in color actions to help the user spot the new position of the just-moved layer.

**Fisheye and search.** In packetfall mode, the records in the right (“packet”) pane are necessarily rendered in tiny fonts or degenerate to pixel lines. Individual records can be viewed by using the fisheye effect (Fig. 4, 6–7). The base (non-fisheye) view allows text query searches on its elements: all matching characters (or pixels, when such squashing is necessary) are highlighted in it, showing the occurrences of the sought substrings (Fig. 5). This highlighting can then be used to zero in on individual matches with fisheye (Fig. 6).

### 3 Why Use Information-Theoretic Metrics?

Entropy of a random variable  $X$  that takes distinct values  $\{x_i, i = 1, \dots, n\}$  with probabilities  $\{p_i, i = 1, \dots, n\}$  is defined as  $H(X) = \sum_{i=1}^n p_i \log \frac{1}{p_i}$ .  $H(X)$  is interpreted as the “information content” of  $X$ , or a *measure of uncertainty* about  $X$ . We are interested in the latter interpretation.

<sup>4</sup> For reasons of space, we omit the details of our layout algorithm and refer the reader to our upcoming technical report.

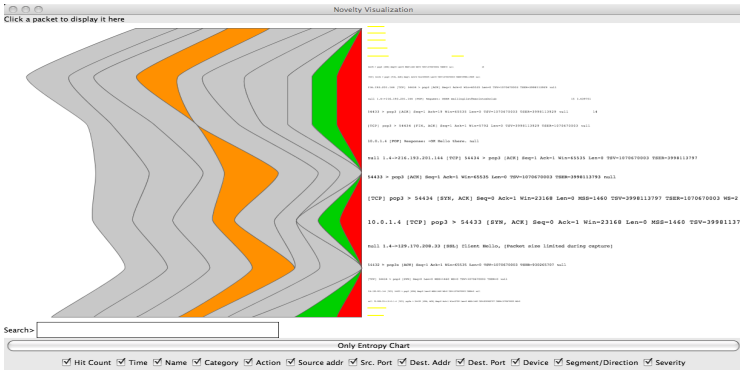


Fig. 4. Packetfall mode, fisheye view

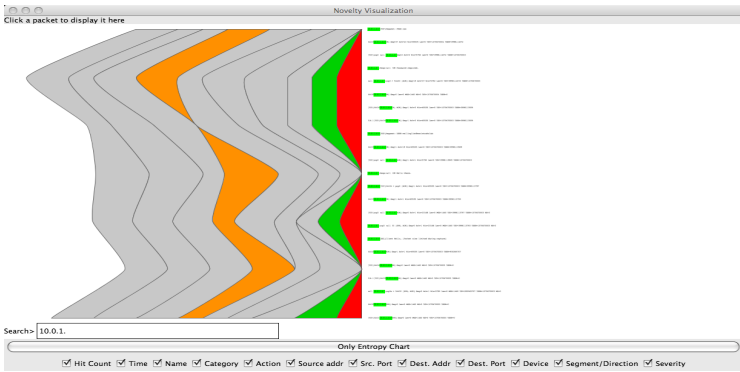


Fig. 5. Packetfall mode, search activated

Consider a set of  $N$  log records with just one field of interest per record. The values occurring in that field form a discrete probability distribution  $X$ , each distinct value  $x_i$  having the probability  $p_i = n_i/N$ , where  $n_i$  is the number of times  $x_i$  occurs in the set. The less certain we are about which of these values you encounter in a record, the higher is the entropy  $H(X)$ .

Even for one-value records entropy can provide important security clues, especially when the expected variation of that value in normal operating conditions is small. A raise in entropy would then signal a change of conditions (see, e.g., Lee et al. [8], [7]).

Entropy of a joint distribution of two variables  $X, Y$  is defined as  $H(X, Y) = \sum_{ij} p_{ij} \log \frac{1}{p_{ij}}$ , where  $p_{ij}$  is the probability that the distinct values  $x_i$  of  $X$  and  $y_j$  of  $Y$  are taken simultaneously (in the simplest case, consider log records with just two fields of interest and the distribution of pairs of values seen in these fields together in a record). For independent  $X$  and  $Y$  we have  $H(X, Y) = H(X) + H(Y)$ ; in this case knowing the value taken by  $X$  tells us nothing about the likelihood of any particular value of  $Y$  co-occurring with it. Whenever

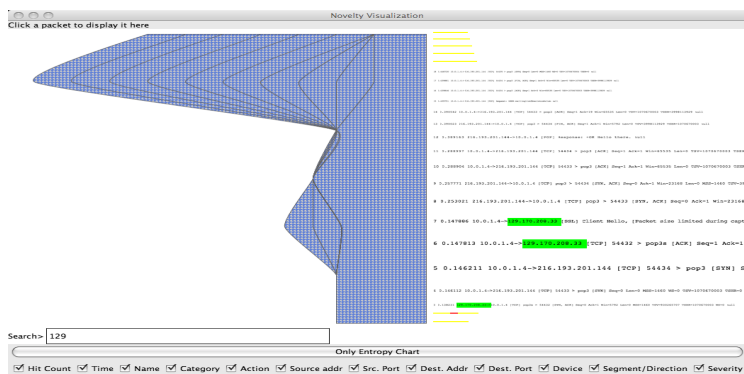


Fig. 6. Packetfall mode, all strata conditioned on `destAddr`, search + fisheye

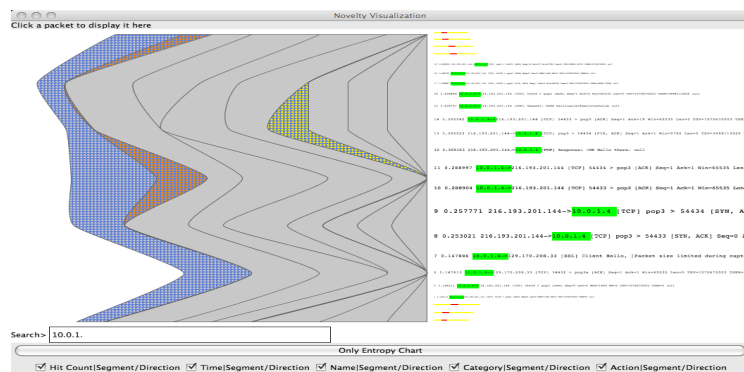


Fig. 7. Mixed mode conditional, search + fisheye

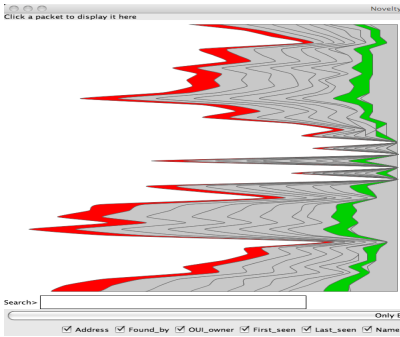
dependence exists, knowing the value of  $X$  does help us predict the value of  $Y$ , through our knowledge of the likelihood of their observed co-occurrence.

Statistics that compare the joint entropy  $H(X, Y)$  with the single feature entropies  $H(X)$  and  $H(Y)$  are very useful for describing dependence between variables. In particular, the *conditional entropy*  $H(X|Y) = H(X, Y) - H(Y)$  is a measure of uncertainty about the value  $X$  when that of  $Y$  is already known.

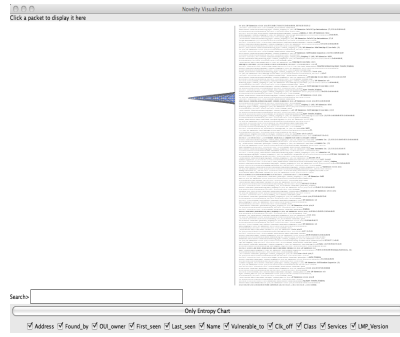
A big increase in  $H(X|Y)$  means that  $Y$  is no longer a good predictor for  $X$ , which has strong security implications for many mutually predictive field pairs of many protocols designed with flexibility in mind but no longer exercising that variability in normal operating environments, such as the use of diverse IP options or fragmentation, overly long variable length headers, and so on.

Unusual use of protocol fields is characteristic of many exploits, but sophisticated attackers take pains to disguise it, as IDSes might be watched for it<sup>5</sup>. It is much harder, however, to disguise unusual payloads in such a way that does not

<sup>5</sup> The “Dissembler” by Jon Erikson provides an interesting example among many.



**Fig. 8.** The BlueSnarfing scenario, en-



**Fig. 9.** BlueSnarfing, conditional entropies

introduce unusual statistical effects in any pair of protocol features. It is these effects that our visualization aims to make conspicuous.

**Scalability** is a significant concern, since computing information-theoretic metrics has high CPU and RAM costs. We are exploring lowering these costs through use of streaming entropy estimation algorithms.

## 4 Evaluation

Backhoe was developed to assist with specific traffic and log analysis tasks. Here we briefly cover two scenarios in which it proved itself.

**Proprietary protocol analysis.** In the course of vulnerability testing a proprietary product, we needed to discover some facts about a proprietary client–server protocol. Luckily, the protocol’s packets were not encrypted, and we could obtain a packet trace. Furthermore, the trace contained many repetitive transaction legs, and we were interested in spotting new kinds among these repetitions.

We configured one of the features computed on the packets to be *the length of the byte string resulting from Lempel–Ziv compression of the packet*, using the string table accumulated so far from compressing previous packets. This feature served as a rough measure of the packet’s “novelty”: packets that consisted mostly of previously seen substrings compressed really well, unlike “novel” packets that started some new types of transactions.

The running stratum of the compression-length feature allowed us to quickly locate different kinds of command packets in the *packetfall* mode, whereas highlighting of search byte strings helped us see the degree of repetition of function codes and their approximate offsets in the packet traces at a glance.

**The BlueSnarfing story.** In another study, a group of Ohio-based security researchers made available to us the logs of proximity scanning of Bluetooth devices (the so-called BlueSnarfing). These logs were collected in the public spaces of security conferences such as Defcon and Notacon. One of the objectives of the analysis was to find out whether any spoofing of Bluetooth device MACs

was actually occurring. The “snarfed” log consisted of records that showed MAC, manufacturer ID and capabilities strings, and other device-specific information.

Having loaded the logs into Backhoe (Fig. 8) and taking *conditional* views, we noticed that the other features were perfectly predictable conditioned on the device MAC address, except for one interval. In that interval, capability and manufacturer strings strata showed non-zero thickness, i.e., non-zero conditional entropy  $H(\cdot|MAC)$ , making it obvious that the relationship between their values was many-to-one (Fig. 9 shows that conditional view with the MAC stratum hidden away for further clarity). This suggested that within that particular time interval some device responses were inconsistent and probably spoofed. Backhoe enabled us to zero in on this property of the log dataset right away.

## 5 Related Work

Most log visualization tools currently available are concerned with ways to represent frequency distributions of the data according to predefined rules. Despite the complexity of selecting the optimal representation from the sheer graphical point of view (as summarized, for example, in 6), these tools frequently fall short in their support for anomaly detection. Another broad summary of security visualization tools is given in 3.

Lee et al. in 8 argued for using information-theoretic measures to build models for anomaly detection in datasets, and Lakhina et al. in 7 demonstrated their usefulness for practical traffic monitoring. Although sophisticated detection methods have been proposed since then (e.g., 4), we are not aware of uses of entropic metrics for visualization proper, although Kaminsky’s *Sequitur* visualizations 6 show the potential of related ideas.

## References

1. Aslam, J., Bratus, S., Pavlu, V.: Semi-supervised data organization for interactive anomaly analysis. In: ICMLA 2006: Proceedings of the 5th International Conference on Machine Learning and Applications, pp. 55–62 (2006)
2. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. In: IEEE Trans. Information Theory, vol. 14, pp. 462–467 (1968)
3. Conti, G.: Security Data Visualization: Graphical Techniques for Network Analysis. No Starch Press (2007)
4. Gu, Y., McCallum, A., Towsley, D.: Detecting anomalies in network traffic using maximum entropy estimation. In: IMC 2005: Proceedings of the 5th ACM SIGCOMM conference on Internet measurement, pp. 1–6 (2005)
5. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: CHI 2005: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 421–430 (2005)

---

<sup>6</sup> See [http://www.doxpara.com/slides/dmk\\_shmoo2007.ppt](http://www.doxpara.com/slides/dmk_shmoo2007.ppt),  
<http://seattle.toorcon.org/2007/talks/dankaminsky.ppt>.

6. Keim, D.A.: Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics* 6(1), 59–78 (2000)
7. Lakhina, A., Crovella, M., Diot, C.: Characterization of network-wide anomalies in traffic flows. In: *IMC 2004: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 201–206 (2004)
8. Lee, W., Xiang, D.: Information-theoretic measures for anomaly detection. In: *Proc. of the 2001 IEEE Symposium on Security and Privacy*, pp. 130–143 (2001)
9. Wattenberg, M.: Baby names, visualization, and social data analysis. In: *INFOVIS 2005: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, p. 1 (2005)



# Existence Plots: A Low-Resolution Time Series for Port Behavior Analysis

Jeff Janies

CERT Network Situational Awareness Group  
4500 Fifth Avenue  
Pittsburgh, PA 15213  
janies@cert.org

**Abstract.** An existence plot is a low-resolution visualization that concurrently represents the activity of all  $2^{16}$  ports on a single host. By doing so, we are able to show patterns of port usage which can indicate server activity and demonstrate scanning. In this work we introduce the existence plot as a visualization and discuss its use in gaining insight into a host's behavior.

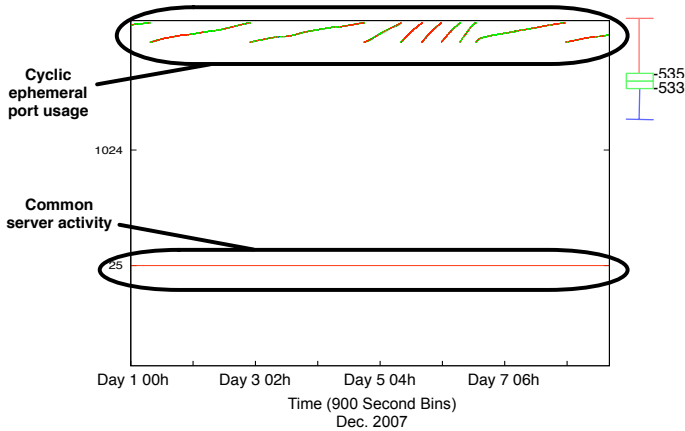
**Keywords:** Network traffic visualization, Low-resolution visualization, Time series.

## 1 Introduction

An *existence plot* is a time-series visualization of traffic over *all* the active ports on a single monitored host. Existence plots summarize activity for a single host in a limited space, regardless of the number of unique sources with which the host communicates. For example, Figure 1 is the existence plot for an SMTP server. Ports are listed on the *y*-axis, while the *x*-axis represents time. The box-and-whisker diagram on the right of the plot shows a color coding of byte magnitude: blue for the 1st quartile, green for the 2nd and 3rd quartiles, and red for the 4th quartile. We also see two families of lines on the plot - a constant, horizontal red line at port 25, indicating the host's SMTP server activity, and a collection of lines with similar slope in the ephemeral port range (1024-65535) indicating client activity.

Existence plots offer useful, high-level summaries of traffic from a particular host, due to their coarse representation of activity. Using the existence plot, we provide a high-level view of *all* activity originating from a host. While we cannot render exact information on the magnitude of activity from a particular port and maintain readability, we note that the majority of network traffic contains noise generated by automated scanning, bots and other hostile activity [1][7][10]. Because of this, it is not unusual for simple magnitude-based indicators to include a large amount of trivial data due to prevalent, but meaningless low-volume interactions caused by hostile activity.

This paper is a tutorial on the construction and use of existence plots. We demonstrate that existence plots provide a method for analysts to rapidly identify aggregate host behavior such as *hidden servers* (hosts that are operating as



**Fig. 1.** Existence plot showing activity from December 1st to 7th, 2007 from an SMTP server

public servers which the administrator may not be aware of), scanning conducted by compromised hosts, and scan response.

The remainder of this paper is structured as follows: §2 describes the composition and construction of existence plots from traffic data; for our examples we use the SiLK toolkit<sup>1</sup>, but plots can also be constructed using raw `tcpdump` data. §3 shows how to interpret results from an existence plot, in particular the identification of hidden servers and scanning activity. §4 discusses other visualizations that represent individual hosts. §5 concludes this work with a discussion of future applications.

## 2 Constructing Plots

In this section, we outline the data requirements and method for generating existence plots, as follows: §2.1 describes our source data and its format, and §2.2 describes the process of plotting formatted source data.

### 2.1 Source Data Format

Existence plots represent a unidirectional count of bytes transferred on individual ports. We format network traffic summaries as a series of values,  $X_{p,\tau}$ , where  $p$  is a port and  $\tau$  represents a time interval. As with other time series,  $\tau$  is a discrete interval of time, in this case measured in seconds. Since we format the data as unidirectional traffic summaries, each host,  $A$ , can be represented with two non-equal data sets, `inbound` and `outbound`, where `inbound` is the set of byte counts of all packets destined to  $A$ , and `outbound` is the set of byte counts of all packets originating from  $A$ .

<sup>1</sup> Available at <http://tools.netsa.cert.org/silk/>

## 2.2 Plotting from Data

For the images in this paper, the  $y$ -axis represents the  $2^{16}$  TCP and UDP ports plotted in log-scale, unless otherwise specified. This representation is a *low-resolution* time-series; that is, instead of a precise delineation of every discrete value (as is the case with MRTG<sup>2</sup>), we bin the values into broad categories to provide a complete view of the data. We denote drastic changes in magnitude with color and are able to compress more information into each image by using the  $y$ -axis to represent unique variables instead of a shared scale measuring magnitude.

$$color(p, t) = \begin{cases} none & X_{p,\tau} = 0 \\ blue & 0 < X_{p,\tau} < S_0 \\ green & S_0 \leq X_{p,\tau} \leq S_1 \\ red & S_1 < X_{p,\tau} \end{cases} \quad (1)$$

Equation 1 shows the mapping of magnitude to colors using the data-dependent values:  $X_{p,\tau}$ ,  $S_0$  and  $S_1$ . We set  $S_0$  and  $S_1$  as the 1st and 3rd quartile of all non-zero values of  $X_{p,\tau}$  (*i.e.* the values change per data set). However, alternative approaches are viable: for example, if the magnitude of traffic is predictable,  $S_0$  and  $S_1$  can be fixed across images to provide consistency.

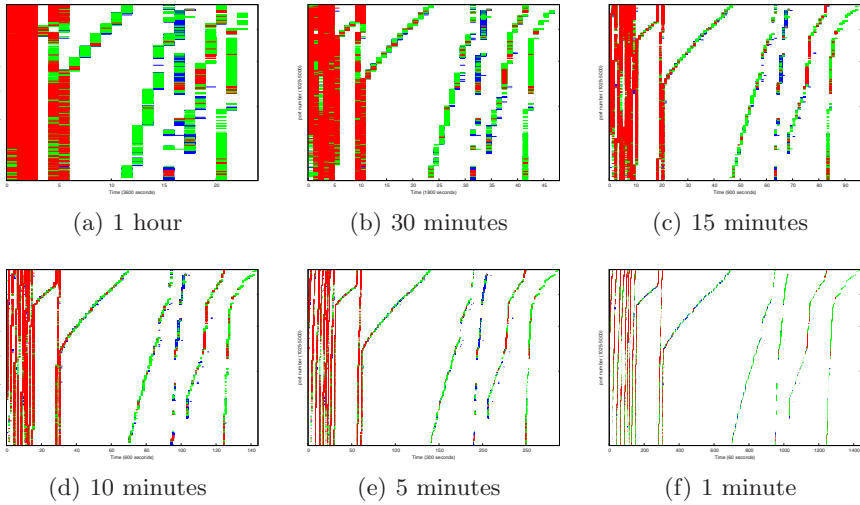
With the existence plot, we are equally interested in periods of activity *and* inactivity. By representing both, patterns emerge. The most common ephemeral port usage pattern is a series of lines with similar slopes, depicting *port cycling* in client interaction (*i.e.* the host sequentially uses a set of ports in a finite range, and this sequence is repeated). Servers consistently use ports common to the service they provide. This results in a horizontal line of activity.

Figure 2 shows how variations in the size of  $\tau$  affect the shapes in existence plots. In this figure, we represent one day of **outbound** traffic from ports 1025 through 5000 of a frequently used Microsoft Windows machine, which comprises 32,100 NetFlow records. We vary  $\tau$  sizes to be 1 hour, 30 minutes, 15 minutes, 10 minutes, 5 minutes and 1 minute and only display the port range 1025 through 5000. With the largest  $\tau$  size of one hour, only the port cycles with a longer period are discernible. As the  $\tau$  size decreases the shorter period port cycles become discernible. At 10 minutes the lines are distinct; therefore, we use a resolution of 10 minutes for a majority of the images in this paper.

## 3 Interpreting Plot Data

Existence plots display aggregate port activity. In this section, we demonstrate how this aggregated view can be used to identify various phenomena, specifically *hidden* servers, scanning done by an internal host and scan response. This section is divided as follows: §3.1 explains how existence plots can be used to identify hidden servers. §3.2 shows how existence plots can portray scans.

<sup>2</sup> See <http://oss.oetiker.ch/mrtg/>



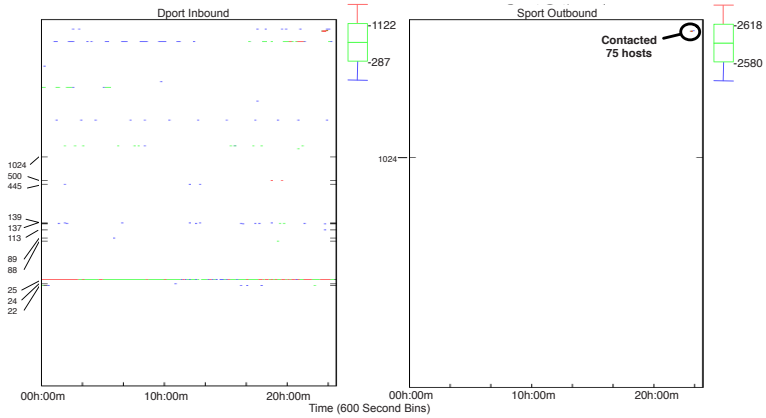
**Fig. 2.** The  $x$ -axis represents time and the  $y$ -axis represents the port range 1025 through 5000. As the size of  $\tau$  decreases the port cycles become more distinct.

### 3.1 Hidden Server Identification

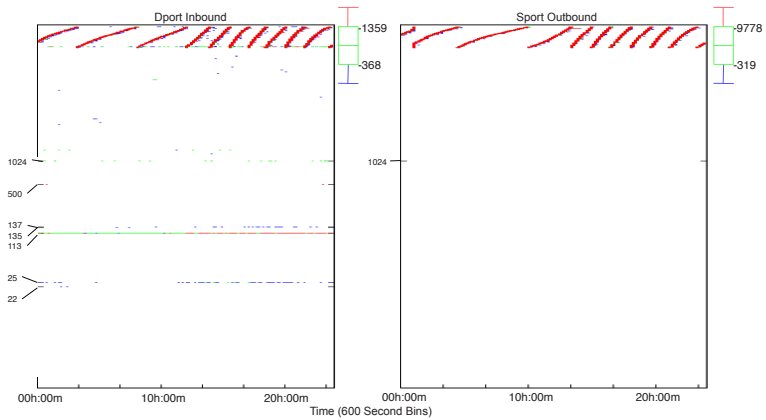
We define a *hidden* server to be any host that provides a service to hosts outside of the network, without the administrator’s knowledge or consent. Specifically, we show a misconfiguration causing a client to function as its own mail relay. We compare this behavior with a known mail relay.

Here, we display one day of a host’s activity with two existence plots, which we refer to as *existence plot pairs* (see Figure 3). The existence plot on the left represents the destination ports of packets in the data set `inbound` for the host in question (`dport inbound`). The existence plot on the right represents the source ports of packets in the data set `outbound` for the host in question (`sport outbound`). In both plots we use a  $\tau$  size of ten minutes (600 seconds).

Figure 3 shows a misconfigured host. Nominally, the host should use an internal mail server for mail inspection and distribution. However, due to a misconfiguration, the host begins forwarding mail to external hosts itself. As the figure on the left shows, the host receives a great deal of traffic to a limited number of ports but does not respond to these connection attempts (*e.g.* there is a lack of activity in the figure on the right). Instead, a short quick burst of activity occurs in the ephemeral port range. Upon inspection of the NetFlow records associated with this burst, we concluded that this burst encapsulates connections to 75 unique mail servers and lasts for approximately one minute, after which no further mail activity is observed from the host. This visualization yields two key insights. First, the activity occurs in a very short time and is inconsistent with the host’s *modus operandi*. Second, the host receives consistent scans to ports associated with well-known vulnerabilities but does not respond. Therefore, we



**Fig. 3.** Existence plot pair of a misconfiguration. The host contacts 75 mail servers in a short time.



**Fig. 4.** Existence plot pair of a legitimate SMTP relay. The host only acts as a client, forwarding mail to external mail servers.

are able to rule out the possibility of this being a compromised host functioning as a mail relay for external hosts.

Contrary to the activity demonstrated in Figure 3, Figure 4 shows an existence plot pair of a known mail relay. Similar to Figure 3, this host receives a large number of connection attempts from external hosts, and likewise, does not respond. However, unlike Figure 3, the host has a consistent pattern of ephemeral port activity, with no reserved port activity. Additionally, the ephemeral port activity does not show drastic deviations, *e.g.* the port cycling continues through the course of the day.

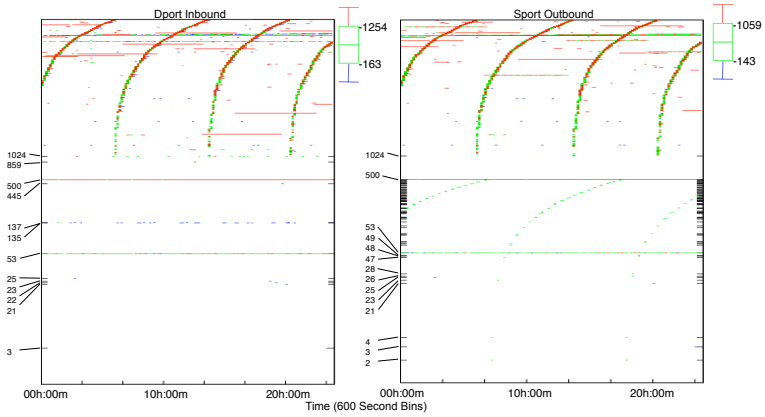


Fig. 5. Existence plot pair of a host using reserved ports to scan. Port cycling is present in the reserved port range.

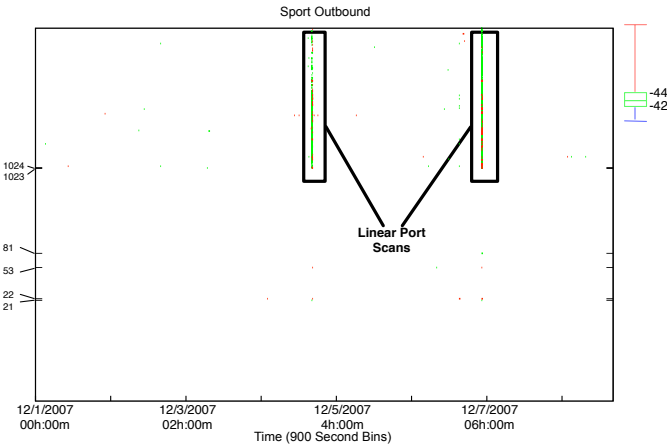


Fig. 6. Existence plot showing rampant scan response. The vertical bars denote scan response.

### 3.2 Scan Detection

Figure 5 shows an existence plot pair representing the port usage of a compromised host. Upon inspection, we find that the host appears to have two different instances of port cycling. The first instance is in the ephemeral range, as is expected from a client. The second is the port range 54 through 499. Since the second instance only occurs in the image on the right, we can exclude scan response as a possible reason. Upon inspection of the host's connections, we find that this activity is directed at external hosts listening on port 53 (DNS), 67 (DHCP) and 137 (NetBIOS). All of the packets observed are of similar size and have a greater than 99% fail rate over an observation period of 5 days (e.g. at no time in the observation period did the

victims of the scan attempt to complete a connection with the host). From this we conclude that this is an internal host scanning external hosts (presumably as the result of compromise).

In addition to being able to visualize monitored hosts' scan attempts, existence plots can demonstrate external hosts' scan successes. By concurrently representing the ports, existence plots are well-suited at demonstrating vertical scanning (*i.e.* every port on the host was contacted by the scanner) and scan response. Figure 6 shows the source port utilization of outbound traffic for a host over one week. During this time the host was vertically scanned twice, which is indicated by two vertical bars of activity.

## 4 Related Works

Work centering on visualizing a single host's activities is limited in comparison to work on large-scale visualizations of network communication. Some work [3][11] centers on visualization of networks at multiple levels, but the focus is still heavily skewed to higher-level views of network interactivity, providing only rudimentary measurements of individual hosts.

It can be argued that existence plots are quite similar to scatterplots, which are common in network traffic analysis [5][6][9]. Indeed, there are similarities, but we note that, with the exception of Marchette [5], previous work limits traffic representation to specific ports or events (an instance of specific phenomena that meets some selection criteria), where existence plots place no emphasis on aggregating activity based on attribution. Instead, existence plots emphasize expressing activity from all ports for a given host over time. Existence plots can be seen as an expansion on Marchette's work in visualizing port activity over time by adding color as a representation of changes in magnitude.

Two other visualization methods that represent individual hosts are graphlets and heat maps. Graphlet approaches visualize expected protocol usage patterns [4]. Mansmann et al. [4] abstracts expected server behavior into gravitational entities that affect a host's position on a plane; hosts are drawn to their most prevalent activity. Heat maps aid in detecting obfuscation by comparing the commonalities among patterns of communication [2][8]. Wright et al. [8] use heat maps to classify encrypted connections based on prevalent patterns in a host's network traffic, and Hernandez-Campos et al. [2] use heat maps for broader-scale representation of network traffic. Unlike heat maps, existence plots use only four discrete states to display magnitude instead of a continuous color map. Since varying time resolution can greatly affect the smoothness of magnitude changes (causing jarring color shifts in heat maps), we find the existence plot to be more informative.

## 5 Conclusions

In this work, we have demonstrated the use of a low-resolution visualization, which we call the existence plot, in representing the port usage of individual hosts, particularly in detecting hidden servers and scanning activity. In both

cases, the existence plot provides useful insight into the host's activities by concurrently representing ports' usage over time. In the future, we intend to use this visualization to further discussion about port interactivity and systematic patterns of port usage. Also, we intend to introduce user interactivity to the visualization, which will allow the user to select specific patterns of interest and obtain the original NetFlow records that the pattern represents.

## References

1. Collins, M., Shimeall, T., Faber, S., Janies, J., Weaver, R., De Shon, M., Kadane, J.: Using uncleanliness to predict future botnet addresses. In: Proceedings of IMC 2007 (2007)
2. Hernandez-Campos, F., Nobel, A., Smith, F., Jeffay, K.: Understanding patterns of tcp connection usage with statistical clustering. In: Proceedings of MASCOTS 2005. IEEE Computer Society, Los Alamitos (2005)
3. Lakkaraju, K., Yurcik, W., Lee, A.: Nvisionip: netflow visualizations of system state for security situational awareness. In: Proceedings of VizSEC 2004 (2004)
4. Mansmann, F., Meier, L., Keim, D.: Visualization of host behavior for network security. In: Proceedings of VizSEC 2007 (2007)
5. Marchette, D.: Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint. Springer, New York (2001)
6. McPherson, J., Ma, K., Krystosek, P., Bartoletti, T., Christensen, M.: Portvis: a tool for port-based detection of security events. In: Proceedings of VizSEC/DMSEC 2004. ACM, New York (2004)
7. Pang, R., Yegneswaran, V., Barford, P., Paxson, V., Peterson, L.: Characteristics of internet background radiation. In: Proceedings of IMC 2004 (2004)
8. Wright, C., Monrose, F., Masson, G.: Using visual motifs to classify encrypted traffic. In: proceedings of VizSEC 2006. ACM Press, New York (2006)
9. Xiao, L., Gerth, J., Hanrahan, P.: Enhancing visual analysis of network traffic using a knowledge representation. VAST 0, 107–114 (2006)
10. Yegneswaran, V., Barford, P., Ullrich, J.: Internet intrusions: Global characteristics and prevalence. In: Proceedings of ACM SIGMETRICS 2003 (2003)
11. Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: Visflowconnect: netflow visualizations of link relationships for security situational awareness. In: Proceedings of VizSEC/DMSEC 2004. ACM Press, New York (2004)



# Using Time Series 3D AlertGraph and False Alert Classification to Analyse Snort Alerts

Shahrulniza Musa<sup>1</sup> and David J. Parish<sup>2</sup>

<sup>1</sup> University Kuala Lumpur, Malaysian Institute of Information Technology,  
1016, Jalan Sultan Ismail, 50250 Kuala Lumpur  
shahrulniza@mit.unikl.edu.my

<sup>2</sup> Electronic and Electrical Eng. Dept. Loughborough University,  
Loughborough, LE11 3TU U.K  
D.J.Parish@lboro.ac.uk

**Abstract.** A top-level overview of Snort alerts using 3D visual and alert classification is discussed. This paper describes the top-level view (time series 3D AlertGraph) with the integration of alert classification to visualise Snort alerts. The advantages of using this view are (1) It summarised the alerts into different colours to indicate the quantity of alerts from (SRCIP, DPORT) pairs; (2) It used alert classification to highlight the true alerts; (3) Through interaction tools, the alerts can be highlighted according to the source IP, destination IP or destination port;. (4) A large numbers of alerts can be viewed in a single display and (5) A temporal characteristic of attacks can be discovered.

**Keywords:** machine learning, alert visualization, network security information visualization, alert classification.

## 1 Introduction

The difficulty to explore and analyse large quantities of network security alerts in text form has inspired many researchers to use visual methods as an alternative. Current research in network security visualisation has grown and many display techniques have been explored. Some of the tools developed were NVisionIP [1], IDSRainstorm [2], SnortView [3], VisFlowConnect [4] and many others. In general, the main focus of visualisation software is to achieve visual patterns of the true alerts and to help the analysts to explore them. Even with the aide of the visualisation software, identifying the attack patterns is still a difficult task. This may be due to the occlusion or due to the numerous alerts crowding the visualisation display.

One-way to overcome overcrowding alerts is by reducing the numbers of alerts to ease the analysis. Lee [5] and Viinikka [6] employed statistical method such as Granger Causality Analysis and EWMA Control Charts to remove alerts that formed the normal behaviour of the monitored network. Tedesco [7] used token bucket filter to limit the quantity of alerts in each alert category and monitoring window. Pietraszek [8] and Bloedorn [9] employed machine learning techniques to classify alerts into false and true alerts and then removed the false ones. In these methods though, the classified alerts were not visualised and still presented in text form.

Work by J. Colombe [10] used multivariate Bernoulli event representation to convert the text alerts from RealSecure IDS to a visual form with statistical method. In this method, they convert each comma limited text alarm to a binary form (0 or 1) that marks the present or absence of a text descriptor in the particular field. The binary string will then form an alarm vector that corresponds to a specific descriptor token. The typicality score of each alert is calculated using statistical method. This score is then represented using colour code to give a differentiation of highly typical alerts and anomaly alerts. On the other hand, work by Stefan Axelsson [11] used visualisation with a self-learning Bayesian system to calculate each alert a token score and assigned the token a colour code. Then, the alerts are visualised according to its colour code.

## 2 The Visualisation Prototype Overview

We have developed prototype visualisation software to visualise Snort alerts for network security monitoring and analysis using multiple 3-D visualisation [12]. The visualisation prototype combines various displays in 3D to give visual insight of the data such as scatter plot, parallel coordinates plot, timeline view, plane view and geographical view. In this visualisation, we visualise the alert as an object that gives abstract information of its value. Glyphs, lines, colour-coding and visual positioning become the abstract information of the attacks. The details of the attack are the signature or the class type, the severity, the victim IP address, the source IP address, the protocol, the indicative quantity, the instance and the alert classification, the true or false positive alert. We have also included interactions, filtering and drill-down in our visualisation. We also incorporate animation of the alerts according to the timeline and real-time monitoring.

In addition, we made an improvement to the visualisation prototype by summarising the alerts data and gave the overview of the network security status. This view was plotted in a 3D plot named as time series 3D AlertGraph which was an extension of the 2D histograms [13] into 3D. Ren and others [13] designed an interactive visualisation system, IDGraph by plotting the aggregated number of unsuccessful TCP connections on the vertical axis versus the time ordered sequence on the horizontal axis of a 2D plot. Then the brightness of the mapped point was changed according to the density of the data at each pixel. The mapped data was either the pair of source IP-destination port, source IP-destination IP or destination IP-destination Port. This technique was named the histograms technique and was derived from the information mural visualisation technique [14].

This paper describes the novel time series 3D AlertGraph and false alert classification to visualise Snort alerts. The advantages of using the 3D AlertGraph are:

- It summarises the alerts into different colours to indicate the quantity of alerts from (SRCIP, DPORT) pairs.
- It uses false alert classifier to highlight the true alerts.
- Through interaction tools, the alerts are highlighted according to the source IP, destination IP or destination port.
- A huge numbers of alerts are viewed in a single display.
- A temporal characteristic of attacks can be discovered.

## 2.1 Dataset Used

The dataset used was obtained from the published Darpa Intrusion detection evaluation dataset 1999 (DARPA 1999) from the Lincoln Laboratory, Massachusetts Institute of Technology [15]. The offline tcpdump dataset was analysed using Snort with a standard configuration and the alerts were logged into a MySQL database. Another set of data for this study was obtained from honeynet tcpdump traffic data from the High Speed Network research lab, Electronic and Engineering department, Loughborough University. A month of data from 01/04/2006 until 30/04/2006 were visualised and studied.

## 3 The Time Series 3D AlertGraph

### 3.1 The Design

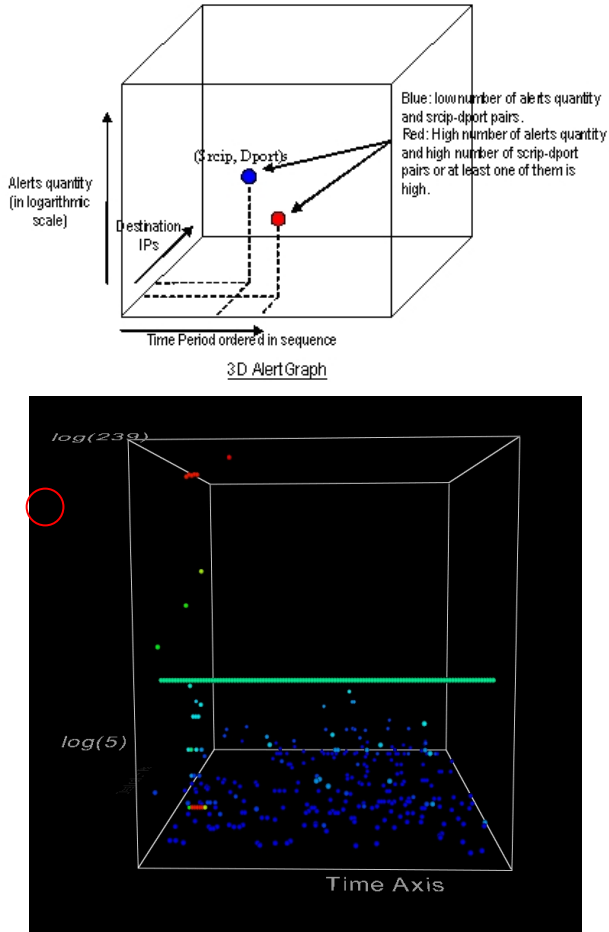
The time series 3D AlertGraph was plotted by mapping the quantity of alerts received from the source IP and destination port (SRCIP, DPORT) pair on the Z-axis, the destination IP addresses on the Y-axis and the time ordered sequence on the X-axis of the 3D plot (Figure 1). Each mapped point in the 3D plot was attached with a coloured sphere. The sphere was coloured according to the number of the different pairs of (SRCIP, DPORT) and the number of alerts received at that point.

### 3.2 The Colour Scheme

Each map point represented the number of alerts received by a destination IP in a given interval from (SRCIP, DPORT) pairs. In an interval, there might be more than one (SRCIP, DPORT) pair that had the same number of alerts. Therefore, we counted the (SRCIP, DPORT) pairs and used it as one of the variables to colour the sphere. Another variable used to colour the sphere was the number of alerts at that point which was the Z-axis value. The scalar coefficient to colour the sphere was the sum of number of alerts and number of (SRCIP, DPORT) pairs at a point.

By considering the quantity of alerts and the number of pairs available at a point, ranges of colours could be achieved. In this method, the quantity of alerts and the number of pairs available at a point would have the same importance in the colour scheme. This variation of colours was helpful to understand the network status and identify the true attacks. To automate the identification of true attacks, users can apply the false alert classification. In this case, the true alerts sphere will be coloured in red. This was achieved by assigning the maximum value between the number of pairs and the number of alerts in each pair to the scalar coefficient of the true alert sphere.

Mapping the scalar coefficient to colour was achieved by using the VTKLookupTable class in the VTK module. For the VTKLookupTable class to create range of colours, we supplied the class with the minimum and the maximum range of the scalar coefficient in the monitoring period. We set the mapping colour table to vary from blue to red. The colour transformation was using the logarithmic function in the VTKLookupTable. The blue colour represents the lowest value and the red being the highest.



**Fig. 1.** The schematic diagram and a day of Snort alerts in 3D AlertGraph

This colour variation is to highlight to the user, the density of alerts and the (SRCIP, DPORT) pairs in the data. The red colour will suggest either

- Many alerts from a unique (SRCIP, DPORT) pair or
- Many alerts from many (SRCIP, DPORT) pairs but with few alerts each or
- The alerts are the true alerts

### 3.3 The Interactive Features

This 3D AlertGraph also gave a temporal characteristic of the alerts received by destination IP address, source IP address and destination port. We added interactive features in such a way as to perform the Shneiderman's visual information-seeking Mantra, 'Overview first, zoom and filter, details-on-demand' [16] and to understand

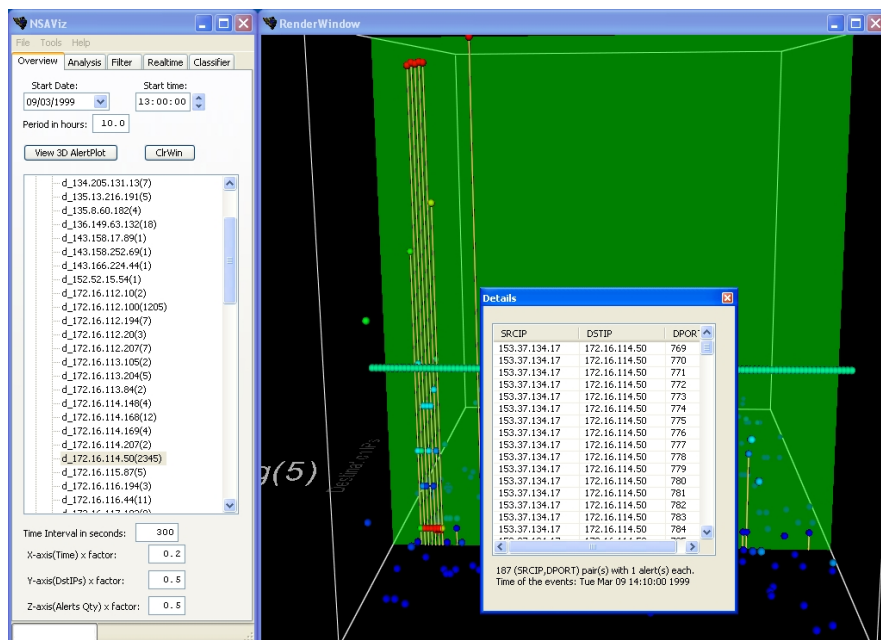
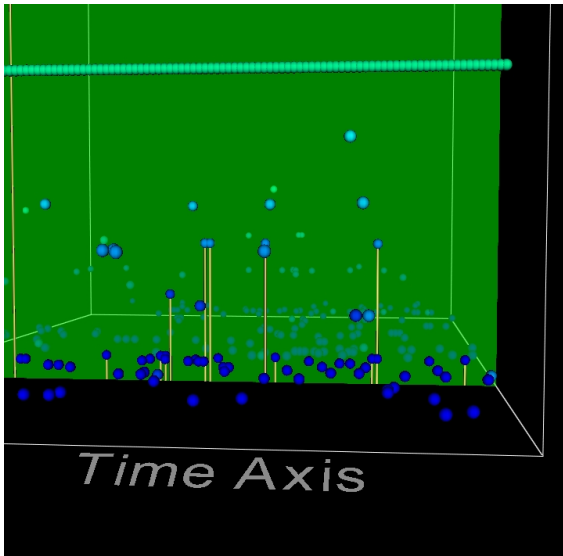


Fig. 2. Pop-up window when user picks a coloured sphere actor

the data easier. The first interaction was by ‘picking’ the coloured sphere actor. The details of the source IP, destination IP and destination port will be popped-up with the time period of the event, the alerts quantity and the number of (SRCIP, DPORT) pairs (Figure 2).

The second interaction tool was to highlight a specific source IP or destination IP or destination port. It was achieved by clicking the child item in the SRCIP-DSTIP-DPORT tree panel. By double clicking a destination IP, a transparent green plane will appear with yellow lines from the 3D plot base to all data points in that respective destination IP (Figure 3). The yellow lines represent a histogram graph of the number of alerts received according to the time. However, double clicking the source IP or destination port will highlight the spheres having the source IP or destination port. At the same time, yellow lines will be drawn from the 3D plot base to the respective data points. Other settings the users can modify include the start time, the monitoring period, the time interval and the scales of the 3D plot.

In summary, the 3D AlertGraph showed the alerts received by the destination IP address from the pair of (SRCIP, DPORT) in a time interval. The colour of the sphere showed the sum of the (SRCIP, DPORT) pairs with the number of alerts at a point. By analysing the red colour spheres with the known attacks in the dataset, the red colour spheres were the true attacks. A single red sphere at the top of the 3D plot may suggest a possible DOS attack and continuous horizontal red spheres at the bottom of the 3D plot may point out a port scanning or a port sweep attack (see Figure 1).



**Fig. 3.** A transparent green plane at the selected destination IP address

### 3.4 The False Alert Classification

This approach integrated the classification tree algorithm to help users to identify true alerts. The classifier learns from the labelled training sample provided by the user and builds a classifier model from it. As the objective was not to build a new classification algorithm, we applied the decision tree learner provided by an open source machine learning module, Orange [17]. The classification tree learner in Orange is actually based on the C4.5 classification algorithm and in default setting provides the same result as in C4.5 classification algorithm.

The parameters used to teach the classifier were the source IP, source port, destination IP, destination port, alert class type, IP datagram length and IP protocol. Before sending the alerts to the classifier, we first processed them into general form. The source IP address and destination IP address were generalised to local or foreign host. For the source and destination ports, they were generalised into standard ports, ephemeral ports, unassigned ports or unknown ports. This was based on the Berkeley Software Distribution (BSD) TCP/IP stack port ranges. Standard ports were the ports with port number less than or equal to 1024. Ephemeral ports were the port numbers between 1024 and 4999. While the unassigned port numbers, were the port numbers greater than or equal to 5000. The unknown port number was assigned when there was no information of the port number in the IDS alert data. All of the attributes were discrete attributes except for the IP datagram length, which was a continuous attribute. The objective of this generalisation process was to avoid the classifier from memorising a specific example in the training set rather than producing general predictive rules. Table 1 shows the summary of the alert generalisation.

**Table 1.** Alert Generalisation

<b>Alert attributes</b>	<b>Generalisation</b>
Source / Destination IP address	Local host Foreign host
Source / Destination port	Standard ( < 1024) Ephemeral (between 1024 and 4999) Unassigned ( > 5000) Unknown
Alert Classtype	Class type as specified by Snort
IP Datagram length	Actual byte value
IP protocol	UDP, TCP, ICMP, Reserve, Other

In a decision tree learner, the end leaf will be the attribute associated with the classification result, true or false. While the internal nodes correspond to each value of its associated attributes takes. The decision tree is generated by a recursive loop of learning element and by splitting the best attributes that splits the training example into their proper class. We also applied a post-pruning (backwards pruning) to simplify the classification tree. Pruning is one of a technique employed in machine learning to tackle the situation when the algorithm memorises the training data but fails to predict a new instance well in the future. This situation is also known as overfitting.

We used k-fold cross-validation technique [18] to measure the classifier performance. In this technique, the data set was divided randomly into k equal subset. The classifier was then built from the (k-1) data subset and tested against the remaining subset. This procedure was repeated k times with different training set and test set. The average performance measures from all tests were then calculated. We chose k=10 as recommended in the literature. The standard machine learning performances scores calculated were the classification accuracy (CA), the Brier Score (BS) and the area under the receiver operating characteristic (ROC) curve (AUC).

We measured the classifier performance using a training sample that consisted of 561 alerts. The training sample had been added through Add/Edit training sample feature in the prototype. As the dataset is a skewed class distribution where there are high numbers of false alert and low numbers of true alert, we sampled the training set with balance number of true and false alerts. The training sample consisted of 273 false alerts and 288 true alerts. The performance scores and the confusion matrix were shown in Table 2 and Table 3.

The result showed the performance scores of the classifier were excellent. The area under the curve (AUC) ROC and classification accuracy (CA) were above 0.9857. For Brier score (BS), the score was 0.0265 which showed high accuracy. In contrast, Table 3 showed the confusion matrix score and the achieved true positive (TP), true

**Table 2.** Classifier Performance Scores

	<i>CA</i>	<i>BS</i>	<i>AUC</i>
<i>Scores</i>	<i>0.9857</i>	<i>0.0265</i>	<i>0.9892</i>

**Table 3.** Classifier Confusion Matrix

	<i>Negative (false) Predicted</i>	<i>Positive (true) Predicted</i>
<b>Negative (false)</b>	<i>a</i> 267 ( <i>TN=0.9780</i> )	<i>b</i> 6 ( <i>FP=0.0220</i> )
<b>Positive (true)</b>	<i>c</i> 5 ( <i>FN=0.0174</i> )	<i>d</i> 283 ( <i>TP=0.9826</i> )

negative (TN), false positive (FP) and false negative (FN) rates. The result showed the classifier was not biased towards any class category in the training sample.

Finally, we tested the classifier with alerts dataset from DARPA1999. To measure the classifier performance, we tested the classifier against the second week of the dataset. We compared the classifier detections with the attacks in the intrusions list from the dataset [15]. We assessed the classifier performance by counting the number of its true detections. From the intrusions list, there were initially 43 attacks. However, as Snort missed some of the attacks, the available attacks left were 18. We noted that five of the attacks were in the training sample. We assumed it was appropriate to test the classifier with the dataset, as the alerts used to build the classifier model were in general form. There was no details information of the attacks available to the classifier such as IP address, port number and others.

From the testing we found that our classifier detected 17 out of 18 attacks in the intrusions list. Furthermore, the one wrongly classified attack could still be pointed out with the help of visualisation. The circled red sphere in Figure 1 was the wrongly classified alerts but was still highlighted in the 3D AlertGraph. Our classifier also classified five more anomalies as true attacks.

### 3.5 The Overview Example

#### PortSweep

The display in the 3D AlertGraph for 24 hours period starting from 13:00 Monday 9 March 1999 highlighted a continuous horizontal red spheres and a high quantity of alerts at the beginning hours of the monitoring. The details from the pick object revealed that at the red spheres area, there were a unique host attacking a unique local host using multiple destination port (see Figure 4). This was an indication of a portsweep attack.

#### Slammer Worm Propagation

The slammer worm is a self-propagating malicious code that exploits the vulnerability in the Resolution Service of Microsoft SQL Server 2000 and Microsoft Desktop Engine (MSDE) 2000. It attacks port 1434 using the UDP protocol. An observation of a day (07/04/2006) dataset which consists of 442 alerts from the Honeynet traffic showed there were attacks by the slammer worm to the Honeynet. Observation in the



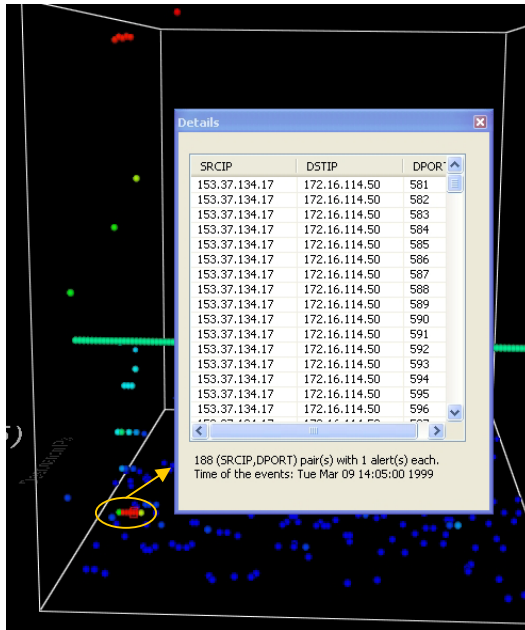


Fig. 4. Image of PortSweep

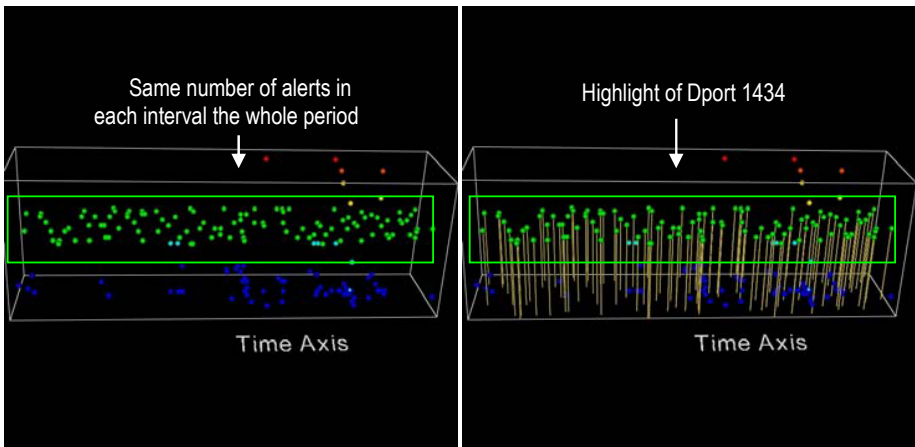


Fig. 5. The image of Slammer Worm propagation

3D AlertGraph on that day showed there were a constant number of alerts in each interval for the whole period attacking the same destination port 1434 to the multiple hosts. The two pictures in Figure 5 showed the view in the 3D AlertGraph.

## 4 The User Evaluation

Following the Nielsen [19] findings, we have conducted the usability test of three participants. We tested our tool with researchers at High Speed Networks laboratory. All of them were familiar with intrusion detection and have a good background in network security in general. Their participation in this study is voluntary and they have never used this tool before. In summary the usability issues found are (1) the method to control the motion in 3D and (2) the mouse-over information when user brings the mouse cursor over an object. Some users felt difficult to control the 3D image movement using the VTK mouse control built-in features. It is due to lack of training. However, after several trials, they managed to control it correctly. One participant suggested a navigation control as in the ‘google map’ application to be used. We plan this feature in the future work.

In the user evaluation, apart to find the usability problems and finding bugs, we also asked each participant to analyse a day of alerts data which contained 1539 alerts. From the data, we asked each participant to list three local hosts that were under attacks and to identify the one that was severely attacked using the time series 3D AlertGraph. All the participants had successfully identified the local hosts and the one that was severely attacked almost immediately. They were agreed the time series 3D AlertGraph was easy to understand and to identify the potential true attacks in the network. We also asked the participants to evaluate the software on the following themes (Table 4) using a Likert Scale 1 to 5 with 1 for most dissatisfied and 5 for most satisfied.

For each of the themes, the average score was above 4.0 with standard deviation less than or equal to 0.58. The result was positive and showed to us that this prototype were successfully designed and addressed the needs of the user.

**Table 4.** Themes for user evaluation

Themes	Average	Std dev
GUI - user friendly	4.33	0.58
Interaction	4.67	0.58
Classifier features	4.67	0.58
Filter features	4.67	0.58
Real-time	4.00	0.00
Reporting features	4.67	0.58
Perform Security tasks	4.67	0.58

## 5 Conclusions

We have designed a visualisation tool with alerts classification to provide a better visual pattern or highlight of true attacks. The 3D AlertGraph serves as a top level overview of the network security status with temporal outline of the events. Using a

classification tree based on C4.5 classification algorithm with visualisation also helps users to identify the true positive alerts. User evaluation shows that the design provide better understanding of the alerts and can identify true attacks in the network.

## References

1. Lakkaraju, K., Yurcik, W., Lee, A.J.: NVisionIP: netflow visualizations of system state for security situational awareness. In: *VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. ACM Press, New York (2004)
2. Abdullah, K., et al.: IDS rainStorm: visualizing IDS alarms. In: *IEEE Workshop on Visualization for Computer Security, 2005 (VizSEC 2005)* (2005)
3. Koike, H., Ohno, K.: SnortView: visualization system of snort logs. In: *VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. ACM Press, New York (2004)
4. Yin, X., Yurcik, W., Slagell, A.: The Design of VisFlowConnect-IP: A Link Analysis System for IP Security Situational Awareness. In: *IWIA 2005: Proceedings of the Third IEEE International Workshop on Information Assurance (IWIA 2005)*. IEEE Computer Society, Los Alamitos (2005)
5. Lee, W., Qin, X.: Statistical Causality Analysis of INFOSEC Alert Data. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) *RAID 2003*. LNCS, vol. 2820, pp. 73–93. Springer, Heidelberg (2003)
6. Viinikka, J., et al.: Time series modeling for IDS alert management. In: *ASIACCS 2006: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, New York (2006)
7. Tedesco, G., Aickelin, U.: Data Reduction in Intrusion Alert Correlation. In: *WSEAS Transactions on Computers*, pp. 186–193 (2006)
8. Pietraszek, T.: Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) *RAID 2004*. LNCS, vol. 3224, pp. 102–124. Springer, Heidelberg (2004)
9. Bloedorn, E.E., Talbol, L.M., DeBarr, D.D.: Data Mining Applied to Intrusion Detection: MITRE Experiences. *Machine Learning and Data Mining for Computer Security*, pp. 65–87 (2006)
10. Colombe, J.B., Stephens, G.: Statistical profiling and visualization for detection of malicious insider attacks on computer networks. In: *VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. ACM Press, New York (2004)
11. Axelsson, S.: Combining a bayesian classifier with visualisation: understanding the IDS. In: *VizSEC/DMSEC 2004: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*. ACM Press, New York (2004)
12. Musa, S., Parish, D.J.: Visualising Communication Network Security Attacks. In: *11th International Conference*. IEEE Computer Society, Zurich (2007)
13. Ren, P., et al.: IDGraphs: Intrusion Detection and Analysis Using Histograms. In: *Proceedings of the IEEE Workshops on Visualization for Computer Security*. IEEE Computer Society, Los Alamitos (2005)
14. Dean, F.J., John, T.S.: *The Information Mural: A Technique for Displaying and Navigating Large Information Spaces*. IEEE Educational Activities Department, pp. 257–271 (1998)

15. MIT, L.L.: DARPA Intrusion detection evaluation dataset (1999)
16. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: Proceedings of the IEEE Symposium on Visual Languages. IEEE Computer Society Press, Washington (1996)
17. Demsar, J., Zupan, B., Leban, G.: Orange: From Experimental Machine Learning to Interactive Data Mining. White Paper, Faculty of Computer and Information Science, University of Ljubljana (2004), <http://www.ailab.si/orange>
18. Kohavi, R.: A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI). Morgan Kaufmann, San Francisco (1995)
19. Nielsen, J.: Usability inspection methods. In: Conference companion on Human factors in computing systems, ACM Press, Boston (1994)

# Network Traffic Exploration Application: A Tool to Assess, Visualize, and Analyze Network Security Events

Grant Vandenberghe

Network Information Operations Section  
Defence Research and Development Canada (DRDC)

**Abstract.** Defence Research and Development Canada (DRDC) is developing a security event / packet analysis tool that is useful for analyzing a wide range of network attacks. The tool allows the security analyst to visually analyze a security event from a broad range of visual perspectives using a variety of detection algorithms. The tool is easy to extend and can be used to generate automated analysis scripts. The system architecture is presented and its capabilities are demonstrated through the analysis of several covert tunnels.

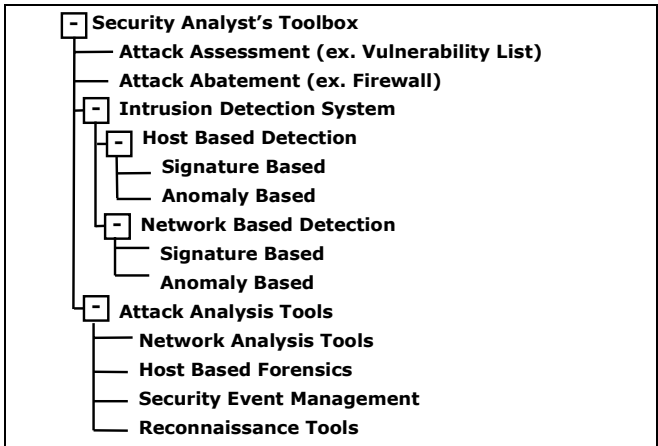
**Keywords:** Packet Analysis, Network Forensics, Visualization, Covert Tunnels.

## 1 Introduction

Although many organizations recognize the value of an automated response to a cyber attack, the trust in the response is lacking. False positives are routinely seen in many systems and when a suspicious event occurs, an analyst is required to review the event. The review process determines if the event is malicious. If an alert occurs often enough then the analyst may write a script to validate it. Often, the time required to write a script is considerable and requires the skills of a highly trained analyst.

Described in this paper is a data visualization tool that allows the analyst to investigate network events and automatically create validation scripts that embed the analyst's experience into the security infrastructure. It also allows security teams to progressively transform security analysis from a manual to an automated defensive detection system.

Analysts make use of a range of tools to detect incursions, repel attacks, and eliminate false positives. A taxonomy of these tools is shown in Figure 1. Analysts use attack assessment tools such as vulnerability databases to understand intent and determine what types of systems are vulnerable. Vulnerable systems can be protected using attack abatement tools like a firewall. Although the analyst can limit network exposures using a firewall, some potentially malicious packets may still inadvertently enter the network. To detect malicious packets, the analyst will use either a host or network-based intrusion detection system (IDS). A network-based product examines all traffic crossing a point on the network. Alternatively, a host-based IDS is intended to monitor a single host.



**Fig. 1.** Security Tool Taxonomy

The host and network-based IDS tools can be further categorized by their detection algorithms. Both anomaly-based and signature-based algorithms are capable of detecting a wide range of attacks but both have a number of weaknesses. Signature-based tools are only effective at detecting known signatures and rely on well-written signatures to reduce false alarms. Anomaly-based IDS's are able to detect unusual patterns in network traffic but often require an analyst to judge the significance of suspicious patterns [1,2]. Often, the resulting text-based security events do not adequately convey the magnitude of an abnormality that is best presented in graphical terms.

Some IDS systems are starting to support both signature-based and anomaly-based algorithms. However, most network-based tools have a fairly rigid architecture that makes adding new anomaly detection algorithms difficult. Even open source signature-based tools like SNORT [3] have relatively few anomaly-based plug-ins available for them.

A typical process used to collect and respond to a security event is shown in Figure 2. With respect to IDS implementation, some security events generated by the IDS are reliable enough to warrant an automated response. In other cases, the security events need to be reviewed by an analyst. All events generated by the IDS are sent to some type of database or security events management (SEM) system. Although SEM systems can correlate events that surround a potential intrusion, the analyst still needs to validate the event and determine a course of action [4].

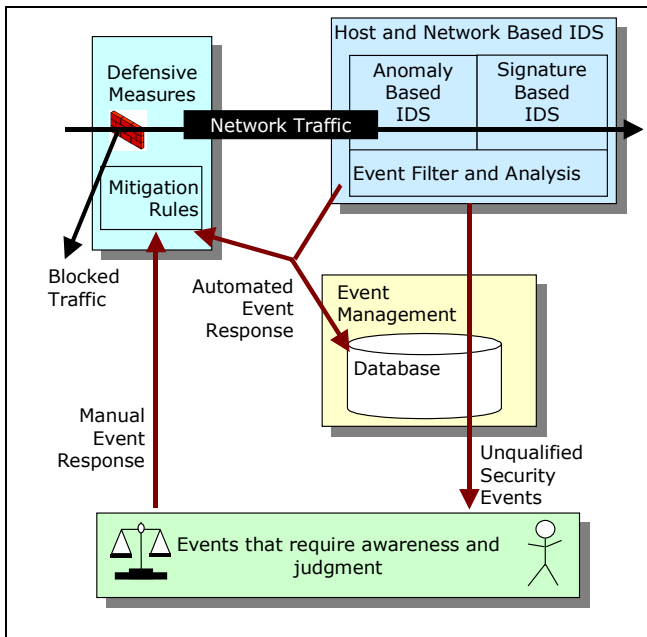
There is a wide range of network-centric tools that can assist the analyst in analyzing a network-based event. A quick survey showed that there were more than 50 tools freely available [5-10]. Many of these tools complement one another but, in general, are not interoperable and do not provide unencumbered exchange of information among them. Rather than trying to discuss all 50 tools that have already been well covered in [5-10], four tools will be discussed.

SGUIL [11] is a GUI-based tool that combines SNORT [3], Wireshark [12], P0f [13], tcpflow [14] and libpcap [15] / tcpdump [16] into a single environment. Some reviewers have described SGUIL as a tool “by analysts, for analysts” [10]. This tool does not have any script writing capability and has limited graphical visualization options.

OPNET-ACE [17] is a commercial tool that allows an analyst to characterize application interactions. The product is not specifically tuned to security analysis but it does have a very strong visualization component associated with it.

BRO [18] is essentially an enhanced signature-based IDS but it includes a very robust scripting language. The tool significantly extends signature-based analysis and is easier to program than scripts written in PERL or C. However, it does not try to combine signature and anomaly-based methods together into a single environment.

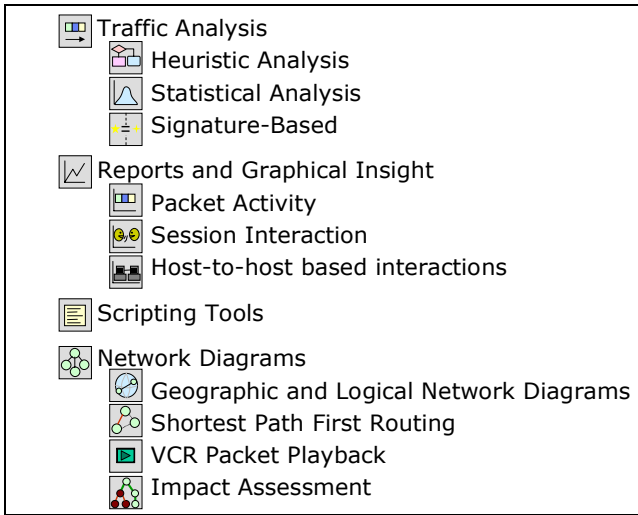
CA- eHealth [19] is a network management system which can store and present the state of the enterprises networking equipment however this tool does not allow the user to examine packet activity on the network.



**Fig. 2.** Typical Cyber Defence Process

When the above tools are applied to the security event analysis process small feature gaps exist. These feature gaps can be summarized as follows:

- Many IDS's allow the analyst to add new signatures but most do not allow the analyst to add new anomaly detection algorithms.
- Most security-based tools deal with events on a text basis and do not exploit the full range of visualization options available to them.
- Scripting tools and analysis tools are highly separated. No existing tools take the experience gained from performing an analysis and use it as a basis for strengthening the defences in the future.
- There are many niche tools available but they are limited in their ability to exchange information.



**Fig. 3.** Core Functional Areas of the NTE

- Most tools deal with the high level network or the low level packet analysis but there is little available to transition between the views. The ability to relate a tiny packet level event to the significance of the network at large is not well addressed.

To address the above limitations, the Network Traffic Exploration (NTE) tool was developed. This in house tool is being used to explore different event analysis approaches and is used internally for a wide range of packet analysis activities. This tool combines six key functional areas into a single package as shown in Figure 3. The tool includes rudimentary intrusion detection and extensive analysis functionality. NTE readily connects to the SNORT signature-based IDS and includes several simple anomaly-based features as well. The tool works with both anomaly detection algorithms and text-based logs. It uses a wide range of visualization features and text-based reports to bridge the gap between the different detection techniques. The tool can generate scripts and is open enough to allow the analyst to easily add new algorithms to any of the core areas.

NTE also provides several methods of traffic analysis. Each method can be used individually or techniques can be combined to provide a stronger overall analysis capability. Consider an ICMP packet exchange with the following abnormal characteristics:

- Unusual timing patterns
- Unusual packet length
- Multiple responses to a single query
- Incorrect ICMP checksums.

Individually, each characteristic is circumstantial but together they point to a potential covert tunnel.



Beyond the straight analysis capabilities, the NTE provides the analyst with the ability to develop and test customized traffic analysis scripts. Although there are many ways to create traffic analysis scripts, most require knowledge of a programming language like C or PERL. The NTE's approach allows an analyst to create scripts as a background activity and doesn't require prior programming knowledge. The analysis process is automatically stored by the NTE as the security event is processed. The NTE can be later asked to create a custom script based on the user's past actions. This custom script can be manually reused or potentially deployed to an unattended computer that would periodically run a series of these scripts in batches. An operational implementation of the design is shown in Figure 4. In this potential implementation, the script's execution would be triggered when IDS alerts matching certain characteristics are detected. The script would be rerun to validate the alert with the results fed back into a centralized event management system.

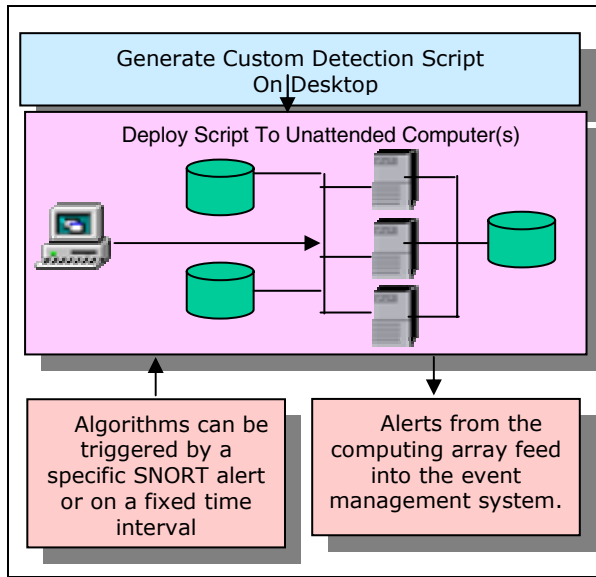
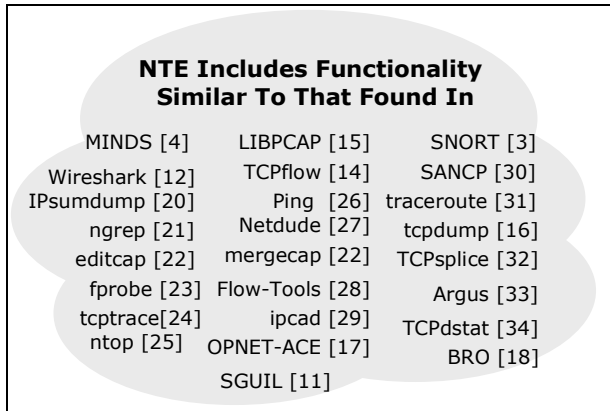


Fig. 4. NTE Process Workflow

## 2 Architecture

The previous section discussed the limitations of the current cyber defence tools. This section explains how NTE is structured to overcome these limitations using a layered software architecture and interwoven feature set. With respect to the software architecture, NTE has three layers. The software uses MATLAB as a development environment, a low level packet analysis library to perform specific tasks, and finally the NTE provides the unified application front end.



**Fig. 5.** NTE Scope (with references)

The choice of MATLAB as a development environment is somewhat unusual but has a number of advantages over a traditional high-level language. In a language like C or JAVA, all variables disappear (unless the data is stored to disk) when the program finishes executing. In MATLAB, the output of each algorithm is retained as variables in the desktop environment. This allows the operator to maintain a highly interactive interface with the data and makes portability between different algorithms much easier. In a C coding paradigm it would be the equivalent trying to code from within the debugger.

The packet analysis library was created by DRDC to provide a series of commonly used traffic analysis functions. The functions are largely written in MATLAB's internal language but there are some functions written in C and JAVA as well.

The NTE software is built on top of the packet analysis library. The tool extends well beyond the scope of the base library by adding an interactive GUI environment, additional data structures, data query capabilities, traffic profiling, help system and reporting capabilities.

The NTE has a broad suite of functions that are similar to that found in a broad range of analysis tools as shown in Figure 5. It interfaces directly to some tools such as SNORT and Wireshark. In other cases, the functions needed for general analysis happen to directly overlap with the other tools. NTE provides an environment where performance, visualization, statistical analysis, session analysis, and protocol analysis functions can all exchange data on an unencumbered basis.

The general architecture of NTE is shown in Figure 6 and provides these functions:

- Interface, manage, control and exchange data with a series of external tools.
- Place information in a series of arrays that follow a common storage philosophy.
- Analyze data from a variety of intrusion detection models
- Present information in text and graphical form to communicate the findings.
- Support analyst activity with a central management framework to hold the surrounding functionality together.

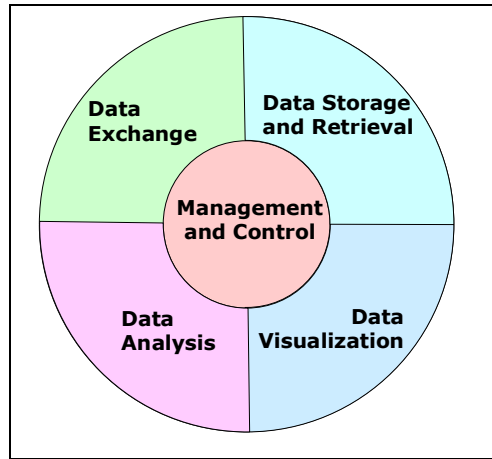


Fig. 6. NTE Architecture

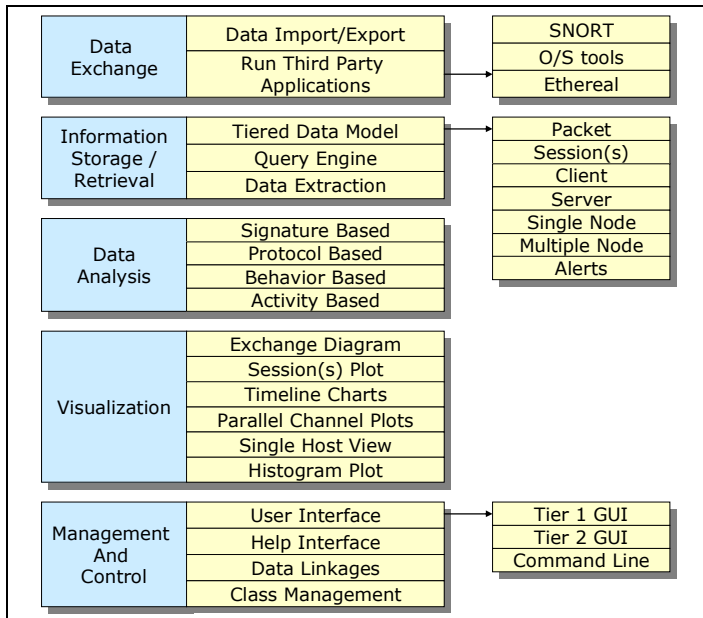


Fig. 7. NTE High Level Diagram

Figure 7 expands on each of the previously described functional blocks. The following subsections describe the functions in this figure in depth.

**DATA EXCHANGE:** NTE interfaces with a series of tools including Ethereal, SNORT, ping, traceroute, nslookup. NTE also reads the libpcap [15] file format and the fast and full SNORT alert data formats. Within NTE, there is a series of linkage

functions for the comparison or manipulation of data structures. Comparisons can be made between different data types. These features allow the operator to associate SNORT alerts to packets, sessions, or all packets communicated between a pair of nodes. The operator can also correlate, merge, as well as split information from multiple traffic loggers that may be skewed in time or distorted by a non-passive networking device such as a router, or VPN tunnel.

**DATA STORAGE AND RETRIEVAL:** The information received by NTE is stored in memory in a series of arrays that are similar to database tables. The tables summarize network or SNORT-based alert information into a relatively compact form that can be searched using a string-based query. Individual columns of the tables can be exported to external tools for analysis.

**DATA ANALYSIS:** NTE includes a wide variety of investigation and detection functions to provide:

- Checks for specific protocol attributes and behaviors
- Analysis of session-based communications
- Analysis of all packets exchanged between two end points
- Protocol profiling and session clustering functions to look for network traffic behaviors that do not conform to the norm
- Activity and usage behavior statistics for clients and servers
- Incoming and outgoing communication activity associated with a single node

**VISUALIZATION:** At every stage of the analysis, the NTE offers either text or graphical-based feedback. There is a diversified set of graphs and visualization functions. The ability to see an attack signature is useful when planning or developing countermeasures to known types of attacks.

**MANAGEMENT AND CONTROL:** A wide variety of support, management and control functions are provided. There is a standard model for adding GUI interfaces. The model allows a user to access internal functions through a complete GUI environment, or a command line environment. Each data input field is assigned a class type which allows the tool to know which variables on your desktop fit a given function parameter field. The NTE tracks a user's activity and recommends potential analysis options.

### 3 Environment

Many security products have a closed environment. Most vendors do not allow end users to add their own anomaly detection algorithms to a product. In contrast, the NTE tool is an open and flexible environment. It has several hundred functions for helping an analyst recognize network-based abnormalities, attacks and intrusions. These functions were written in a way that allows the analyst to “surf” through large volumes of data and display the results in a graphical window. By providing a suite of visualization functions, the end user can understand the nature of an attack and the validity of a given security event.

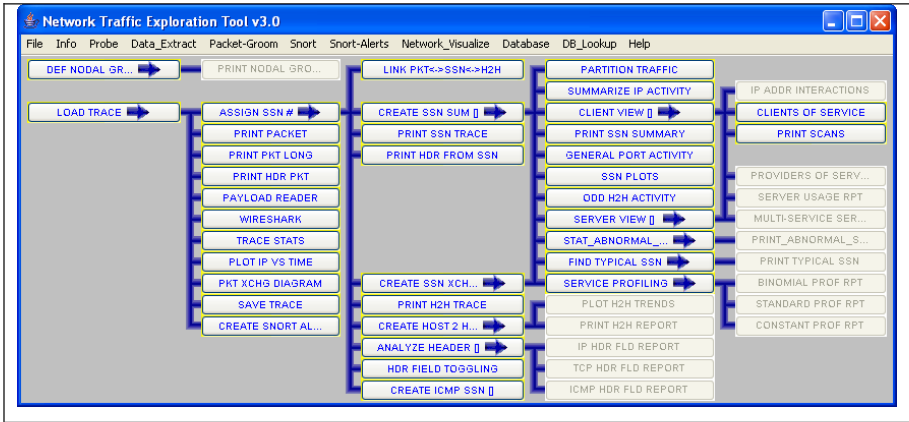


Fig. 8. NTE User Interface

NTE does not require that the analyst learn the hundreds of function calls. To facilitate use, an interactive GUI environment was created as shown in Figure 8. The buttons on the GUI show the options available to the user to solve a given problem. It is important to note that the GUI environment is optional and the user can enter or leave it at will. If there is a test or function that is not available, the user can swap back and forth between the GUI and the MATLAB command line environment without losing environment context or desktop variables.

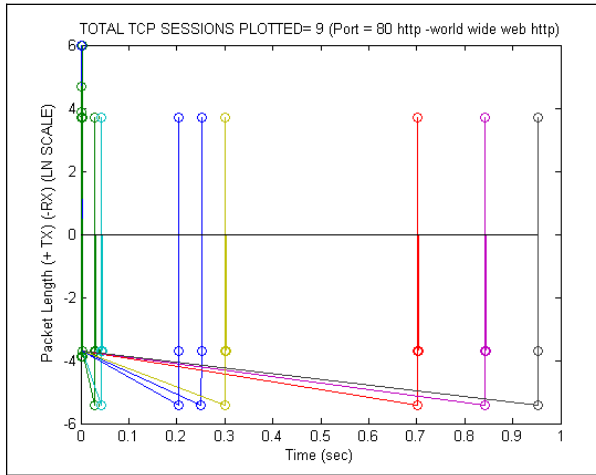
To create scripts using the NTE the analyst must indicate the general approach to be taken in analyzing the problem by pressing the various GUI buttons. As the buttons are pressed, the tool builds the actual working program in the background. When the user is finished, the script can be saved to disk or displayed to screen for verification. The custom scripts can be reused in subsequent analysis sessions by entering the script name or command line or pasted into larger programs.

## 4 Preliminary Evaluation of Covert Tunnel Detection Techniques

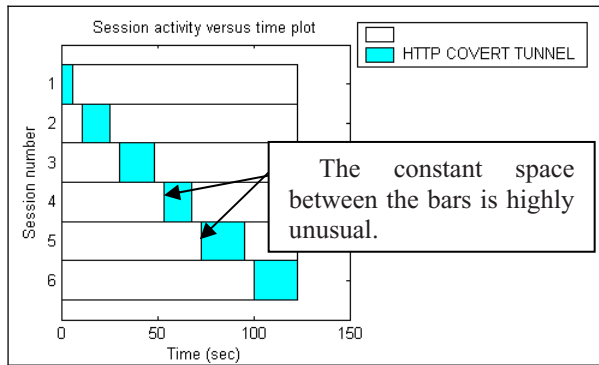
Covert tunnels or covert channels allow an attacker to communicate with another system through a means that it was not intended or designed to do [35]. The most common tunnels today are HTTP, but ICMP and DNS tunnels also exist [36-38]. To demonstrate the NTE, a series of HTTP and ICMP tunnel applications were downloaded from the Internet. Some of the signatures have been described [39-47] and the tools to detect some of these covert tunnels are described in [39, 43]. The emphasis here is to demonstrate the breadth of analysis and the way in which the tunnel can be spotted using the built in visualization functions of the NTE.

**EXAMPLE 1 – ICMP TUNNEL:** The ICMP protocol typically follows a query-response (ping) or is used to send a single packet error message. There are a variety of ways to detect an ICMP covert tunnel, namely:





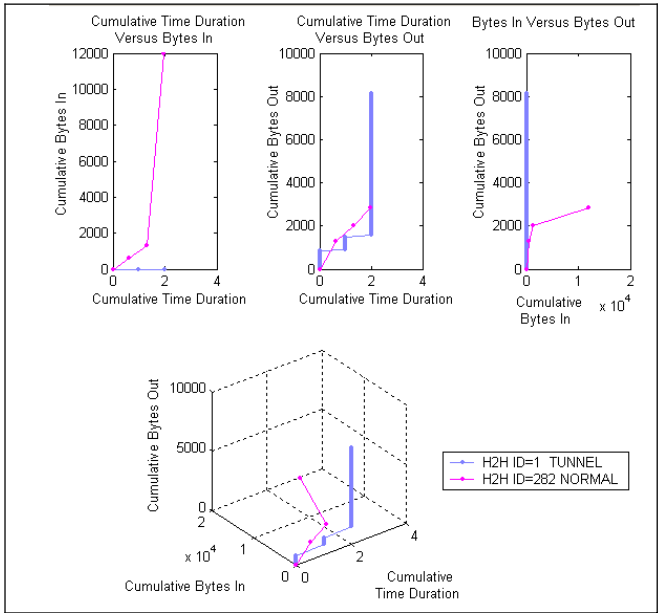
**Fig. 10.** Overlay Packet Exchange Diagram for a sample spyware - Data points above the black horizontal line represent data transmitted and points below are data received



**Fig. 11.** GANTT Diagram for a sample In-band HTTP Tunnel

to network delays. Note how the packet exchange is exactly the same. Other visualizations show that communication is periodic as one host repeatedly tries to the same messages to three different websites (as the spyware attempts to connect to several home bases simultaneously).

**EXAMPLE 3 – IN-BAND HTTP TUNNEL:** There are several types of of HTTP tunnels. In-band tunnels hide their data in the payload while out-of-band tunnels communicate using the packet/protocol header. HTTP in-band tunnels are more difficult to detect because the HTTP protocol is very flexible. HTTP in-band tunnels are frequently machine driven, with multiple session exchanges showing little diversity in structure or timing characteristics. A sample HTTP tunnel [49] is shown in Figure 11. Each bar on this gantt chart represents an HTTP session. The tunnel



**Fig. 12.** Host-to-host Diagram for a sample out-of-band HTTP Tunnel

signature is in the constant space between the bars. Typically an end user does not wait X seconds from the end of one HTTP session before initiating the next.

**EXAMPLE 4 – OUT-OF-BAND HTTP TUNNEL:** HTTP out-of-band tunnels manipulate the packet header fields to send messages in one or both directions. In this example [50], the IP ID field is being manipulated to send commands in one direction (there is no payload, only a protocol header) and HTTP messages are used to send data back. The strange mix of incomplete and complete session patterns result in a very distinctive multi-session profile. Figure 12 shows the multi-session host-to-host graph with a covert tunnel and a single normal HTTP session for illustration. In this case, the tunnel looks like a step function while the regular traffic appears as a wandering line.

**EXAMPLE 5 – AUTOMATED DETECTION USING NTE CODE:** The signatures in the previous 4 examples were extracted manually by an analyst. In this example, the conversion of that experience to an automated response is demonstrated. Specifically, this example shows how code could be generated by the NTE to detect the out-of-band HTTP tunnel (see Figure 13). The code associated with plotting the data has been removed. The code (including comments) was generated as a standard part of the problem solving process and is executable as a standard MATLAB program. The program can be summarized as follows:

- Load a trace (load\_pcap\_file\_plus)
- Enumerate the communication sessions (assign session number)



```

% Load the the PCAP trace array
[PKT_DATA,IP_OPT,TCP_OPT,PAYLOAD]=load_pcap_file_plus_ev(...
    'C:\DATA \good_bad_2.dmp','TRACE_NUMBER','', 'BPF','ip');

% Assign session id numbers and host-2-host id numbers
% to the trace array
PKT_IDX=assign_session_number(PKT_DATA,PAYLOAD);

% Create the host to host array
[H2H_SUM,H2H_LIST]=create_h2h_array(PKT_DATA,PKT_IDX);

% Print a summary of the host to host connections
[SSN_NUM,H2H_NUM]=print_host_to_host_details(H2H_SUM, ...
    H2H_LIST,'ALL',5,'TTL_PKTS=1&SERV_PORT=80&PROTOCOL=6',50);

% Convert host to host connections back to packets
[PKT_REF_NUM]=find_pkt_from_h2h(PKT_IDX,H2H_NUM);

% Create SNORT alert
[S_ALERT,S_CPAYLOAD,S_PAYLOAD]=create_snort_alert_ev(...
    PKT_DATA,PKT_REF_NUM(1),'GID',1,'SID',6969,'VER',1, ...
    'MSG','Out of Band Covert Tunnel','PRIORITY',1);

%Store the SNORT alert
save_snort_alert_ev('my file',S_ALERT,S_CPAYLOAD,S_PAYLOAD);

```

**Fig. 13.** Code generated by the NTE during the problem solving process to spot an out-of-band HTTP tunnel

```

07/09-11:25:00.381048 [**] [1:6969:1] Out of Band
Covert Tunnel [**] [Classification: ] [Priority: 1]
{TCP} 192.168.6.3:1234 -> 192.168.6.2:80

```

**Fig. 14.** SNORT alert generated by the NTE

- Compile a list of all systems which spoke together (builds the host-to-host array)
- Search the intercommunication arrays for high numbers of unrequited communication sessions between two hosts that communicate using HTTP (print\_host\_to\_host\_details)
- Convert activity that passes this test into packets (find\_pkt\_from\_h2h) (note if array is empty it is ignored)
- Convert packets to a SNORT alert which is saved to disk.

The function calls contained in the listing are standard NTE function calls. When this program is run, no GUI appears and the code outputs events to a standard SNORT alert file (see Figure 14). Note that the analyst can still modify this code and add extra features using either the NTA toolbox, NTE or using any of the MATLAB prepackaged function calls.

## 5 Discussion and Conclusion

The covert tunnel detection demonstration shows that NTE can handle a wide range of situations. The tool's broad range of graphs, reports, and visualization features offers a high level of insight into security events.

The tool allows an analyst to examine a particular type of attack and recognize signatures that distinguish it from routine traffic. Through the process of signature detection and analysis, it is capable of writing the code in the background necessary to detect future attacks or eliminate false positives.

NTE includes a wide range of functions and is easy to use and extend. The tool provides good analyst support via an interactive GUI, detailed user guide, individual help pages for functions as well as a unique tracking and guidance system. There is a detailed technical manual available to guide developers through each step of the function addition process. The tiered GUI system allows custom GUI front-ends for different users while maintaining a single backend.

NTE allows the user to store data in a desktop environment and port data easily between different functions. Linkage tools exist to facilitate comparison between different data types. The data query language allows the analyst to find data quickly and easily. The correlation functions allow the analyst to match multiple rows of data across multiple data sets.

Currently, NTE comes equipped with interfaces to external tools such as SNORT and Wireshark. The tool can read and write PCAP formatted files as well as SNORT's full and fast alerts. Future extensions could include fully automated event analysis and passive host-fingerprinting. These additions would allow an analyst to respond to network events faster and minimize repetitive analysis tasks.

## References

- [1] Valeur, F., et al.: A Comprehensive Approach to intrusion Detection Alert Correlation. *IEEE Transactions on Dependable and Secure Computing* 1(3), 146–149 (2004)
- [2] Farshchi, J.: Statistical based approach to Intrusion Detection, SANS Institute(2003) (Access date 1 April 2008), [http://www.sans.org/resources/idfaq/statistic\\_ids.php](http://www.sans.org/resources/idfaq/statistic_ids.php)
- [3] Roesch, M.P: SNORT (Access date 1 April 2008), <http://www.snort.org/>
- [4] Ertöz, L., Eilerston, E. Lazarevic, A., Tan P. Srivastava, J. and Kumar, V.: Detection and Summarization of Novel Network Attacks Using Data Mining, Technical Report (2003), <http://www-users.cs.umn.edu/~aleks/MINDS/papers/raid03.pdf>
- [5] Chakchai, S.: A Survey of Network Traffic Monitoring and Analysis Tools, (2006) (Access date 1 April 2008), <http://www.cse.wustl.edu/~cs5/567/traffic/index.html>
- [6] Ranum, M.: Packet Peekers, *Information Security Magazine*, p. 28 (2003)
- [7] Keshav, T.: A Survey of Network Performance Monitoring Tools (2006)(Access date 1 April 2008), [http://www.cs.wustl.edu/~jain/cse567-06/ftp/net\\_perf\\_monitors1.pdf](http://www.cs.wustl.edu/~jain/cse567-06/ftp/net_perf_monitors1.pdf)
- [8] Fortunato, T.: The Technology Firm, web page (2007), <http://www.thetechfirm.com/reviews/>

- [9] Lyon, G.: Top 100 Security Tools, Insecure.org (2006), <http://www.insecure.org/tools.html>
- [10] Bejtlich, R.: The Tao of Network Security Monitoring: Beyond Intrusion Detection, pp. 105–344. Addison-Wesley, Boston (2005)
- [11] Vissler, R.: SGUIL (2007) (Access date 2 April 2008) , <http://sguil.sourceforge.net/>
- [12] Combs, G., et al.: wireshark (2008) (Access date 2 April 2008), <http://www.wireshark.org/>
- [13] Zalewski, M.: P0f (2006) (Access date 2 April 2008), <http://lcamtuf.coredump.cx/p0f.shtml>
- [14] Elson, J.: tcpflow (2003) (Access date 2 April 2008), <http://www.circlemud.org/~jelson/software/tcpflow>
- [15] Jacobson, V., et al.: Libpcap (2007) (Access date 2 April 2008), <http://www.tcpdump.org/>
- [16] Jacobson, V., Leres, C., and McCanne, S.: tcpdump (2007) (Access date 2 April 2008), <http://www.tcpdump.org/>
- [17] OPNET ACE Application Characterization Environment (2007) (Access date 2 April 2008), <http://www.opnet.com/solutions/brochures/Ace.pdf>
- [18] Paxon, V.: BRO (2007) (Access date 2 April 2008), <http://bro-ids.org/>
- [19] Computer Associates, eHealth (2008) (Access date 2 April 2008), <http://www.ca.com/us/products/product.aspx?ID=5637>
- [20] Kohler, E.: ipsumdump (2006) (Access date 2 April 2008), <http://www.cs.ucla.edu/~kohler/ipsumdump/>
- [21] Ritter, J.: ngrep (2006) (Access date 2 April 2008), <http://ngrep.sourceforge.net/>
- [22] Combs, G., et al.: editcap/mergcap (2008) (Access date 2 April 2008), <http://www.wireshark.org/>
- [23] Astashonok, S.: Fprobe (2005) (Access date 2 April 2008), <http://sourceforge.net/projects/fprobe>
- [24] Ostermann, S.: tcptrace (2003) (Access date 2 April 2008), <http://www.tcptrace.org/>
- [25] Deri, L.: ntop (2008) (Access date 2 April 2008), <http://www.ntop.org/>
- [26] Postel, J.: RFC 792 - Internet Control Message Protocol, (1981) (Access date 2 April 2008), <http://www.faqs.org/rfcs/rfc792.html>
- [27] Kreibich, C.: netdude (2007) (Access date 2 April 2008), <http://netdude.sourceforge.net/>
- [28] Fullmer, M.: flow-tools (2005) (Access date 2 April 2008), <http://www.splintered.net/sw/flow-tools/docs/flow-tools.html>
- [29] Walkin, L.: ipcad (2007) (Access date 2 April 2008), <http://sourceforge.net/projects/ipcad/>
- [30] Curry, J.: SANCP (2003) (Access date 2 April 2008), <http://www.metre.net/sancp.html>
- [31] Kernen, T.: Traceroute (2008) (Access date 2 April 2008), <http://www.traceroute.org/>
- [32] Fenner, B.: tcpslice (2002) (Access date 2 April 2008), <http://sourceforge.net/projects/tcpslice/>
- [33] Buylard, C.: Argus, (2008) (Access date 2 April 2008), <http://www.qosient.com/argus>

- [34] Cho, K., Dittrich, D.: *tcpdstat* (2000), <http://staff.washington.edu/dittrich/talks/core02/tools/tools.html>
- [35] Naval Research Laboratory, “Handbook for the Computer Security Certification of Trusted Systems”, Technical Memorandum 5540, 062A (1996)
- [36] Temmingh, R.: *Setiri: Advances in Trojan Technology* (2002) (Access date 2 April 2008), <http://www.blackhat.com/presentations/bh-asia-02/Sensepost/bh-asia-02-sensepost.pdf>
- [37] Smith, J.: *Covert Shells* (2000) (Access date 2 April 2008), [http://www.s0ftpj.org/docs/covert\\_shells.htm](http://www.s0ftpj.org/docs/covert_shells.htm)
- [38] Kieltyka, P.: *ICMP Shell* (2002) (Access date 3 April 2008), <http://sourceforge.net/projects/icmpshell>
- [39] Borders, K.: *Web Tap: Detecting Covert Web Traffic*. In: *Proceedings of the 11th ACM conference on Computer and communications security*, pp. 110–120. ACM, Washington (2004)
- [40] Northcutt, S., Novak, J.: *Network Intrusion Detection, An Analyst’s Handbook*, New Riders, Indianapolis, Indiana, pp. 63–65 (2000)
- [41] Northcutt, S., Cooper, M., Fearnow, M., Fredrick, K.: *Intrusion Signatures and Analysis*, New Riders, Indianapolis, Indiana, p. 137 (2001)
- [42] Knight, G., et al.: *Detecting covert tunnels within the hypertext transfer protocol* (2003), [http://www.rmc.ca/academic/gradrech/abstracts/2003/ece2003-2\\_e.html](http://www.rmc.ca/academic/gradrech/abstracts/2003/ece2003-2_e.html)
- [43] Castro, S.: *Covert Channel and Tunneling over the HTTP protocol Detection: GW implementation theoretical design* (2003), <http://www.infosecwriters.com/hhworld/cctde.html>
- [44] Dyatlov, A.: *Exploitation of data streams authorized by a network access control system for arbitrary data transfers: tunneling and covert channels over HTTP protocol* (2003) (Access date 2 April 2008), <http://www.net-security.org/dl/articles/covertpaper.txt>
- [45] Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., Karger, D.: *Infranet: Circumventing Web Censorship and Surveillance*. In: *11th USENIX Security Symposium*, San Francisco, CA (2002)
- [46] Crotti, M., Dusi, M., Gringoli, F., Salgarelli, L.: *Detecting HTTP Tunnels with Statistical Mechanisms*. In: *ICC 2007. IEEE International Conference on Communications*, pp. 6162–6168 (2007)
- [47] Castro, S.: *Cctde - Covert Channel and Tunneling Over the HTTP Protocol Detection* (2003) (Access date 2 April 2008), <http://gray-world.net/projects/papers/html/cctde.html>
- [48] Vecna. *PacketStorm - 007Shell.tgz* (1999) (Access date 2 April 2008), <http://packetstormsecurity.org/groups/s0ftpj/>
- [49] Rowland, C.: *Covert Channels in the TCP/IP Protocol Suite* (1996) (Access date 2 April 2008), [http://www.firstmonday.dk/issues/issue2\\_5/rowland/](http://www.firstmonday.dk/issues/issue2_5/rowland/)
- [50] Hauser, V.: *Reverse-WWW-Tunnel-Backdoor v1.6* (1998) (Access date 2 April 2008), <http://packetstormsecurity.org/groups/thc/rwwwshell-1.6.perl>

# Author Index

- Blue, Ryan 119  
Bratus, Sergey 152
- Carver, Jeffery 80  
Conti, Gregory 1
- D'Amico, Anita D. 136  
Dampier, David 80  
Dean, Erik 1  
Dunne, Cody 119  
Dymacek, Tomas 144
- Fairbanks, Kevin 26  
Fischer, Fabian 111  
Franck, Josh 80  
Fuchs, Adam 119
- Goodall, John R. 136
- Hansen, Axel 152  
Heitzmann, Alexander 18  
Homer, John 68
- Ingols, Kyle 44
- Janies, Jeff 161  
Jankun-Kelly, T.J. 80
- Keim, Daniel A. 111  
King, Kyle 119  
Kohlenberg, Toby 95  
Kopylec, Jason K. 136
- Liebrock, L.M. 36  
Lippmann, Richard 44
- Ma, Kwan-Liu 95  
Mansmann, Florian 111  
McQueen, Miles A. 68
- Minarik, Pavel 144  
Musa, Shahrulniza 169
- Noel, Steven 60
- O'Hare, Scott 60  
Ou, Xinming 68  
Owen, Henry 26  
Owen, Scott 87
- Palazzi, Bernardo 18  
Papamantou, Charalampos 18  
Parish, David J. 169  
Pellacini, Fabio 152  
Pietzko, Stephan 111  
Prole, Kenneth 60, 136
- Sangster, Benjamin 1  
Schulman, Aaron 119  
Schwartz, Moses 36  
Shearer, James 95  
Shubina, Anna 152  
Sinda, Matthew 1  
Suo, Xiaoyuan 87  
Swan II, J. Edward 80
- Tamassia, Roberto 18
- Vandenberghe, Grant 181  
Varikuti, Ashok 68
- Waldvogel, Marcel 111  
Williams, Leever 44  
Wilson, David 80
- Xia, Ying 26
- Zhu, Ying 87