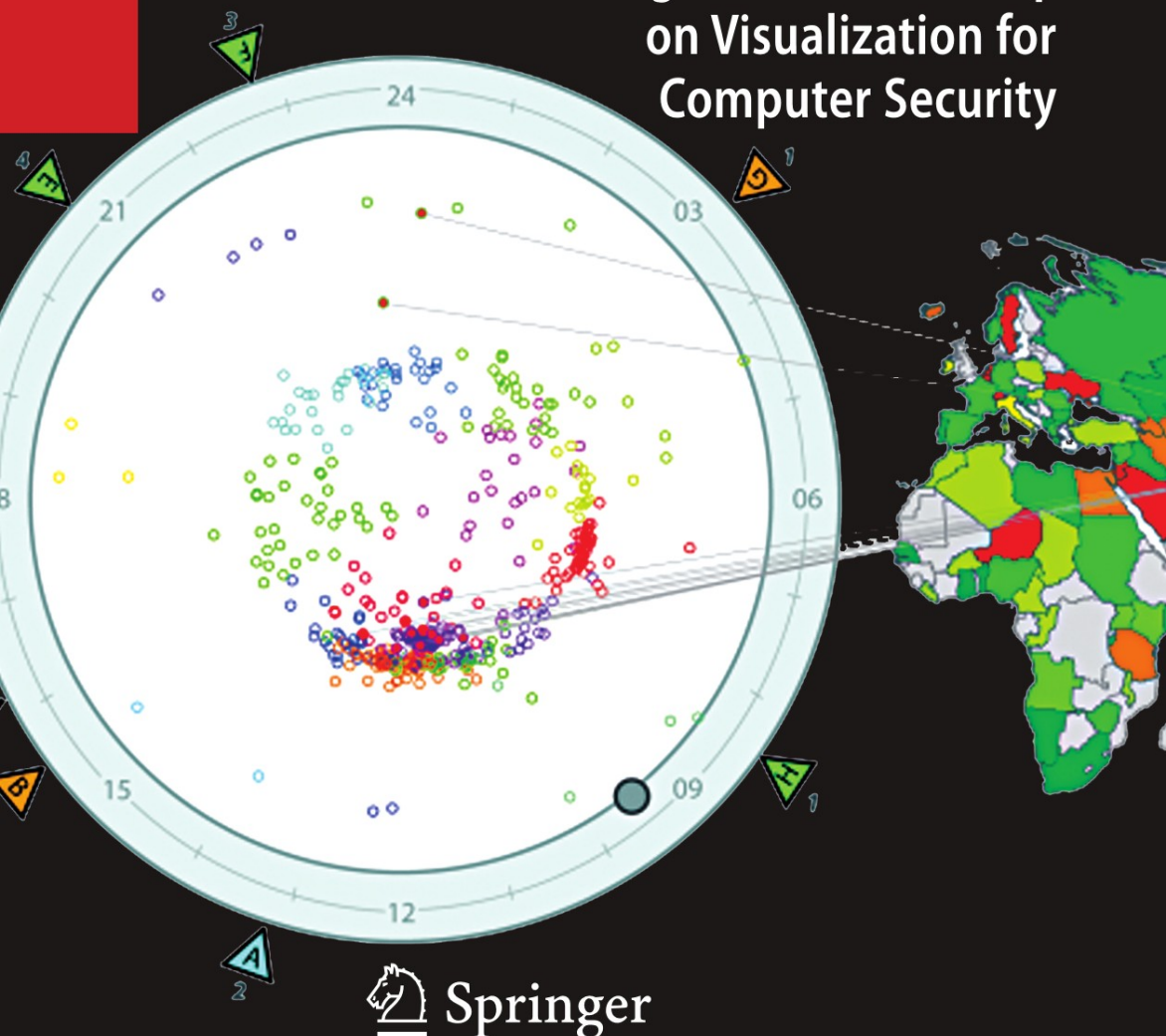


John R. Goodall Gregory Conti
Kwan-Liu Ma Editors

Mathematics + Visualization

VizSEC 2007

Proceedings of the Workshop
on Visualization for
Computer Security



 Springer

Mathematics and Visualization

Series Editors

Gerald Farin

Hans-Christian Hege

David Hoffman

Christopher R. Johnson

Konrad Polthier

Martin Rumpf

John R. Goodall
Gregory Conti
Kwan-Liu Ma

Editors

VizSEC 2007

Proceedings of the Workshop on Visualization
for Computer Security

With 140 Figures, 117 in Color and 5 Tables

 Springer

Editors

John R. Goodall

Secure Decisions Division
Applied Vision, Inc.
6 Bayview Ave.
Northport NY 11768, USA
johnhg@securedesigns.avi.com

Gregory Conti

Department of Electrical Engineering
and Computer Science
United States Military Academy
West Point, NY 10996, USA
gjconti@rumint.org

Kwan-Liu Ma

Department of Computer Science
University of California
One Shields Avenue
Davis, CA 95616, USA
ma@cs.ucdavis.edu

ISBN 978-3-540-78242-1

e-ISBN 978-3-540-78243-8

Mathematics and Visualization ISSN 1612-3786

Library of Congress Control Number: 2008924865

Mathematics Subject Classification (2001): 68-06, 68U05

© 2008 Springer-Verlag Berlin Heidelberg

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, reuse of illustrations, recitation, broadcasting, reproduction on microfilm 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.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: WMX Design GmbH

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

This volume is a collection of the papers presented at the 4th International Workshop on Computer Security – VizSec 2007. The workshop was held in conjunction with the IEEE Visualization 2007 Conference and the IEEE InfoVis Conference in Sacramento, California on October 29, 2007.

This volume includes an introductory chapter and two chapters from the workshop’s invited speakers: *The Real Work of Computer Network Defense Analysts* by Anita D’Amico and Kirsten Whitley, and *VisAlert: From Idea to Product* by Stefano Foresti and Jim Agutter. All other papers were peer-reviewed by the VizSec program committee.

January 2008

John R. Goodall

Acknowledgements

We thank the VizSec 2007 Sponsor, NSA's National Information Assurance Research Laboratory (NIARL), and the VizSec committee members:

Workshop Chair

- John R. Goodall, Secure Decisions division of Applied Visions, Inc.

Program Co-Chairs

- Kwan-Liu Ma, University of California at Davis
- Gregory Conti, United States Military Academy

Program Committee

- Kulsoom Abdullah, Georgia Institute of Technology
- Jim Agutter, University of Utah
- Stefan Axelsson, Blekinge Institute of Technology
- Anita D'Amico, Secure Decisions
- Glenn Fink, Pacific Northwest National Laboratory
- Deborah Frincke, Pacific Northwest National Laboratory
- John Gerth, Stanford University
- Patrick Hertzog, NEXThink S.A.
- Kiran Lakkaraju, University of Illinois at Urbana-Champaign
- Yarden Livnat, University of Utah
- Raffael Marty, Splunk
- Daniel Keim, University of Konstanz
- Stephen North, AT&T Research
- Penny Rheingans, UMBC
- Walt Tirenin, Air Force Research Laboratory
- Soon Tee Teoh, San Jose State University
- Kirsten Whitley, Department of Defense

Contents

- Introduction to Visualization for Computer Security** 1
- J.R. Goodall
 - 1 Computer Security 1
 - 2 Information Visualization 3
 - 3 Visualization for Computer Network Defense 5
 - 3.1 Data Sources for Computer Network Defense 6
 - 3.2 VizSec to Support Computer Network Defense 6
 - 4 Papers in This Volume 11
 - 4.1 Users and Testing 11
 - 4.2 Network Security 13
 - 4.3 Communication, Characterization, and Context 14
 - 4.4 Attack Graphs and Scans 15
 - 5 Conclusion 15
 - References 16
- The Real Work of Computer Network Defense Analysts** 19
- A. D’Amico and K. Whitley
 - 1 Introduction 19
 - 2 Related Work 20
 - 3 Methods 22
 - 4 Findings 23
 - 4.1 Data Transformation in CND Analysis 24
 - 4.2 CND Analysis Roles 27
 - 4.3 CND Analysis Workflow Across Organizations 29
 - 5 Implications for Visualization 33
 - 5.1 Visualization Across the CND Workflow 33
 - 5.2 Visualization as Part of a CND Analysis Environment 35
 - References 36

Adapting Personas for Use in Security Visualization Design	39
J. Stoll, D. McColgin, M. Gregory, V. Crow, and W.K. Edwards	
1 Introduction	39
2 Overview of the Personas Method and Related Work	40
2.1 Personas Method	41
2.2 Related Work	42
3 Case Study: First Look	43
3.1 Five Steps to Persona Implementation	43
3.2 Discussion	49
4 Application to Security Visualizations	49
5 Conclusion	51
References	51
Measuring the Complexity of Computer Security Visualization Designs	53
X. Suo, Y. Zhu, and G. Scott Owen	
1 Introduction	53
2 Related Work	54
3 Technical Approach	55
3.1 Hierarchical Analysis of Data Visualization	57
3.2 Visual Integration	57
3.3 Separable Dimensions for Visual Units	58
3.4 Interpreting the Values of Visual Attributes	60
3.5 Efficiency of Visual Search	61
3.6 Case Study with RUMINT	63
4 Future Work	65
5 Conclusion	65
References	66
Integrated Environment Management for Information Operations Testbeds	67
T.H. Yu, B.W. Fuller, J.H. Bannick, L.M. Rossey, and R.K. Cunningham	
1 Introduction	67
2 Related Work	68
3 Technical Approach	70
3.1 LARIAT Overview	70
3.2 Design Goals	72
3.3 Interface and Visualization	72
4 Future Work	80
5 Conclusions	81
References	82

Visual Analysis of Network Flow Data with Timelines and Event Plots	85
D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd	
1 Introduction	85
2 Network Flow Data	86
2.1 Flow Sensor	86
2.2 Database Repository	87
3 The Investigation Process	87
4 Flow Maps	88
5 Progressive Multiples of Timelines and Event Plots	89
6 A Case of Mysterious IRC Traffic	90
7 Related Work	96
8 Future Work and Conclusions	98
References	98
NetBytes Viewer: An Entity-Based NetFlow Visualization Utility for Identifying Intrusive Behavior	101
T. Taylor, S. Brooks, and J. McHugh	
1 Introduction	101
2 Related Work	102
3 Technical Approach	105
3.1 NetBytes Viewer User Interface	105
3.2 User Interaction	107
3.3 Implementation Details	110
3.4 Case Studies	110
4 Future Work	113
5 Conclusions	114
References	114
Visual Analysis of Corporate Network Intelligence: Abstracting and Reasoning on Yesterdays for Acting Today	115
D. Lalanne, E. Bertini, P. Hertzog, and P. Bados	
1 Introduction	115
2 Background	117
3 On the Need to Support Visual Analysis	118
3.1 Types of Analyses	120
3.2 Analysis Tasks	120
4 User and Application Centric Views of the Corporate Network	122
4.1 The RadViz: Visually Grouping Similar Objects	122
4.2 The OriginalityView: Plotting the Uncommon	124
5 Alarm/Event Centric Views	126
6 Limitations and Challenges	128
7 Conclusion	129
References	129

Visualizing Network Security Events Using Compound Glyphs From a Service-Oriented Perspective	131
J. Pearlman and P. Rheingans	
1 Introduction	131
2 Related Work	133
3 Technical Approach	134
3.1 Network Node Glyph	134
3.2 Layout	136
3.3 Comparing to a Model	137
3.4 Results	138
4 Future Work	144
5 Conclusions	145
References	145
 High Level Internet Scale Traffic Visualization Using Hilbert Curve Mapping	147
B. Irwin and N. Pilkington	
1 Introduction	147
2 Related Work	148
3 Technical Approach	150
4 Results	151
4.1 Output Analysis	153
4.2 Other Applications	154
5 Future Work	156
6 Conclusions	157
References	158
 VisAlert: From Idea to Product	159
S. Foresti and J. Agutter	
1 Introduction	159
1.1 The Project and Team	160
1.2 The VisAlert Metaphor	160
2 Related Work	161
2.1 Visualization of Network Security	161
2.2 Design	162
2.3 Inter-Disciplinary Collaboration	163
3 Technical Approach	163
3.1 The Team Dynamics	163
3.2 The Design Process	164
3.3 Sketches	165
3.4 Refined Conceptual Ideas	167
3.5 Implementation	169
4 Future Work	171
5 Conclusions	172
References	174

Visually Understanding Jam Resistant Communication	175
D. Schweitzer, L. Baird, and W. Bahn	
1 Introduction	175
2 Related Work	176
2.1 BBC and Concurrent Codes	177
2.2 BBC Implementations	178
3 Technical Approach	179
3.1 An Audio Solution	179
3.2 A Visual Representation	180
4 Future Work	184
5 Conclusions	185
References	186
Visualization of Host Behavior for Network Security	187
F. Mansman, L. Meier, and D.A. Keim	
1 Introduction	187
2 Related Work	189
2.1 Analysis of Application Ports	190
2.2 Graph-Based Approaches for Network Monitoring	190
2.3 Towards Visual Analytics for Network Security	191
2.4 Summary	191
3 Technical Approach	191
3.1 Layout Details	193
3.2 Implementation	194
3.3 User Interaction	194
3.4 Abstraction and Integration of the Behavior Graph in HNMap	196
3.5 Application and Evaluation	197
4 Future Work	200
5 Conclusions	200
References	201
Putting Security in Context: Visual Correlation of Network Activity with Real-World Information	203
W.A. Pike, C. Scherrer, and S. Zabriskie	
1 Introduction	203
2 Related Work	204
2.1 The Importance of Maintaining Context	204
2.2 Visualizing Packets and Flows	205
2.3 Visualizing Correlated Activity	206
3 Technical Approach	206
3.1 “I Just Want to Know Where to Focus My Time”	207
3.2 “We Need to Organize Our Hay into Smaller Piles”	208
3.3 Behavior Modeling	209
3.4 Building Context	213
3.5 Visualizing Behavior in Context	214

4	Future Work	217
5	Conclusions	218
	References	219
An Interactive Attack Graph Cascade and Reachability Display		221
L. Williams, R. Lippmann, and K. Ingols		
1	Introduction	221
2	Related Work	222
	2.1 Limitations of Existing Approaches	222
	2.2 NetSPA System	223
3	Technical Approach	224
	3.1 Design Goals	225
	3.2 Initial System Design	225
	3.3 Example Network Results	227
	3.4 Field Trial Results	230
4	Future Work	232
5	Conclusions	234
	References	235
Intelligent Classification and Visualization of Network Scans		237
C. Muelder, L. Chen, R. Thomason, K.-L. Ma, and T. Bartoletti		
1	Introduction	237
2	Related Work	239
3	Technical Approach	240
	3.1 Scan Data and Representation	241
	3.2 An Intelligent Method	242
	3.3 Visualization Integration	246
	3.4 A Case Study	249
4	Future Work	250
5	Conclusions	251
	References	252
Using InetVis to Evaluate Snort and Bro Scan Detection on a Network Telescope		255
B. Irwin and J.-P. van Riel		
1	Introduction	255
	1.1 The Merits and Difficulties of Scan Detection	256
2	Related Work	257
	2.1 Intrusion Detection and the False Positive Problem	257
	2.2 Network Telescopes	257
	2.3 Classifications of Network Scan Activity	258
	2.4 Algorithmic Approaches to Scan Detection	258
	2.5 Network Security Visualisation	259
3	InetVis Network Traffic Visualisation	259
	3.1 Key Features and Enhancements	260
4	Investigative Methodology	261

4.1	Network Telescope Traffic Capture	262
4.2	Scan Detection Configuration and Processing	262
4.3	Graphical Exploration and Investigation with InetVis	264
5	Results and Analysis	264
5.1	Address Scans and the Distribution of Unique Addresses	265
5.2	Scans Discovered and Characterised with InetVis	266
6	Future Work	270
7	Conclusion	271
	References	271

Introduction to Visualization for Computer Security

J.R. Goodall

Abstract Networked computers are ubiquitous, and are subject to attack, misuse, and abuse. Automated systems to combat this threat are one potential solution, but most automated systems require vigilant human oversight. This automated approach undervalues the strong analytic capabilities of humans. While automation affords opportunities for increased scalability, humans provide the ability to handle exceptions and novel patterns. One method to counteracting the ever increasing cyber threat is to provide the human security analysts with better tools to discover patterns, detect anomalies, identify correlations, and communicate their findings. This is what visualization for computer security (VizSec) researchers and developers are doing. VizSec is about putting robust information visualization tools into the hands of humans to take advantage of the power of the human perceptual and cognitive processes in solving computer security problems. This chapter is an introduction to the VizSec research community and the papers in this volume.

1 Computer Security

In *The Cuckoo's Egg*, astronomer-turned-systems administrator Cliff Stoll (Stoll, 1989) recounted his experience identifying and tracking a hacker through the nascent Internet in the mid-1980s. Through perseverance, creativity (he once dangled his keys over the telephone modem lines to create interference to slow down and frustrate the intruder), and extensive coordination and collaboration with other systems administrators, Stoll's actions led to the uncovering of an international spy ring that had infiltrated U.S. military systems. The intruder was initially detected from a 75 cent accounting error.

J.R. Goodall

Secure Decisions Division of Applied Visions, Inc., 6 Bayview Ave. Northport, NY 11768, USA,
e-mail: johng@securedecisions.avi.com

In the two decades since Stoll's investigation, computer security has become an overriding concern of all types of organizations. New systems and protocols have been developed and adopted to prevent and detect network intruders. But even with these advances, the central feature of Stoll's story has not changed: humans are still crucial in the computer security process. Administrators must be willing to patiently observe and collect data on potential intruders. They need to think quickly and creatively. They collaborate and coordinate their actions with colleagues. Humans are still as central to computer security today as they were 20 years ago. Technologies have evolved and many security processes have been automated, but the analytic capabilities and creativity of humans are paramount in many security-related practices, particularly in intrusion detection, the focus of this chapter. Because of this, not all security work should be or can be automated. Humans are – and should be – central to security practice. This central feature of computer security is at the core of visualization for computer security (VizSec).

Many things have changed since Stoll's time. In conjunction with the rapid growth of the Internet and increased organizational dependence on networked information technology, the frequency and severity of network-based attacks has increased drastically (Allen et al., 1999). At the same time, there is an inverse relationship between the decreasing expertise required to execute attacks and the increasing sophistication of those attacks; less skill is needed to do more damage (McHugh, 2001). As we have come more and more to rely on the ability to network computers and access information online, attacks are becoming more pervasive, easier to carry out, and more destructive.

Despite this increasing threat and concerted efforts on preventative security measures, vulnerabilities remain. The reasons for these include: programming errors, design flaws in foundational protocols, and the insider abuse problem of legitimate users misusing their privileges (Lee et al., 2000). While it is theoretically possible to remove all security vulnerabilities through formal methods and better engineering practices, practically it remains infeasible (Hofmeyr et al., 1998). Thus, even as security technologies and practices improve, the threat to network infrastructures remains.

Automated systems to combat this threat are one potential solution, but most automated systems require vigilant human oversight. This automated approach undervalues the strong analytic capabilities of humans. While automation affords opportunities for increased scalability, humans provide the ability to handle exceptions and novel patterns. A technical report on intrusion detection technologies noted that while security vendors attempt to fully automate intrusion diagnosis, a more realistic approach is to involve the human in the diagnostic loop; computers can process large amounts of data, but cannot match humans' analytic skills (Allen et al., 1999).

Humans excel at recognizing novel patterns in complex data and computer security support tools should integrate these intricate sense-making capabilities of the human analyst with the ability of technology to process vast quantities of data. In order to effectively support human analysts and keep them in the diagnostic loop, it is necessary to fully comprehend the work security analysts do, how they do it,

and how their work processes can be improved by taking advantage of the inherent strengths of both technology and humans.

One method to counteracting this ever increasing threat is to provide the human security analysts with better tools to discover patterns, detect anomalies, identify correlations, and communicate their findings. This is what VizSec researchers and developers are doing. VizSec is about putting robust information visualization tools into the hands of humans to take advantage of the power of the human perceptual and cognitive processes in solving computer security problems.

2 Information Visualization

Because of the vast amounts of data analysts work with, the need to recognize patterns and anomalies, and the importance of keeping humans in the loop, information visualization shows great potential for supporting computer security work. Put simply, information visualization turns data into interactive graphical displays. Information visualization takes advantage of the highest bandwidth human input device, vision, and human perceptual capabilities. Information visualization can be used for exploration, discovery, decision making, and to communicate complex ideas to others.

Information visualization is distinct from the broader field of data graphics. Information visualization is interactive; the user will have tools to adjust the display in order to gain a more meaningful understanding of the data being presented. Unlike scientific visualization, which is concerned with representing physically based data (such as the human body, molecules, or geography), information visualization represents abstract data; to do so often requires creativity on the designers' part since there is no existing structure to map the data to the graphical display. This is one of the inherent problems in developing an effective information visualization: mapping the data spatially in a meaningful manner. At the core of information visualization is the goal of amplifying cognition, the intellectual processes in which information is obtained, transformed, stored, retrieved, and used (Card, 2003). Information visualization is able to augment cognition by taking advantage of human perceptual capabilities.

Information visualization involves the use of computer-supported, visual representations of abstract data to amplify cognition by taking advantage of human perceptual capabilities (Card et al., 1999). Card, Mackinlay, and Shneiderman (1999) propose six ways that information visualization can amplify cognition: (1) increased resources, (2) reduced search, (3) enhanced recognition of patterns, (4) enabling perceptual inference, (5) using perceptual monitoring, and (6) encoding information in a manipulable medium. Visualization increases memory and processing resources by permitting parallel processing of data and offloading work from the cognitive to perceptual memory. Graphical information displays can often be processed in parallel, as opposed to textual displays, which are processed serially. Visualization shifts the cognitive processing burden to the human perceptual



Fig. 2 The FilmFinder information visualization application combining a starfield display with dynamic queries. ©1994 ACM, Inc. Included here by permission

FilmFinder, shown in Fig. 2, is an early example of an information visualization that highlights the importance of interaction (Ahlberg and Shneiderman, 1994). FilmFinder combines a starfield display, a scatterplot where each data item is represented by a point, with dynamic queries so that the display is continuously updated as the user filters to refine the selection. This is an excellent example of the importance of interaction in information visualization. The display itself is fairly simple, time is plotted on the x axis and ratings on the y axis with color coded to genre. But the dynamic queries through sliders and other widgets prevent user errors and instantly show the results of complex queries. The system is an exemplar of the visual information-seeking mantra: overview first, zoom and filter, then details on demand (Shneiderman, 1996). This approach encourages exploration and understanding of the data set as a whole, while providing a method for drilling down to the actual data details. Many of the VizSec systems described below follow this methodology.

3 Visualization for Computer Network Defense

There are many potential applications of information visualization to the problems of computer security, including:

- Visualization for detecting anomalous activity
- Visualization for discovering trends and patterns
- Visualization for correlating intrusion detection events
- Visualization for computer network defense training
- Visualization for offensive information operations
- Visualization for seeing worm propagation or botnet activity
- Visualization for forensic analysis
- Visualization for understanding the makeup of malware or viruses
- Visualization for feature selection and rule generation
- Visualization for communicating the operation of security algorithms

This is a non-exhaustive list of the kinds of tasks that VizSec tools can be designed to support. Because networks and the Internet are so important to the operations of today's organizations and since the network is the source of most computer based attacks, the majority of VizSec research has targeted supporting the tasks associated

with the defense of enterprise networks from outside attack or insider abuse. This section will focus on the data sources and results of the research into visualization for computer network defense (CND).

3.1 Data Sources for Computer Network Defense

The research of VizSec for CND can be organized according to the level of networking data to be visualized. At the base, most raw level is a network packet trace. A packet consists of the TCP/IP header (which defines how a packet gets from point A to point B) and payload data (the contents of the packet). At a higher level of abstraction is a network flow. Originally developed for accounting purposes, network flows have been increasingly used for computer security applications. A flow is an aggregated record of the communications between two distinct machines. A flow is typically defined by the source and destination Internet Protocol addresses, the source and destination ports, and the protocol. Flows are much more compact than packet traces, but sacrifice details and have no payload data. At a higher level of abstraction are automated systems that reduce network data to information such as an intrusion detection system (IDS). An IDS examines network traffic and automatically generates alerts of suspicious activity. All three of these levels operate on the enterprise network level. At a finer level of granularity is the visualization of data about individual computer systems or applications, and at a higher level is the visualization of data about the Internet.

The remainder of this section will describe a selection of VizSec research that targets the enterprise network level, which is generally the focus of CND.

3.2 VizSec to Support Computer Network Defense

This section presents representative visualization research projects for each of the levels of enterprise network security. The examples presented here each solve an important problem. Rumint facilitates the understanding of packet payloads; tnv allows analysts to move from a high-level overview of packet activity to raw details; NVisionIP enables analysts to use visualization to create automation rules; FlowTag assists collaboration and sharing through tagging of data; VisAlert enables the integration of multiple data sources through a what, where, when paradigm; and IDS Rainstorm highlights the importance of multiple, linked views at different levels of semantic detail.

3.2.1 Packet Trace Visualizations

At the most granular level of enterprise network data are raw packet traces. This kind of data is useful for understanding the behavior of networks and as a supplementary

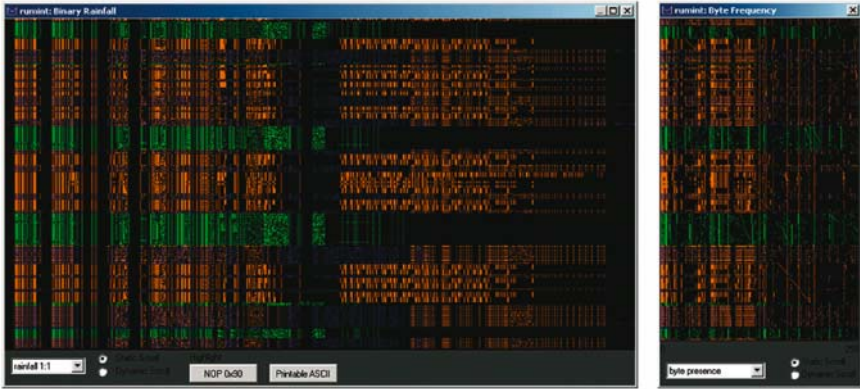


Fig. 3 Rumint visualization: binary rainfall visualization where each row represents a packet and each column in the row represents a bit in the packet (*left*), and byte frequency visualization where each row represents one of 256 byte values and each column in the row represents the frequency of that byte in the packet (*right*). ©2006 IEEE, Inc. Included here by permission

source for analyzing security events, but is typically collected and analyzed on an ad hoc basis, not systematically, since the data can become very large. To help analysts cope with this copious packet data, researchers are looking at ways to visualize packet headers and payloads.

One example is rumint, shown in Fig. 3, which uses a novel visualization called binary rainfall, in which each packet is plotted one per row where each pixel represents a bit in the packet (Conti et al., 2006, 2005). Multiple packets are shown in time series order at multiple semantic levels. An additional view presents a byte frequency visualization, where each packet is plotted on a row where each pixel represents byte values of 0–255. Pixels for each row are drawn according to the frequency of that byte in the packet. The system is unique in that it provides a graphical plotting of packet payload data, plotted according to the bit value. Rumint also includes other views into the data, such as a parallel coordinate plot to show network connections.

Tnv, shown in Fig. 4, is a visualization tool designed to facilitate the analysis processes of CND by providing a visual display that can facilitate recognizing patterns and anomalies over time – thereby increasing support for learning and recognizing normal traffic behavior patterns – coupled with more focused views on packet-level detail that can be understood in the context of the surrounding network traffic (Goodall et al., 2005, 2006). The display is split between three areas. To the left is a narrow area that displays remote hosts, in the center is the area that displays links between hosts, and the large area to the right displays local hosts (those defined as being local to the user), which is divided into a matrix where each row represents a unique local host and each column represents a time interval, with each resulting cell color coded to the number of packets to and from that host within that time period. Bisecting the display to separately show local and remote hosts increased the scalability of the visual display, so that many more hosts can be displayed at

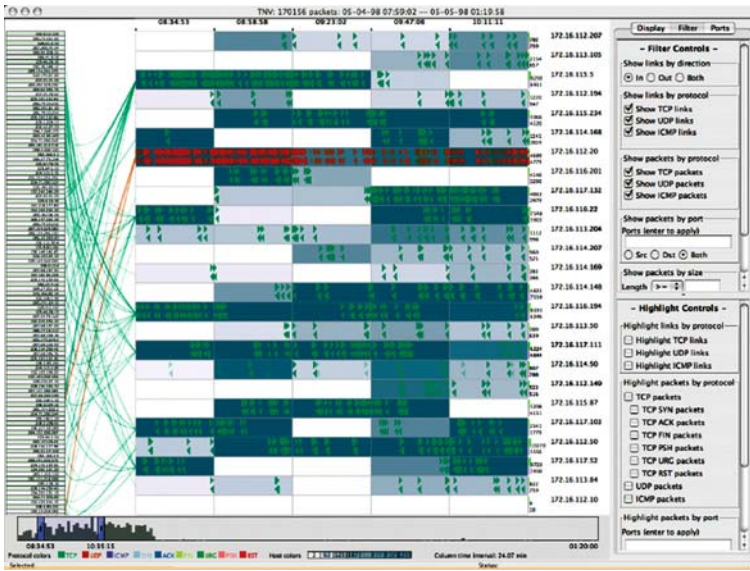


Fig. 4 Ntv visualization showing 170,000 packets. Remote hosts at the left and local hosts at the right of the display, with links drawn between them; packets are drawn for local hosts over time and color is used to represent protocol and packet frequency for a time period

once by dividing the available screen real estate between local and remote hosts. In addition to being able to display more hosts at a time, this partitioning also fits well with analysts’ perceptions of what they deem to be important. Because local hosts are of primary concern in ID analysis, the majority of the display space is devoted to the local hosts. The details of individual packets can be displayed on demand.

3.2.2 Network Flow Visualizations

Network flows are aggregations of packet traces according to the hosts, ports, and protocol involved. Because it is aggregated, flows can be systematically collected and stored, and then used in forensic analysis when an intrusion occurs or monitored for anomalous activity. In either case, the volume of data makes textual analysis difficult and a number of researchers are looking at visualization methods for analyzing flow data.

NVisionIP is geared to increasing an analyst’s situational awareness by visualizing flows at multiple levels of detail (Lakkaraju et al., 2004, 2005). At the highest level of aggregation, NVisionIP, shown in Fig. 5 displays an entire class-B network (65,534 possible addresses) as a scatterplot of colored hosts to facilitate understanding the state of a network. NVisionIP also provides the ability to drill down into the data through a small-multiple view and a histogram of host details. NVisionIP was also extended to “close the loop” by allowing users to create rules from the visualization that can then automatically alert on new data. This concept will likely become

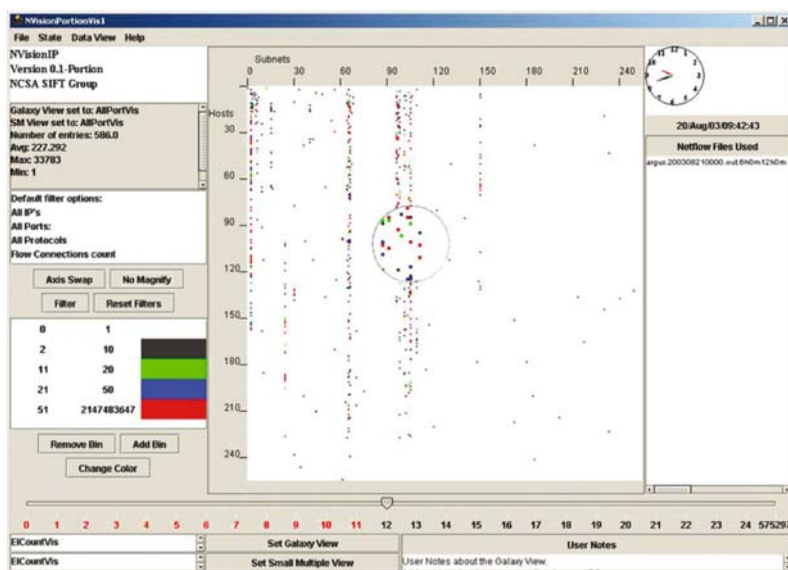


Fig. 5 NVisionIP visualization’s galaxy view, a scatterplot that puts subnets (the third octet of the class-B network) along the x axis and hosts (the fourth octet) along the y axis to present an overview of network flows for a class-B network. Animation can be used to visualize traffic flows over time. ©2004 ACM, Inc. Included here by permission

increasingly common in VizSec applications in the years to come. Machines excel at pattern matching, humans excel at recognizing novel patterns. This approach allows for both machines and humans to do what they do best.

FlowTag, shown in Fig. 6, is a system to visualize network flows and to tag the data to support analysis and collaboration (Lee and Copeland, 2006). Tagging allows analysts to label key elements during the analytic process to reduce the cognitive burden of analysis and maintain context. Tagging can also be used for sharing and collaboration. Tagging has become popular recently with social networking and social bookmarking sites; adapting the concept to CND should be encouraged in all VizSec applications. FlowTag brings the popular concept of tagging to the problems of analyzing and sharing network security data.

3.2.3 Alert Visualizations

Intrusion detection, the process of using computer network and system data to identify potential cyber attacks, has become an increasingly essential component of the information security infrastructure. However, due to the dynamic and complex nature of computer networks and the potential for inappropriate or self-damaging responses to potential attacks, IDSs are only effective when complemented by a human analyst. To help manage the analysis of IDS alerts, several researchers have turned to information visualization.

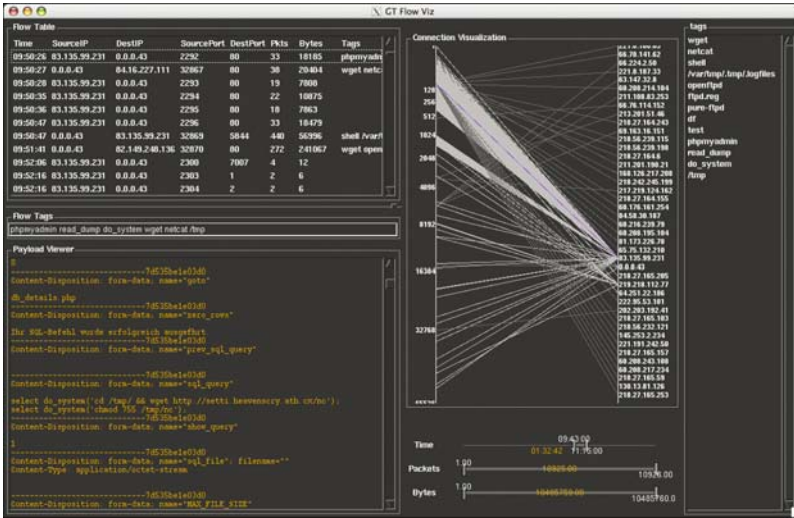


Fig. 6 FlowTag visualization showing flow connection information on a parallel coordinate plot of destination port on one axis and source IP address on the other organized in order of appearance; color represents the selection state. ©2006 ACM, Inc. Included here by permission

VisAlert is a flexible visualization that correlates multiple data sources, such as IDS alerts and system logs files (Livnat et al., 2005a,b). Correlation is based on the What, When and Where attributes of the data. VisAlert, shown in Fig. 7, integrates these into a single display depicting alerts as vectors between the perimeter, representing alert time (when) and type (what), and the interior, representing network topology (where), of a radial view. This system represents one of the more sophisticated and novel visualizations to solve the important problem of correlating disparate events. This is a significant example of a novel approach to support the integration of multiple data sources within a unified display.

IDS Rainstorm, shown in Fig. 8, focuses on scalability, mapping IDS alerts to pixels over time (Abdullah et al., 2005; Conti et al., 2006). Zooming and drilling down to the details allow the users to understand the details of their IDS data. The overview visualization aggregates 20 IP addresses for each row of pixels, organized sequentially from top to bottom and the columns wrap around at the bottom of the display. Each column represent 24 h of alerts. By wrapping the columns, IDS Rainstorm can represent 2.5 class B IP networks (163,830 hosts) in a single display. This type of display, similar to the software visualization tool SeeSoft (Eick et al., 1992), maximizes the available display space to provide an overview of very large data sets. The color of the pixels represent the severity of the associated alerts (the highest severity of the group of 20 is used). A second display screen is used to show a zoomed in view, which shows larger glyphs to represent alerts and also adds semantic details to show connections between the internal IP address space and external IP addresses represented in the alert. Like NVisionIP, this is a noteworthy example of synchronizing multiple views to show different levels of semantic detail.

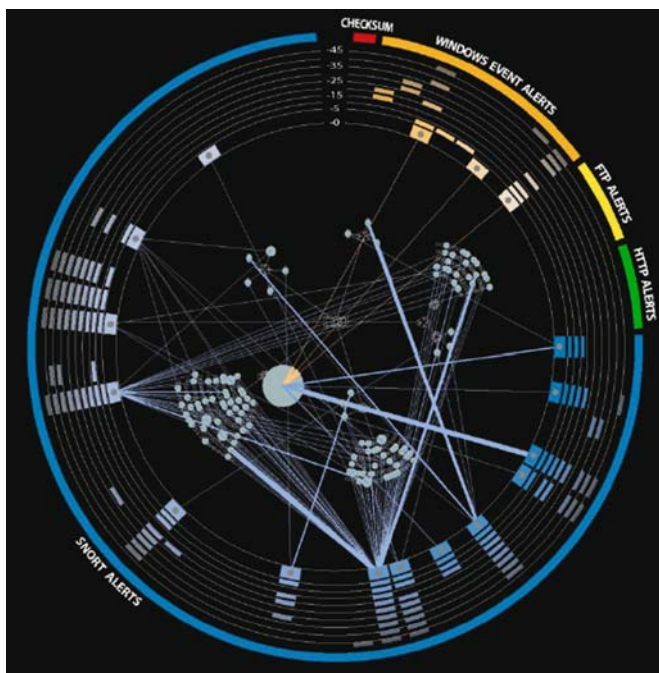


Fig. 7 VisAlert visualization of correlated intrusion detection alerts showing alerts along outer rings and network topology maps in the center. ©2005 IEEE, Inc. Included here by permission

4 Papers in This Volume

The papers collected in this volume were presented at the Fourth VizSec Workshop for Computer Security, held in conjunction with IEEE Vis and InfoVis in Sacramento, California in 2007. This collection presents the state of the art in VizSec research.

4.1 Users and Testing

Anita D’Amico and Kirsten Whitley open this volume with an invited chapter entitled *The Real Work of Computer Network Defense Analysts: The Analysis Roles and Processes that Transform Network Data into Security Situation Awareness*. This chapter is intended to frame the central problems of CND work that security visualization applications attempt to solve. The authors report on the results of their cognitive task analysis of CND analysts in the U.S. Department of Defense. They cover three of the findings from the task analysis: the cognitive transformation process from raw data into security situation awareness, the identification and

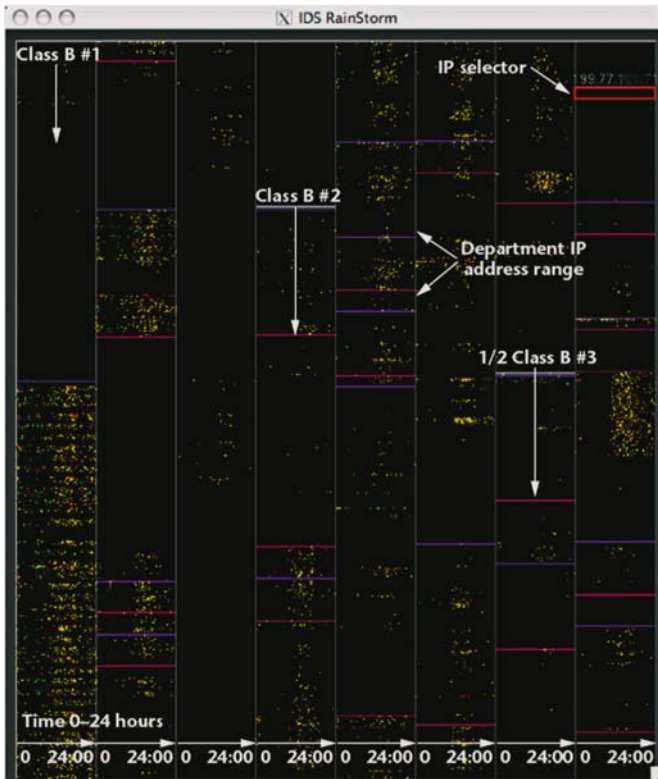


Fig. 8 IDS Rainstorm maps intrusion detection alerts to pixels in the overview visualization that wraps columns of IP address activity over a 24 h time period. ©2006 IEEE, Inc. Included here by permission

description of the analysis roles in CND, and CND analysts' workflow across organizations. The authors conclude by linking their findings to visualization design; drawing valuable implications for future VizSec researchers and developers.

Jennifer Stoll, David McColgin, Michelle Gregory, Vern Crow, and W. Keith Edwards apply a user-centered design method to VizSec in *Adapting Personas for Use in Security Visualization Design*. The authors turn to human-computer interaction and participatory design research to solve the problem of requirements capture by using personas. Personas are an archetype description of a system's target users that provide a framework for organizing requirements. Rather than approach users for feedback on design, designers can turn to the personas to simulate how well a design meets user requirements. This chapter demonstrates how user-centered design methodologies can be applied to VizSec software development.

Xiaoyuan Suo, Ying Zhu, and G. Scott Owen focus on evaluating VizSec software in *Measuring the Complexity of Computer Security Visualization Designs*. The authors propose an alternative evaluation method to user studies: complexity analysis. VizSec designers developers can use this method to evaluate a set of factors

that affect the ability of users to understand a visualization. Complexity is measured across several dimensions, including visual integration, separable dimensions for each visual unit, the complexity of interpreting the visual attributes, and the efficiency of visual search. The authors demonstrate the complexity analysis method with two VizSec applications, *rumint* and *tnv*, which were described in Sect. 3.2.1.

Tamara H. Yu, Benjamin W. Fuller, John H. Bannick, Lee M. Rossey, and Robert K. Cunningham address the difficulty of supporting network testbed operations in *Integrated Environment Management for Information Operations Testbeds*. Network testbeds are crucial in the design and testing of information operations software, but as testbeds become more realistic, they also become more complex to set up and manage. The authors present a visual interface that facilitates test specification, testbed control, and testbed monitoring through multiple information visualization techniques.

4.2 Network Security

Doantam Phan, John Gerth, Marcia Lee, Andreas Paepcke, and Terry Winograd present a VizSec system called *Isis* in *Visual Analysis of Network Flow Data with Timelines and Event Plots*, which was named the workshop's Best Paper winner. *Isis* supports the analysis of network flow data through two visualization methods, progressive multiples of timelines and event plots, to support the iterative investigation of intrusions. *Isis* combines visual affordances with structured query language (SQL) to minimize user error and maximize flexibility. *Isis* keeps a history of a user's investigation, easily allowing a query to be revisited and a hypothesis to be changed. A detailed case study using anonymized data of a real intrusion demonstrates the features of *Isis*.

Teryl Taylor, Stephen Brooks, and John McHugh present another VizSec system for network flow analysis in *NetBytes Viewer: An Entity-Based NetFlow Visualization Utility for Identifying Intrusive Behavior*. *NetBytes Viewer* plots network flow data per port of an individual host machine or subnet on a network over time in 3D. The Z axis displays the ports, the X axis displays time, and the Y axis displays the magnitude of traffic (in flows, packets, or bytes) seen by the host (or subnet) in an hour.

Denis Lalanne, Enrico Bertini, Patrick Hertzog, and Pedro Bados describe a visualization approach to support multiple user roles in *Visual Analysis of Corporate Network Intelligence: Abstracting and Reasoning on Yesterdays for Acting Today*. The authors present a pyramidal vision of network intelligence to support more than just the daily monitoring of networks. In addition to the system and security analysts, the authors argue that other user profiles are interested in network intelligence, such as the the helpdesk, legal department, and the chief executive officer. They present two methods of network analysis, taking a user/application centric view and alarm/temporal centric view.

Jason Pearlman and Penny Rheingans take a service-oriented perspective to visualizing network traffic in *Visualizing Network Security Events Using Compound Glyphs from a Service-Oriented Perspective*. The authors present a node-link visualization in which each node is represented as a compound glyph that provides an indication of the node's service usage. Time slicing is also used in these glyphs to provide an indication of time.

Barry Irwin and Nicholas Pilkington attempt to map large IP spaces using Hilbert curves in *High Level Internet Scale Traffic Visualization Using Hilbert Curve Mapping*. Network telescope (also called DarkNets) are large collections of IP space with no hosts; all traffic collected on a network telescope is sent to a non-existent host. These dead end communications are never legitimate and provide indications of backscatter, scanning, and worm activity. The authors use Hilbert curves, a space filling curve that preserves locality (i.e., ordered data will remain ordered along the curve), to map the activity on large network telescopes.

4.3 Communication, Characterization, and Context

Stefano Foresti and James Agutter present their experience with the design of a VizSec system in *VisAlert: From Idea to Product*. VisAlert, described above in Sect. 3.2.3, is a VizSec system that can correlate data from multiple sources into a unified visualization. In this invited chapter, the authors describe the design process from the conception of rough visual sketches to the implementation and deployment of a production-ready software and the issues that the design team had to address to carry the project from concept to product.

Dino Schweitzer, Leemon Baird, and William Bahn present a visualization of their security algorithm in *Visually Understanding Jam Resistant Communication*. Their algorithm, BBC, is based on a new type of coding theory known as concurrent codes that is resistant to traditional jamming techniques. The authors found it difficult to explain the formal definition and proofs to non-mathematicians, and so turned to visualization as a communication device to visually demonstrate the algorithm's effectiveness.

Florian Mansman, Lorenz Meier, and Daniel A. Keim present an approach to visualizing host behavior in *Visualization of Host Behavior for Network Security*. The authors use a force-directed graph layout to look at changes in host behavior over time to assist in the detection of uncommon behavior. A node represents the state of one host for a specific interval and its position is determined by its state at that interval. So as hosts' states change, their position also changes, allowing analysts to easily see changes over time.

William A. Pike, Chad Scherrer, and Sean Zabriskie focus on bringing context into visualization in *Putting Security in Context: Visual Correlation of Network Activity with Real-World Information*, which was named the workshop's Best Paper runner-up. The central tenant of the paper is that CNL analysts use their own understanding of the world to put security events into context. In order to support this

necessary analytic step, the authors demonstrate a system, called NUANCE, that creates behavior models for network entities at multiple levels of abstraction and fuses these models with contextual information on current threats and exploits from textual data sources.

4.4 Attack Graphs and Scans

Leevar Williams, Richard Lippmann, and Kyle Ingols present an elegant solution to visualizing attack graphs in *An Interactive Attack Graph Cascade and Reachability Display*. Attack graphs present potential critical paths that could be used by adversaries to compromise networked hosts based on their known vulnerabilities. Attack graphs are useful for understanding the vulnerability level of a network, but are often too complex to understand. The authors present a visual solution for attack graph comprehension based on treemaps. Multiple treemaps are used to cluster host groups in each subnet. Hosts within each treemap are grouped based on reachability, attacker privilege level, and prerequisites.

Chris Muelder, Lei Chen, Russell Thomason, Kwan-Liu Ma, and Tony Bartoletti combine machine learning and visualization to tackle the problem of classifying scanning activity in *Intelligent Classification and Visualization of Network Scans*. The authors present a system that uses associative memory learning techniques to compare network scans in order to create classifications. The classifications can be used with visualization to characterize the source of scans.

Barry Irwin and Jean-Pierre van Riel describe a 3D visualization for traffic analysis in *Using InetVis to Evaluate Snort and Bro Scan Detection on a Network Telescope*. Source IP address, destination IP address, and destination port are mapped to the three axes in InetVis for TCP and UDP traffic and a separate plane is shown below this cube (with no port information) for ICMP traffic. InetVis also incorporates textual filtering and querying using the powerful and flexible the Berkeley Packet Filter syntax. The authors use the visualization to examine the scan detection capabilities two IDSs to identify possible flaws in those scan detection algorithms.

5 Conclusion

VizSec is a growing community that is attempting to solve the important problems of computer security through enabling humans through information visualization. This chapter has highlighted the motivation for VizSec and presented some of the tasks VizSec tools support and the data sources visualized. Examples of visualizations of packet traces, network flows, and intrusion detection alerts were presented to provide an understanding of some of the themes that VizSec research has grappled with and solved, particularly for CND.

References

- Abdullah, K., Lee, C., Conti, G., Copeland, J.A., Stasko, J.: Ids rainstorm: Visualizing ids alarms. In: Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC), pp. 1–10 (2005)
- Ahlberg, C., Shneiderman, B.: Visual information seeking using the filmfinder. In: ACM Conference Companion on Human Factors in Computing Systems (CHI), pp. 433–434. ACM, New York (1994)
- Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., Stoner, E.: State of the practice of intrusion detection technologies. Tech. Rep. CMU/SEI-99-TR-028, Carnegie Mellon University/Software Engineering Institute (1999)
- Card, S.K.: Information visualization. In: Jacko, J.A., Sears, A. (eds.) *The Human Computer Interaction Handbook*, pp. 544–582. Lawrence Erlbaum Associates, Mahwah, NJ (2003)
- Card, S.K., Mackinlay, J.D., Shneiderman, B. (eds.): *Information Visualization: Using Vision to Think*. Morgan Kaufman, San Francisco, CA (1999)
- Conti, G., Grizzard, J., Ahamad, M., Owen, H.: Visual exploration of malicious network objects using semantic zoom, interactive encoding and dynamic queries. In: Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC), pp. 83–90 (2005)
- Conti, G., Abdullah, K., Grizzard, J., Stasko, J., Copeland, J.A., Ahamad, M., Owen, H., Lee, C.: Countering security analyst and network administrator overload through alert and packet visualization. *IEEE Computer Graphics and Applications* **26**(2), 60–70 (2006)
- Eick, S.G., Steffen, J.L., Eric, E., Sumner, J.: Seesoft—a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering* **18**(11), 957–968 (1992)
- Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Preserving the big picture: Visual network traffic analysis with mv. In: Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC), pp. 47–54. IEEE Press, New York (2005)
- Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Focusing on context in network traffic analysis. *IEEE Computer Graphics and Applications* **26**(2), 72–80 (2006)
- Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. *Journal of Computer Security* **6**(3), 151–180 (1998)
- Lakkaraju, K., Yurcik, W., Lee, A.J.: Nvisionip: Netflow visualizations of system state for security situational awareness. In: Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC), pp. 65–72 (2004)
- Lakkaraju, K., Bearavolu, R., Slagell, A., Yurcik, W.: Closing-the-loop: Discovery and search in security visualizations. In: Proceedings of the IEEE Workshop on Information Assurance and Security (IAW), pp. 58–63 (2005)
- Lee, C.P., Copeland, J.A.: Flowtag: A collaborative attack-analysis, reporting, and sharing tool for security researchers. In: Proceedings of the ACM Workshop on Visualization for Computer Security (VizSEC), pp. 103–108. ACM, New York (2006)
- Lee, W., Stolfo, S.J., Mok, K.W.: Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review* **14**(6), 533–567 (2000)
- Livnat, Y., Agutter, J., Moon, S., Erbacher, R.F., Foresti, S.: A visualization paradigm for network intrusion detection. In: Proceedings of the IEEE Workshop on Information Assurance and Security (IAW), pp. 92–99 (2005a)
- Livnat, Y., Agutter, J., Shaun, M., Foresti, S.A.F.S.: Visual correlation for situational awareness. In: Agutter, J. (ed.) *IEEE Symposium on Information Visualization (InfoVis)*, pp. 95–102 (2005b)
- McHugh, J.: Intrusion and intrusion detection. *International Journal of Information Security* **1**(1), 14–35 (2001)
- Shneiderman, B.: Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics* **11**(1), 92–99 (1992)
- Shneiderman, B.: The eyes have it: A task by data type taxonomy of information visualizations. In: Proceedings of the IEEE Symposium on Visual Languages, pp. 336–343 (1996)

- Shneiderman, B.: Treemaps for space-constrained visualization of hierarchies (2006). <http://www.cs.umd.edu/hcil/treemap-history/>
- Stoll, C.: The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage. Pocket Books, New York (1989)

The Real Work of Computer Network Defense Analysts

The Analysis Roles and Processes that Transform Network Data into Security Situation Awareness

A. D'Amico and K. Whitley

Abstract This paper reports on investigations of how computer network defense (CND) analysts conduct their analysis on a day-to-day basis and discusses the implications of these cognitive requirements for designing effective CND visualizations. The supporting data come from a cognitive task analysis (CTA) conducted to baseline the state of the practice in the U.S. Department of Defense CND community. The CTA collected data from CND analysts about their analytic goals, workflow, tasks, types of decisions made, data sources used to make those decisions, cognitive demands, tools used and the biggest challenges that they face. The effort focused on understanding how CND analysts inspect raw data and build their comprehension into a diagnosis or decision, especially in cases requiring data fusion and correlation across multiple data sources. This paper covers three of the findings from the CND CTA: (1) the hierarchy of data created as the analytical process transforms data into security situation awareness; (2) the definition and description of different CND analysis roles; and (3) the workflow that analysts and analytical organizations engage in to produce analytic conclusions.

1 Introduction

As government and business operations increase their reliance on computer networks and the information available on them, defending these valuable networks and information has become a necessary organizational function. Risks have appeared from many sources. The online world is witnessing increasingly sophisticated technical and social attacks from organized criminal operations. Moreover, an estimated

A. D'Amico
Secure Decisions division of Applied Visions, Inc.

K. Whitley
Department of Defense

120 countries are using the Internet for political, military or economic espionage (McAfee, 2007).

The broad area of cyber security encompasses policy and configuration decisions, virus scanning, monitoring strategies, detection and reaction. In the commercial world, the domain of expertise for securing and defending information resources is referred to as information security (InfoSec). U.S. governmental organizations use the synonymous terms computer network defense (CND) and defensive information operations (DIO).

This paper treats the topic of CND analysis from the perspective of the people working as professional CND analysts. We discuss how their user requirements should apply to the design of CND visualization tools. To describe the nature of CND analysis, we draw upon a cognitive task analysis (CTA) that we conducted in the 2004–2005 timeframe using mainly CND analysts working within U.S. Department of Defense (DOD) organizations. D'Amico et al. (2005) provides a preliminary report on that work. The research was designed to gain a full understanding of the daily CND analysis process. Three design considerations were: to understand both the similarities and differences in how network data was analyzed across different organizations; to include analysts whose responsibilities ranged from defending local networks to looking for attacks more broadly across a community (i.e., the notion of enclave, regional and community monitoring); and to include perspectives stemming from both tactical and strategic missions.

The CTA research was undertaken with several goals in mind, including to serve as foundation material for tool developers who do not have easy access to CND analysts and to provide requirements for the design of successful visualization for computer security. These goals also motivate this paper. This paper summarizes three findings from the CTA: (1) the hierarchy of data created as the analytical process transforms data into security situation awareness; (2) the definition and description of different CND analysis roles; and (3) the workflow that analysts and analytical organizations engage in to produce analytic conclusions. We pinpoint cognitive needs of CND analysts, rather than the software and system requirements. The analytic process is a joint (both human and machine) cognitive system, and the pipeline of CND analysis will not be automated in the near future. The needs of human analysts will remain a critical component of successful CND and should be considered when designing CND visualizations.

2 Related Work

The CND mission is succinctly summarized by Sami Saydjari: “Imagine that you lead an organization under computer attack on your critical information systems. What questions are you likely to ask? *Am I under attack; what is its nature and origin? What are the attackers doing; what might they do next? How does it affect my mission? What defenses do I have that will be effective against this attack? What can I do about it; what are my options? How do I choose the best option? How do I prevent such attacks in the future?*” (Saydjari, 2004).

To answer these questions, CND analysts are responsible for tasks such as collecting and filtering computer network traffic, analyzing this traffic for suspicious or unexpected behavior, discovering system misuse and unauthorized system access, reporting to the appropriate parties and working to prevent future attacks. CND analysts consult the output of automated systems that provide them with network data that have been automatically collected and filtered to focus the analyst’s attention on data most likely to contain clues regarding attacks. These automated systems (such as firewalls, border gateways, intrusion detection systems (IDSs), anti-virus systems and system administration tools) produce log files and metadata that the analyst can inspect to detect suspicious activities.

To gauge the missions and analytic tasks across the CND community, Carnegie Mellon University (Killcrece et al., 2003) conducted a study of 29 Computer Security Incident Response Teams (CSIRTs), of which 29% were military, and listed the major activities of the teams surveyed. A summary appears in Table 1 along with the percentage of organizations reporting these activities. The bold typeface highlights those activities that were of interest to our CND CTA research. Our research focused on understanding how CND analysts inspect raw data and build their comprehension into a diagnosis or decision, especially in cases requiring data fusion and correlation across multiple data sources. The CTA did not include the work of vulnerability assessments, penetration testing, insider threat or malware analysis.

The CMU study also categorized CND activities or functions into three groups: reactive, proactive, and security quality management. Reactive activities are triggered by a preceding event or request such as a report of wide-spreading malicious

Table 1 The major activities performed by CND analysts and percentages of organizations reporting these activities (Killcrece et al., 2003)

Activities of Computer Security Incident Response Teams	
Incident handling	97%
Perform security policy development	72%
Publish advisories or alerts	72%
Perform artifact analysis	66%
Perform virus handling	66%
Monitor IDS	62%
Produce technical documents	62%
Provide and answer a hotline	62%
Do training and security awareness	59%
Perform forensic evidence collection	55%
Perform a technology watch or monitoring service	55%
Track and trace intruders	52%
Pursue legal investigations	44%
Vulnerability handling	41%
Monitoring network and system logs	38%
Security product development	34%
Vulnerability scanning	31%
Vulnerability assessments	28%
Security configuration administration	24%
Penetration testing	17%

code or an alert identified by an IDS or network logging system. Looking to the past, reactive tasks include reviewing log files, correlating alerts in search of patterns, forensic investigation following an attack and identification of an attacker who has already penetrated the network. Looking to the future, proactive activities are undertaken in anticipation of attacks or events that have not yet manifested. Proactive tasks include identifying new exploits before they have been used against the defended network, predicting future hostile actions and tuning sensors to adjust for predicted attacks. Security quality management activities are information technology (IT) services that support information security but that are not directly related to a specific security event; these include security training, product evaluation, and disaster recovery planning. Killcrece et al. reported, and our CTA results support, the fact that most CND analysis work is reactive, not proactive. In the CND CTA, we looked for examples of proactive work; however, the majority of the analytic activity was reactive. In describing the analysis roles below, we note instances in which proactive tasks can occur.

Alberts et al. (2004) extended the work of Killcrece et al. in a report that advocates best-practice workflows for effective incident management. Their models represent what incident response should or could be and do not necessarily represent the actual experiences of most CND analysts. By comparison, our CTA studied the state of the practice, sought to understand the existing factors that impede successful analysis and identified opportunities to improve situation awareness. Biros and Eppich (2001) conducted a CTA of rapid intrusion detection analysts (which include triage and aspects of escalation analysis, as defined below) in the U.S. Air Force and identified four requisite cognitive abilities: recognizing non-local Internet Protocol (IP) addresses, identifying source IP addresses, developing a mental model of normal, and sharing knowledge. We used their work as a starting point, but studied the larger range of CND analysis beyond triage analysis and beyond the Air Force.

3 Methods

Generally, CTA is the study of an individual's or team's cognitive processes, activities and communications within a specific work context. CTA uses naturalistic observation techniques to elucidate expertise and to understand the actual effect of processes and systems (e.g., software systems) built to automate or assist human decision makers. Ideally, a CTA involves both observing individuals as they go about their work and asking directed questions about the way in which they approach the problems, how they decide what steps to take, their communications with their co-workers and the difficulties of their work. In a CTA, care is taken to distinguish between the inherent work of a domain and the work that may be created by the current working environment and tools; in this way, the CTA can provide insight into how the current working environment helps or hinders the ultimate goals of the work. The output of a CTA is a detailed description of the tasks that an individual or team performs, the data on which they operate, the decisions they make and the

processes and activities (cognitive, communicative and perhaps physical) that they engage in to reach those decisions.

During the CTA described in this paper, 41 CND professionals working in seven different organizations participated. Most were currently active analysts; a few were managers who were not performing analysis on a day-to-day basis. They varied in level of expertise and represented a variety of job titles and work roles, as defined by their organizations. We focused on CND analysts who look at network traffic and related data to determine whether the information assets are under attack and who the attacker is. To collect data, we used a combination of four knowledge capture techniques: semi-structured interviews, observations, review of critical incidents and hypothetical scenario construction. In semi-structured interviews, the researcher guided discussion with an analyst by using a checklist of questions, yet also used wide latitude to encourage the subject to describe the day-to-day work in detail. Observations involved watching analysts at work combined with asking questions to clarify the process. Review of critical incidents involved dissecting past incidents that challenged the analyst's skills. The technique of scenario construction involved working with analysts to flesh out an imaginary analysis case of typical offensive actions taken by a sophisticated attacker and defensive actions taken by the CND analyst. Scenario construction allowed analysts to reveal the kinds of information they seek from available data sources, knowledge of adversary operations and techniques, and types of connections they make between seemingly disparate pieces of information.

4 Findings

While the organizations participating in the CTA differed in their stated mission (such as protecting a single network, identifying trends in computer attacks across the entire DOD, or performing CND services for customers outside one's own organization), they had much in common. This overlap, however, was obscured by the lack of standard terminology. Whereas various members of the community used common terms (e.g., *event* and *alert*), they often used the terms differently or without a specific definition. Also, at times, the analysts used different terms that gave an initial impression that they had different missions or needs. Therefore, a primary task in the CTA data analysis became to analyze the participants' usage of terms in the context of the details of their work. By sifting through the details, we identified similarities across the community. For tool designers, these widespread cognitive processes are valuable to understand, because they are fundamental aspects of CND.

The following sections describe CND analysis from three perspectives, highlighting aspects of human cognition. The first section considers the status of the data as raw data are processed into analytic product. The second section presents a range of analysis roles, based on work performed, not on job title. The third section describes a synthesized workflow that captures the steps in the analytic process. Throughout the discussion, the focus is on the similarities across organizations, not

on exceptional cases. In these findings, we use a standardized vocabulary, not in a prescriptive way, but as a way to illuminate the CND process data for the purpose of the CTA.

4.1 Data Transformation in CND Analysis

As a CND analyst works, data are filtered, sorted, retained or discarded based on the analyst’s responsibilities and expertise. Different responsibilities involve different data. For example, some analysts primarily review the newest packet traffic or sensor data; others concentrate on data that have already been identified as suspicious, but require further analysis and correlation with additional data sources. As we discussed this process with the analysts, we ascertained that, as analysis proceeds, data are transformed as an analyst’s confidence in an analytic conclusion increases. The cognitive transformation can be seen as a hierarchy of data filters with the levels reflecting the increasing certainty that a reportable security violation has been identified (depicted in Fig. 1). It is worthwhile to note that the volume of data generally decreases from level to level.

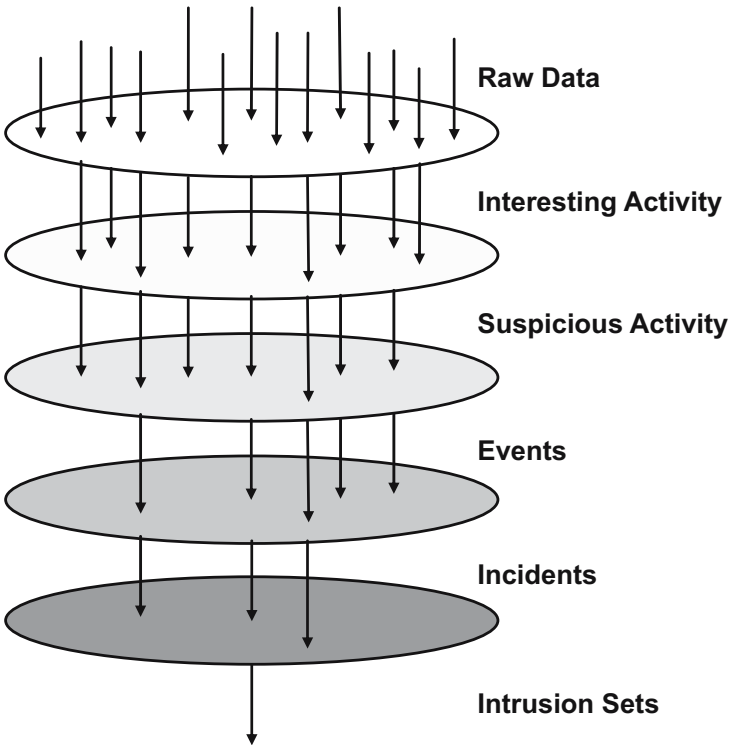


Fig. 1 Data hierarchy as data are transformed into security situation awareness

Raw data are the most elemental data in the hierarchy. At the start of the entire CND analytic workflow, the raw data can be network packet traffic, netflow data or host-based log data. Especially because the amount of raw data is so large, analysts do not generally inspect all raw data. Instead, raw data are passed through an automated process (e.g., an IDS) that makes initial filtering decisions (e.g., based on attack signatures). The automated filtering results in a substantially reduced amount of data requiring human attention.

Interesting activity refers to the data that has been flagged by the initial automated filter and sent to a CND analyst for inspection. We heard it referred to as *activity*, *alerts*, *alarms*, *data*, *logs* and *interesting activity*. Some analysts objected to the term *alerts* because they felt strongly that activity is not an alert until a human analyst has inspected and verified that the activity is worthy of further attention. Interesting activity might be presented to the CND analyst in the form of packet header data from TCPDUMP or as an IDS alert. Depending on the techniques employed by the automated filter, the interesting activity may be largely composed of false positives. Analysts perform triage on interesting activity, examining the alert details and related data, throwing out false positives and retaining the remainder for closer inspection.

Suspicious activity remains after the triage process because the CND analyst believes that the activity is anomalous for the monitored network or because it adheres to a signature or attack pattern associated with malicious intent. Some CTA participants called this type of activity an *event*, *anomaly* or *suspicious activity*. Examples of suspicious activity include a series of scans from the same source IP address; an unusual increase in traffic to or from a server inside the network; virus infections on several workstations in a short time period; and misuse of the monitored network by employees downloading inappropriate content.

Event refers to suspicious activity that a CND analyst has a responsibility to report, based on the organization's mission and policies. For example, an organization might be charged to report only on certain types of intrusion attempts and not on employee policy violations (e.g., using unauthorized peer-to-peer software); in this case, a policy violation would not be escalated as an event.

At the level of events, the volume of data has been significantly reduced from that of raw data. It is also the point at which CND analysts begin grouping individual activity based on common characteristics (such as source and destination IP addresses, time, attack characteristics or attacker behavior). Along the analysis workflow, CND analysts are also expanding their understanding of the data by searching for and adding new facts that show the extent of the security violation including the actors, machines, and information that has been compromised. The work for the CND analyst inspecting event data is to confirm that a security violation has occurred and to provide as full an understanding as possible of the violation.

Incident is the point when a CND analyst(s) has confirmed the occurrence and seriousness of one or more events and reports on the collection of relevant data. The incident level is usually a formal, documented point in the analysis process. A CND

analyst prepares a formal report describing the incident. After any required approval, the incident report is released as an official analytic product. Some organizations have more than one type of reports (e.g., a rapid-release distribution mechanism to distribute early information as quickly as possible and a formal reporting mechanism which is the finalized incident description). Within the DOD, official incidents are assigned to the responsible party for incident handling. Incidents may be tagged with a category type or priority ranking. Currently, there is no international consensus on incident categories or how to measure incident severity.

Incident reports are distributed to interested parties based on factors like category type and official reporting chain. The topic of report distribution and data sharing is closely related to the fact that CND analysis is often done collaboratively across organizations. Monitoring often takes place at enclave, regional and community levels with formal or informal collaboration and sharing across levels. For example, in the DOD, CND analysis occurs at individual military bases (i.e., enclave level), at the military service level (i.e., regional level) and across the entire DOD (i.e., community level). Data and reports flow up and down this reporting chain. Currently, the Joint Task Force for Global Network Operations (JTF-GNO) provides DOD community-wide analysis. For state and local governments, US-CERT, operated by the Department of Homeland Security (DHS), provides the community level. In the commercial world, companies have corporate monitoring (i.e., enclave or regional level) and may also report to a community service (e.g., a financial institution may participate in the Financial Services Information Sharing and Analysis Center (FS-ISAC)). In the case of Managed Security Service Providers (MSSPs), incidents are reported to individual customers (i.e., enclave level); an MSSP might also perform trend analysis across its entire customer base.

The benefit of wider analysis at the community level is indisputable. Aside from individual enclave concerns about the sensitivity of their data, the value of grouping CND data stems from the fact that certain incidents cannot be fully understood within a single enclave. When protecting national interests, it is important to detect related activity and larger trends occurring across individual enclaves.

Intrusion sets are sets of related incidents. In the organizations we visited, *intrusion sets* and *problem sets* were essentially synonymous terms. Intrusion sets commonly arise at the community level when CND analysts can review incidents from different reporting organizations and group these incidents based upon shared features such as source and destination IP addresses, time, attack characteristics or attacker behavior. When a CND community suspects that separately reported incidents emanate from the same source or sponsor, the community groups the incidents into an intrusion set. Just as incidents are almost universally a formal analytic product, the designation of an intrusion set is an official decision point for the organizations in our CND CTA. The community then increases attention and resources to detecting, understanding and responding to relevant activity. This process can include decisions about tuning data collection and IDS signatures to catch all new related data.

4.2 *CND Analysis Roles*

We wanted to understand whether and how analysis duties are divided across analysts and organizations. However, we were initially confronted with the lack of any functional descriptions of jobs performed by the analysts. Job titles, such as *level 1 analyst* or *lead analyst*, varied considerably across organizations. Furthermore, an analyst with a single job title, such as *lead analyst*, often performed many roles, such as rapid intrusion detection, consultation with other analysts and even training of junior analysts. In considering how to address this lack of common descriptions, we decided to categorize analytical function based on the actual tasks performed. The result of this exercise was a set of six broad analysis roles that accounted for all of the cognitive work observed: triage analysis, escalation analysis, correlation analysis, threat analysis, incident response and forensic analysis.

These roles represent categories of analysis; the roles do not directly map to job titles. An analyst with a single job title may perform work across more than one of the analysis roles. The roles illuminate the amount and types of data that the analyst is integrating and the goal of the analysis. The roles also reflect authority boundaries imposed by law and policy (e.g., relating to privacy). Some of the roles align closely with reactive analysis; some include aspects of proactive analysis.

Triage analysis is the first look at the raw data and interesting activity. The triage decision is a relatively fast decision about whether the data warrants further analysis. Triage encompasses weeding out false positives and escalating interesting activity for further analysis, all within a few minutes of viewing the data. Commonly, an analyst inspects IDS alerts and the immediate associated traffic/flow metadata and/or packet contents.

The majority of analysts in the CND CTA performed triage analysis. It is also very common that novice CND analysts are first assigned the job of triage analysis and work under the guidance of more senior analysts. The triage cases that novices encounter provide on-the-job training that increases the range of security violations that they can easily recognize.

Triage analysis is reactive in nature, since it is based on reviewing and sorting activity that has already occurred. Within the CTA, we encountered the following relevant CND job titles: *level 1 analyst*, *first responder* and *real-time analyst*. For the organizations in the CTA, analysts with these job titles spent the majority of their time performing triage analysis. In a small organization or at a remote site within a large organization (e.g., Air Force base), triage analysis may be performed, albeit in a limited way, by the system administrator or network manager.

Escalation analysis refers to the steps taken to investigate suspicious activity received from triage analysis. Escalation analysis requires increasing situation awareness of the suspicious activity. The process may take hours or even weeks from start to finish, during which the CND analyst marshals more data, usually from multiple data sources and from inside and outside the organization, resulting in greater comprehension of the attack methods, targets, goal and severity. The CND analyst may also make an initial assessment of attacker identity and the mission impact of an attack.

A main goal of escalation analysis is to produce incident reports. Compared to triage analysis, escalation looks at related data over longer periods of time (e.g., over the last several months of collected data) and from multiple data sources (e.g., including information from threat reports). The time needed to process these data queries and to interpret and assemble the results accounts for the fact that escalation analysis takes longer than triage analysis. In triage analysis, emphasis is on speed; correspondingly, the analysis usually involves limited queries on a single data source. In the current practice of CND analysis, the combination of triage and escalation analysis is what is often referred to as a real-time monitoring capability (although it does not actually occur in real time).

Sometimes, escalation analysis is based on tip-offs received from colleagues in other analysis groups and from cooperating organizations. This situation occurs particularly for senior analysts who have good contacts throughout the CND community.

Escalation analysis is largely reactive. Less commonly, escalation analysis involves proactive actions such as tuning sensors to look for predicted attacks or activity related to a current investigation. Within the CTA, we encountered the following relevant CND job titles: *level 2 analyst* and *lead analyst*.

Correlation analysis is the search for patterns and trends in current and historical data. At the community level, correlation analysis includes grouping data into intrusion sets; these investigations can take days to months. When conducted at the community level, correlation analysis is closely related to threat analysis.

Correlation tasks include retrospectively reviewing packet data, alert data or incident reports collected over weeks or months of CND monitoring, looking for unexplained patterns. Patterns may arise from different data attributes such as specific source or destination IP addresses, ports used, hostnames, timing characteristics, attack details and attacker behavior. By discovering patterns, CND analysts can uncover suspicious activity that was previously unnoticed. An analyst might not know what patterns they are looking for in advance; instead, the analyst might “know it when they see it.” When they encounter a pattern that they cannot explain, they form hypotheses about potential malicious intent, which they try to confirm or contradict via additional investigation.

In the CND CTA, we encountered few analysts whose primary role was correlation analysis. Only 5% of the CTA participants were primarily responsible for community-wide correlation; another 5% were primarily responsible for the *post hoc* review, at the regional level, to search for anomalies or patterns not found during triage and escalation analysis.

Correlation analysis is reactive when it focuses on discovery within existing data. It has the potential to be proactive if discovered patterns are used to make predictions about next likely actions. Within the CTA, we encountered the following relevant CND job titles: *level 2 analyst*, *correlation analyst* and *site-specific analyst*. We choose the term correlation analysis, not out of a technically correct use of the concept of correlation, but rather due to the prevalent use of the term in the CND community to refer to grouping related data.

Threat analysis is intelligence analysis in support of CND. Threat analysis uses data sources beyond the monitored traffic (e.g., information published on hacker websites) to gain additional insight into the identity, motives and sponsorship of attackers and to forecast upcoming CND attacks. The additional data sources provide a higher-level perspective than is possible by examining computer network traffic and host-based activity. The additional data sources are essential for understanding an attacker's true identity and intent; One of CTA participants explained, "Intelligence is the most important factor in doing prediction and attribution."

Threat analysis may proceed in reaction to a specific attack. However, threat analysis is the most proactive of the analysis categories, since threat analysis can precede and uncover facts before a CND attack occurs. In the CTA, we found that threat analysis was primarily linked with the job title of *threat analyst*.

Incident response analysis recommends and/or implements a course of action in reaction to a confirmed incident. Responses may be as straightforward as blocking a source IP address, or as complex as "caging" or "fish bowling" an attacker inside the network to observe the attacker in action. Incident response involves assessing the tradeoffs of potential responses and how the responses will impact organizational mission. Incident response analysis is, by definition, a reactive activity. Within the CTA, we encountered the following relevant CND job titles: *incident handler* and *incident responder*.

Incident response, as well as forensic analysis, involves the issue of authority. Because of legal ramifications, only certain CND analysts are authorized to implement a response. In the CTA, the majority of analysts was responsible for analysis and reporting and not authorized to take response actions.

Forensic analysis consists of gathering evidence in support of a law enforcement investigation instigated by an incident. Forensic analysis is especially concerned with evidence preservation and has increased need for host-based evidence collection. Forensic analysis takes weeks to months to complete. Forensic analysis is, by definition, a reactive function.

In the DOD, forensic analysis is separated from the other aspects of CND analysis due to the issue of authority. Only certain analysts are authorized to collect and review cases involving U.S. citizens and to prepare this data for legal action. Forensic analysis is performed by members of a law enforcement organization, who may be assisted by incident handlers.

4.3 CND Analysis Workflow Across Organizations

Each organization that participated in the CTA had its own workflow for analyzing CND data, which differed from the other organizations' workflows. Nonetheless, after capturing each process in a workflow diagram and comparing them, we found many commonalities. Figures 2 and 3 depict a synthesized workflow that encapsulates and abstracts observations from all seven organizations.

The entire workflow operates to transform data from interesting activity to incidents and intrusion sets, with the goal of enhancing the situation awareness of

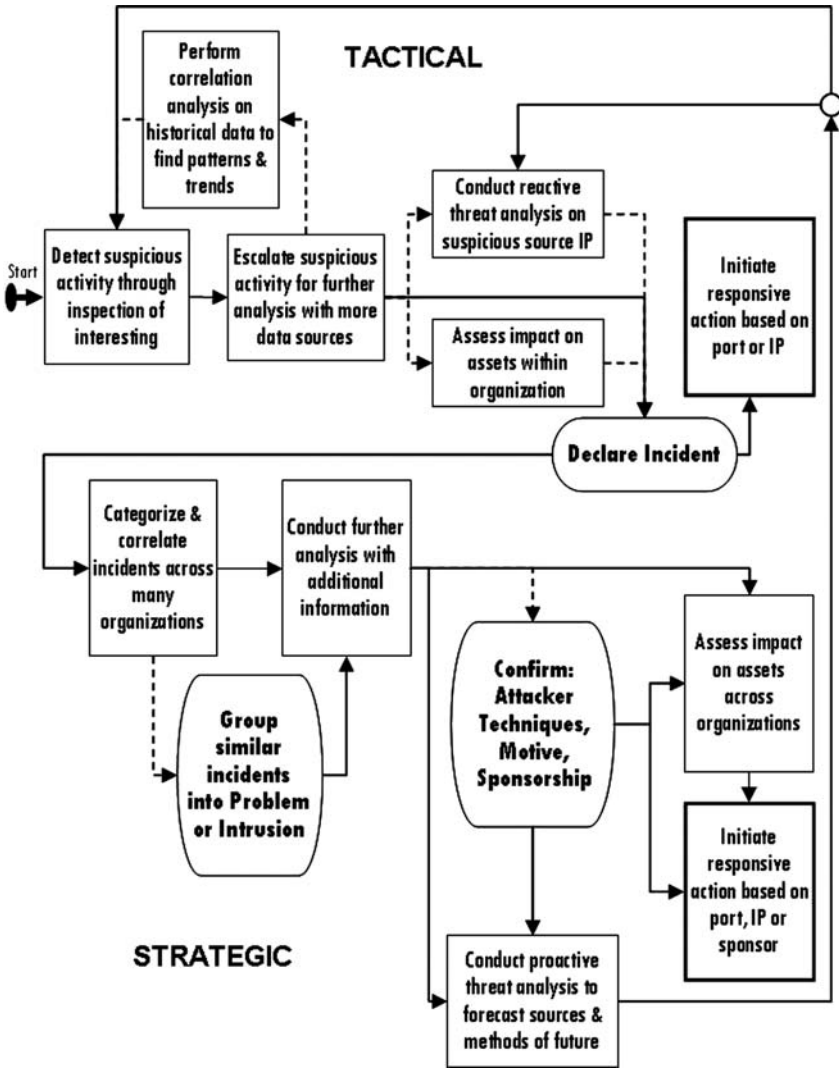


Fig. 2 Generalized CND workflow

individual analysts, organizations and the community of related organizations. All of the analysis roles play a part in this workflow. Some roles operate sequentially (e.g., triage precedes escalation) while others recur through the workflow (e.g., correlation and threat analysis). Feedback loops in the workflow illustrate the interdependence of the analysis roles and the importance of analyst communication in the data transformation process. The workflow contains both tactical and strategic goals (highlighted in Fig. 2 and described below). The workflow also contains three stages that align with principles of situation awareness and sensor data fusion (highlighted in Fig. 3 and described below).

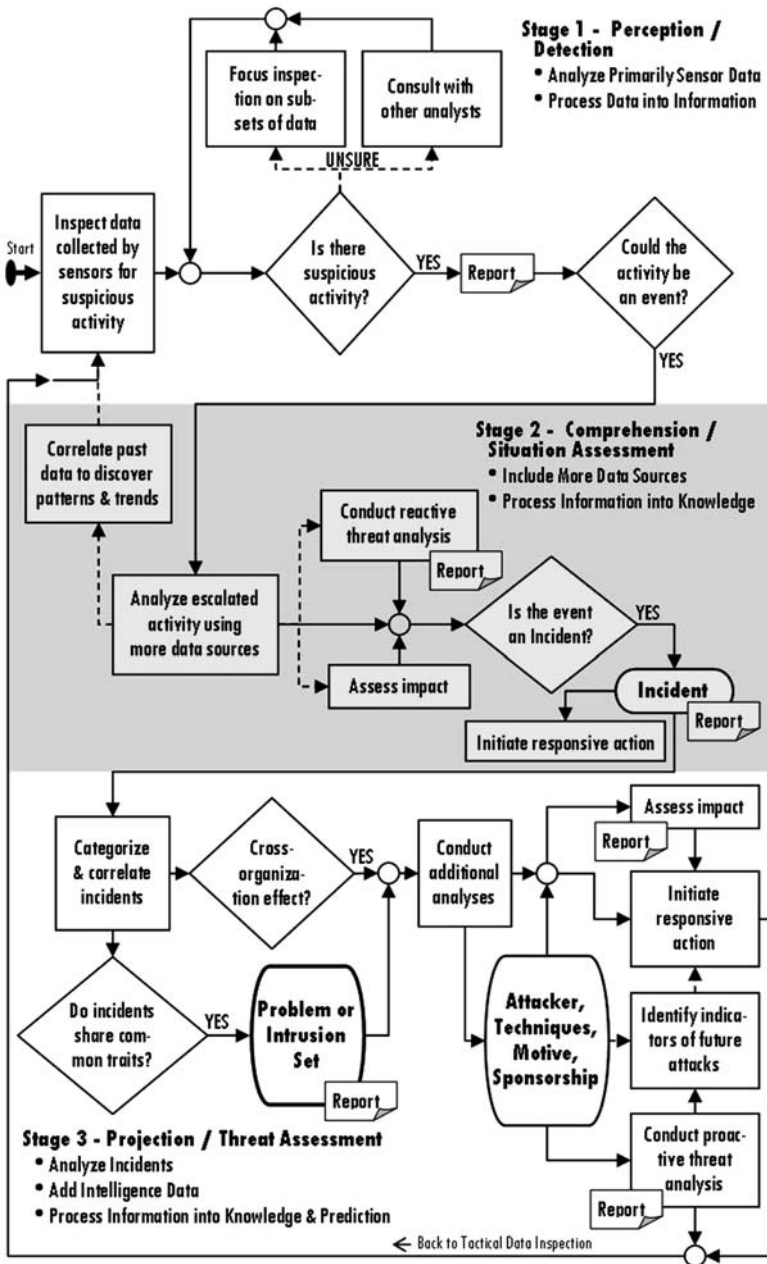


Fig. 3 Stages of CND situation awareness and cognitive data fusion

Tactical analysis. The top half of Fig. 2, ending at the declaration of an incident and initiation of incident response, represents a tactical focus. The goal of tactical analysis is to defend against an immediate, current attack and to maintain the operational status of the monitored networks. Tactical analysis is usually performed at enclave and regional levels. Analysis at the tactical level focuses on inspecting IDS alerts, flow data, firewall logs and TCPDUMP, with additional information drawn from open source and intelligence sources. The declaration of an incident is a definitive point in tactical analysis that initiates a series of incident response tasks. Incident response may be handled by the monitoring organization or by the enclave network administrator.

Strategic analysis. The bottom half of Fig. 2, beginning with the categorization and correlation of confirmed incidents, represents a strategic focus. The goal of strategic analysis is to understand the broader implications of related attacks. Strategic analysis is an important function of the community level (e.g., JTF-GNO and FS-ISAC). Incidents declared across enclaves and regions are collected at the community level. At this strategic level, a community organization examines all constituents' incidents for patterns or trends. Such patterns may indicate a well-resourced, sophisticated attacker with motives beyond nuisance attacks on an individual workstation.

Analysis at the strategic level focuses on confirmed incidents; it may be augmented with data from community sensors, intelligence reports and packet-related data requested from an enclave location. The recognition of an intrusion set is a definitive point in strategic analysis that triggers additional analyses into attack attribution, techniques, motive and sponsorship.

Stages of situation awareness. The CND workflow (see Fig. 3) moves through the three stages of building situation awareness: perception, comprehension and projection (Endsley, 1995; Endsley et al., 2003). In CND, the three stages of situation awareness also align with levels of data fusion. Specifically, analysts engage in detection, situation assessment and threat assessment, which are levels of data fusion identified by the Joint Directors of Laboratories (JDL) and recognized widely by the sensor data fusion community (Llinas and Hall, 1998, Waltz, 1998).

Stage 1: perception/detection. During the first stage, a CND analyst acquires data about the monitored environment, which is typical of the perceptual stage of situation awareness. As the analyst performs triage on interesting activity, comprehension begins. An analyst assembles and integrates data to form a mental model of how the interesting activity might represent an attacker's action. By testing hypotheses through additional data and input from colleagues, an analyst modifies and clarifies his mental model. By the end of the first stage, when the analyst decides whether to escalate, the focus shifts from perception to comprehension. The first CND stage is primarily concerned with initial data inspection and detection and, thus, aligns with JDL Level 1.

Stage 2: comprehension/situation assessment. During the second stage, analysts focus on escalation and correlation analysis, which represent the comprehension aspect of situation awareness. Analysts combine suspicious data, their own knowledge and expertise, and additional data sources to determine whether the suspicious

activity represents an incident. The analyst refines the mental model of the attacker's identity and threat level by tracking the attack path through the network and time. Analysts performing correlation analysis identify patterns of anomalous behavior, which they share with those performing triage and escalation analysis.

Stage 2 also involves limited projection. Escalation analysis includes some postulating about an attacker's actions if left unblocked. Incident responders, in choosing a course of action, project what actions an attacker might take if he realizes he has been discovered. This stage also aligns with JDL Level 2 because a main activity is to inject more data (perhaps from relevant, additional data sources) to refine the analysis.

Stage 3: projection/threat assessment. During the third stage, analysts performing correlation, incident response, and threat analysis review and categorize confirmed incidents at the community level. By comparing incidents and adding data from intelligence sources, they discern attack patterns. By this point, the analysts at the community level have a shared mental model. As comprehension improves, the analysts refine the shared mental model and project into the future to forecast the types of incidents to expect within the community. Furthermore, proactive threat analysis identifies potentially new exploits and attackers that could become active in the future. Based on these projections, tips are fed back into the start of the analysis pipeline as perceptual cues for detection. This feedback loop forms a perception-action cycle across multiple, distributed actors. This stage aligns with JDL Level 3 because the analysis involves inferences about attacker identity, motive and sponsorship.

5 Implications for Visualization

5.1 Visualization Across the CND Workflow

Along the CND workflow, as analysis tasks transform data into situation awareness, analysts' attention widens from single packets and network flows to communication networks and final conclusions. Fundamental tasks include noticing what data should be investigated, formulating good analysis questions, searching data sources, evidence tracking and synthesizing a conclusion. CND visualization can occur within a single packet/flow, across multiple packets/flows, across incident reports and in situations that require blending data from multiple data sources.

In the triage stage, CND analysts are commonly looking at IDS alerts and the metadata and content contained in individual packets and flows. Usually, an analyst's attention has been guided to a specific packet or flow by an automated filter or colleague's tip-off. The analyst needs fast access to the alerts and rapid drill down to the related raw data. Viewing individual data elements, an analyst is not yet forming connections to other pieces of data. Simple visualizations are appropriate at this stage. An example is color highlighting of the interesting portions of

the data (e.g., the content that matched the IDS signature) to ensure that the analyst sees relevant details. One simple, but effective visual cue observed in the CTA was an alert management system that used color to reflect the status of the alert. Lines were colored according to whether the alert was not yet assigned to an analyst, whether the alert had been assigned and was in triage or whether the triage had completed (either by discarding or escalating the alert). This mechanism can apply to CND offices with centralized, shared raw data in which the office needs to track all incoming interesting data.

In the escalation and correlation stages, CND analysts build patterns that span across packets. The work includes analysis needed to understand whether an attempted attack succeeded and to understand the extent of a successful attack. A common exercise during escalation is to build the communication network of the suspected attacker and victim. The analyst wants the picture of all known hosts that have communicated with either the attacker or the victim, and all alerts associated with any of these hosts. This network is a central part of the analyst's mental model of the attack; it indicates the extent of the attack by showing likely attack origination points, data exfiltration paths and further contamination. Unifying visualizations can facilitate comprehension of the sequence of interconnected events; appropriate visualizations include graphs (i.e., link-node diagrams) and flexible timeline-based visualizations.

Other patterns in the escalation and correlation stages include summaries of, for example, the most active ports and IP addresses, frequency of alert types, the size of data payloads and the relationships between ports and protocols. Visual presentations can be very useful for exposing patterns and supporting data exploration. Even traditional visual techniques such as scatterplots and histograms, combined with easy-to-use filtering capabilities, can be applied effectively to the problem. Analysts will look for something to "pop out at them." In fact, the popping out that occurs is a cognitive event in which an analyst associates several pieces of data and adds a hypothesis explaining how these data are related.

For correlation for intrusion sets and threat analysis, there is increasing need to combine facts from multiple data sources and to present the data as a cohesive analytic conclusion. For intrusion sets, in particular, a visualization solution should be capable of providing a temporal context that emphasizes the sequence of events. In these roles, there is more likelihood that an analyst will be manipulating secondary data sources and textual data (e.g., incident reports in correlation analysis; open source text in threat analysis) and assembling all evidence into a single compendium. The combination of visualization and annotation (a feature mentioned again in Sect. 5.2) would allow analysts to provide the combination of facts and interpretation that completes the analytic product.

For successful CND visualization, the value of visualization is closely tied to the utility of the data available for viewing and the analysts' ability to search that data. CND analysis is very much an information retrieval problem. Like web searching, CND analysts need to search data repositories and are limited or empowered by the expressivity and usability of the search capability. For example, if analysts can only query their data repository on attributes in the metadata (as opposed

to characteristics of the traffic content), analysts will have difficulty discovering new attack strategies (i.e., for which no IDS signatures exist). Another example concerns whether analysts can easily query the organization's database of incident reports. Incident reports represent the CND organization's collective memory. Often, incident reports are textual documents without delineated fields to enable straightforward searching. The result can be that an analyst is prevented from checking to see if an unusual attack detail has appeared in any prior incident reports.

Expanding on the example of incident reports, the value of CND visualization is strengthened when contextual data are immediately available for viewing. Contextual data are general (e.g., IP registration information) or site-specific (e.g., "Hot IP" lists, prior incident reports) information that speeds situation awareness. In the case of "Hot IP" lists, CND organizations frequently maintain a changing list of IP addresses of particular interest. An analyst may also have a list related to his individual set of investigations. The analyst's perception and comprehension will be improved if data visualizations automatically highlight IP addresses that appear on the organization's list or his individual list.

As a final suggestion, tool designers should consider user groups beyond professional CND analysts. As a whole, the entire Internet population is increasingly concerned with computer security and identity theft. In particular, tool designers could benefit home users by making the current and past behavior of their home computers more transparent and understandable.

5.2 Visualization as Part of a CND Analysis Environment

Visualization components form a part of a larger CND working environment. The success of the visualization pieces depends upon how well the visual tools are integrated with a CND analyst's other tools and how well the combination of tools addresses the CND workflow. These considerations may not be at the forefront of priorities when a visualization designer begins. However, these considerations ultimately determine whether CND analysts are willing to adopt the tool and whether the tool is commercially viable.

Three important criteria are ease of input and output from the tool, support for report building, and management of evidence and analysis. In today's state of the practice, CND analysts spend considerable time on tasks other than analysis. Examples are time spent massaging data formats and spent creating a final report form. Therefore, for a visualization tool to be attractive to analysts, a tool should minimize the effort needed to get data into and out of the tool. A desirable approach is for the tool to support a variety of input and output options, a strategy that also helps interoperability with other tools. For input, it is common for a CND analyst to use a primary data source, but at times to receive data from other sources with differing formats. Similarly, analysts must communicate their analysis to peers; this may mean sharing the data isolated during analysis (i.e., a subset of the actual data) or

sending a copy of the visualization illustrating a communication network or pattern. Thus, the tool should support the output of both data and visualizations.

An extension of this, CND analysts become very enthusiastic at the idea of a tool that automates some or all of the requisite report building. A tool that can directly add relevant data and views into a report for direct distribution to customers will spare analysts the tedious time and potential errors involved in assembling a report using cut-and-paste and transcription. Analysts are not enthusiastic at the idea about investing energy to create a useful visualization, but then having to compose a textual equivalent for their report.

Finally, the category of evidence and analysis management refers to features that assist analysts in gathering and tracking the data that support their current investigations and in documenting the analysts' hypotheses and interpretations. During the CTA, CND analysts were very desirous of tools that would support "an analytic diary" capability. As an analyst isolates data segments relevant to an investigation, it is helpful to be able to save the data subset and its context (i.e., collection information including data source and time range) in a personal analysis space. As a collection of current evidence, the analysis space should support quick access and manipulation of the evidence. Moreover, the analysis space should allow the analyst to annotate investigations with notes about their interpretations of the evidence and recommended next steps.

Acknowledgements The research described in this document was sponsored by the US Department of Defense and the Advanced Research and Development Activity (ARDA) under contract # F30602-03-C-0260, with the Air Force Research Laboratory (AFRL) in Rome, NY as the contracting agency. Mr. Walter Tirenin of AFRL provided opportunities for collaboration with other scientists, careful review of progress, and assistance in making our results available for public distribution. We would also like to thank Daniel Tesone and Brianne O'Brien for their significant work in the CTA data collection and analysis. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARDA or the US Government.

References

- Alberts C, Dorofee A, Killcrece G et al. (2004) Defining Incident Management Processes for CSIRTS: A Work in Progress, Technical Report CMU/SEI-2004-TR-015, ESC-TR-2004-015. <http://www.sei.cmu.edu/publications/documents/04.reports/04tr015.html>. Accessed 06 December 2007
- Biros D, Eppich T (2001) Human Element Key to Intrusion Detection. Signal, August: 31
- D'Amico A, Whitley K, Tesone D et al. (2005) Achieving cyber defense situational awareness: A cognitive task analysis of information assurance analysts. Proceedings of the Human Factors and Engineering Society Annual Meeting: 229–233
- Endsley M (1995) Toward a theory of situation awareness in dynamic systems. Human Factors, 37(1): 32–64
- Endsley M, Bolte B, Jones D (2003) Designing for Situation Awareness: An Approach to User-Centered Design. Taylor & Francis, New York: 13–18

- Killcrece G, Kossakowski KP, Ruefle R, Zajicek M (2003) State of the Practice of Computer Security Incident Response Teams (CSIRTS), Technical Report CMU/SEI-2003-TR-001, ESC-TR-2003-001
- Llinas J, Hall D (1998) An introduction to multi-sensor data fusion. IEEE Report 0-7803-4455-3/98
- McAfee Virtual Criminology Report—Cybercrime: The Next Wave (2007). <http://www.mcafee.com>. Accessed 04 December 2007
- Saydjari O (2004) Cyber Defense: Art to Science, Communications of the ACM 47(3): 53–57
- Waltz E (1998) Information understanding: integrating data fusion and data mining processes. IEEE Report 0-7803-4455-3/98

Adapting Personas for Use in Security Visualization Design

J. Stoll, D. McColgin, M. Gregory, V. Crow, and W.K. Edwards

Abstract The development of security visualization applications must involve the user in the design process in order to create usable systems. However, it is all too easy to lose track of the user during the design and development process, even though upfront investment in extensive user requirements gathering has proven benefits. To address this challenge, we adapt a user-centered design method called *personas* that enables effective requirements capture for varying scopes of requirements-gathering efforts, and, when used properly, keeps the user involved at every step of the process from design to evaluation.

1 Introduction

The need for usability in security and visualization interfaces is well-documented in (Adams and Sasse, 1999; Cranor and Garfinkel, 2005; Erbacher et al., 2002; Zurko, 2005). Unfortunately, the failure to achieve usability due to neglect or misunderstanding user requirements is also well-known (Jarzombek, 1999; Standish, 1994, 2001). The domain of human–computer–interaction (HCI) provides a wealth of methods, best practices and approaches for mitigating software failures due to missing user requirements. However, for those unfamiliar with classic user-centered design (UCD) methods, the HCI literature can be daunting and easily misapplied without specialized expertise (Seffah, 2003). Additionally, the methods themselves can be too heavy-weight or resource-intensive to successfully apply to smaller projects, research efforts, or projects with tight deadlines, all of which are common among security visualization applications. As such, there seems to be a gap in the application of effective HCI methods in the computer security domain; specifically

J. Stoll and W.K. Edwards

Georgia Institute of Technology, School of Interactive Computing, Atlanta, Georgia

D. McColgin, M. Gregory, and V. Crow

Pacific Northwest National Laboratory, Richland, Washington

methodological details about how to ensure that user requirements are indeed reified in the resulting software design.

We address this gap through an in-depth exploration of an HCI methodology aimed at translating user requirements data *into* the software prototypes. We also demonstrate through a case study how a light-weight user-centered design process can be used to better ensure the usability of the resulting software. The method we discuss, *personas*, provides a framework for effectively using the user in order to get the user requirements right. This method is a user requirements capturing technique arising out of the Participatory Design philosophy (Grudin and Pruitt, 2002). The value we gained through an adaptation of personas to a security visualization project shows similarity to so-called ‘discount usability’ methods that can offer cost-effective measures that improve usability even with limited resources (Nielsen, 1995).

In what follows, we first describe the use of personas as a design method and situate this technique in the context of other human–computer interaction (HCI) methods. In Sect. 3, we describe a use case where this method is applied to the requirements gathering phase of a security visualization project. In Sect. 4 we discuss the lessons learned and how this method might be applied for security visualizations more broadly before concluding in Sect. 5.

2 Overview of the Personas Method and Related Work

As with any software application, the first step in building a usable security visualization is to have a good understanding of user requirements. Bowles (2006) suggests the following six steps for the requirements gathering and specification phase of user-centered design approach:

1. Define the product
2. Define the user
3. Define done
4. Define the user interface
5. Define the functional requirements
6. Define the non-functional requirements and constraints.

This and other frameworks help organize the requirements gathering phase of software development but do not necessarily ensure that the user needs actually drive the design of the security visualization tool from start to finish. Anecdotal evidence attests to the ease of overlooking user requirements and designing software for ourselves rather than the user; or designing solely based on technical considerations. We also believe questions concerning design tradeoffs must be determined by the user’s needs. We argue that it is not enough to merely define the user, but rather usable software is better achieved through utilizing the user *throughout* the design and development process; and we believe this can be accomplished for the security visualization projects in part through the use of personas.

In short, a *persona* is an archetypical user that captures a range of user needs in a coherent narrative (Cooper, 1999). The basic idea behind a persona is that one can

design for a range of users by designing for a single fictional user called a persona which effectively becomes the representative user by embodying the needs of the selected user population (Cooper, 1999). Where fundamental characteristics are in conflict, multiple personas can be developed to account for diversity in the target users. As such, the personas method provides a way to define, represent, and utilize user requirements (Adlin et al., 2006; Cooper, 1999). It does so by providing a foundation for developing usability metrics, task identification and analysis, and devising a coherent design rationale. Additionally, this method facilitates collaboration among project members and stakeholders since user requirements can be referred to more succinctly via individual personas created for the project.

While there are a number of approaches to defining user requirements (i.e. scenario-based design, storyboards, modeling, and contextual inquiry (Dix et al., 2004; Holtzblatt and Beyer, 1999; Young, 2002)), we argue for the complementary use of personas because this method creates the explicit definition of an ever-present user (Grudin and Pruitt, 2002). Personas should be grounded in a strong understanding of real-world users. Contextual inquiry offers proven methods to gather the raw data, from interviews to longitudinal observation of work practice. Once synthesized into personas, this captured data can be used for varying types of task analysis. We agree with Pruitt and Grudin (2003) that scenario-based design also works well with personas. Scenarios involve descriptions of users performing a task, including the relevant details that might drive system requirements. Scenarios bring to light many technical and functional requirements inherent in accomplishing a task, but tying them to personas can add users' broader goals, social and environmental factors, and relevant information about skills, attitudes, and other factors which need to be considered. These processes working together form a solid foundation for the design of usable systems. Personas can even be used along with other techniques to perform summative evaluation (Dix et al., 2004), as discussed in more detail in Sect. 5.

Viewing potential users as collaborators in the design process and co-creators of the eventual product or prototype is a central thrust behind the participatory design philosophy. The personas method, which grows out of participatory design, makes it more feasible to involve the user in every step of the software development process beginning with the requirements gathering phase. The consequences of unusable security visualizations can be severe (especially if they are for national security or critical infrastructure purposes). Therefore utilizing the personas method which creates an ever-present user seems appropriate, especially as personas has a rich history of successful use by both practitioners and researchers in significantly improving usability across domains (Adlin et al., 2007; Grudin, 2006; Grudin and Pruitt, 2002; Nieters et al., 2007).

2.1 Personas Method

Creating a persona consists of identifying and capturing significant details that shape the users' needs. The details considered in a persona include: goals, preferences, challenges and context that influence or dictate what users would need in

an application. User characteristics are based on data gathered in the “real world”. Adlin et al. (2007) provides explicit steps to guide the process of creating a persona. We prefer this five-step method because the persona creation process is explicitly data-driven compared to other more anecdotal approaches (Grudin and Pruitt, 2002). The five steps are:

1. Define the user population and gather data on target users

- Determine the user population.
- Collect user data related to the target population.
- Consider other users such as the international market, disabled persons and the user who is outside the target population.

2. Transform data gathered into a fictional user or persona

- Write-up the details into a persona narrative or description.
- Create a foundation document based on the narrative for each persona that is created.

3. Make the persona personable and introduce as “members” of the team

- Create photos and names for each persona.
- Cross-check each persona with “real world” users.
- Introduce each persona to team members and other stakeholders by holding a kick-off meeting.

4. Consult personas in the design decision-making process throughout the project

- Ask how a particular persona may react to particular features or lack thereof.
- Create scenarios using each personas to highlight needed features.
- Create additional documents such as feature-design maps or persona comparison poster.

5. Evaluate design based on persona requirements; evaluate whether to reuse or discard personas

- Use personas to determine the usability of the resulting prototype.
- If choosing to reuse, continue collecting data about personas to revise them as new user data becomes available and continue to further refine and enrich personas by adding new details such as learning style, book usage patterns, etc.

2.2 Related Work

The extent to which personas are used and the detail captured and used in the five steps are widely varied. Personas themselves can include elaborate personal details and there is variance in how organizations encourage their adoption within a team. Although the personas method is still relatively new, a number of companies have adopted it successfully, including Cisco, NYT.com, Best Buy, Zylom, Pfaltzgraff,

and Medco Health (Adlin et al., 2007; Nieters et al., 2007; Spool, 2006). Some of these companies have written about their best practices and yielded guidance on how to best utilize aspects of the personas method. For example, Microsoft markets their persona internally to larger development teams through the use of swag such as mugs with persona photos, mouse pads, posters and playing cards. Similarly, Yahoo! Media threw a party so the team could “meet” the personas (Adlin et al., 2006; Grudin and Pruitt, 2002; Klee and Goodwin, 2001; Spool, 2006).

However, in this investigation, we apply personas to a small research and design project. In doing so, we demonstrate how lighter-weight requirements gathering can be well represented in development settings with the smaller teams, budgets, and shorter timelines that likely characterize most current security visualization design. We provide a use case of personas by introducing a computer security persona and discuss how that persona can be used to increase the usability security visualizations.

3 Case Study: First Look

Our implementation of personas was adapted to the research needs of the first look visualization research project driven by a pressing need: As professional information analysts approach a large and dynamic source of data, how can meaningful changes in information be represented and summarized to a user? Analysts spend an inordinate amount of time probing and orienting to their information space before moving on to productive analysis of the relevant portions. Our goal is to develop methods to reduce the time it takes to prioritize work and start analysis. As new data pours in, many aspects change but the significance of these changes depends on type, magnitude, and the perspective of the user.

Our designs need to be domain independent and with such variety in users and data, we needed a way to organize our requirements. We met with a varied set of individuals who are tasked with analyzing content of dynamic, large datasets. From these discussions, it became clear that the use of personas would greatly aid in scoping user requirements. Given the research nature of the project, we could not afford the time or budget to create elaborate personas. Thus, we followed the five steps outlined above for persona building, but made alterations based on our needs and constraints. Our tailored adaptation of personas has been instrumental in guiding our work and is evidence of the robustness of this approach to different scales and domain applications.

3.1 Five Steps to Persona Implementation

3.1.1 Step One: Defining a User Population

The First Look project as a whole is aimed at supporting basic problems that are common to a broad range of information analysts. Personas helped us organize

requirements into cogent sets, contextualized by the myriad details common to a particular organization or user type. Creating the personas also brought to light a number of conflicting assumptions that may otherwise have remained hidden. We chose three initial user types: a policy analyst, an intelligence analyst, and a cyber-security analyst. Gathering data on similarities and unique properties across these three analysis domains continues to inform our design decisions.

Dix et al. (2004) and Young (2002) describe a variety of ways that user data can be collected through both direct means (engaging real-world users) and indirect means (relying on existing data about the target user). We mainly relied on indirect data gathering, borrowing from prior experience with analysts and analyzing many different analyst job postings, organizational charts, and online descriptions. While more exhaustive methods of contextual inquiry have demonstrable benefits, they can often be expensive in time and resources. There can also be problems of user access, client buy-in, and training the project team member. Particularly when resources are limited, personas from other efforts can also be reused or repurposed as we hope ours will be, and personas still have life-cycle benefits even with lower-cost data gathering.

3.1.2 Step Two: Transform Data into A persona (or Personas)

The primary outcome of this step is the creation of what is called a foundation document. This document offers a primary means for communicating the persona to all other team members and stakeholders. This method also explicitly links the details of the persona with the supporting user data, i.e. according to Grudin and Pruitt, this document should contain, in narrative form, the details gathered in the first step (2002). They suggest an extensive list of information to include. For our needs, we created the narrative foundation document with the following set of details:

- *Job position*
- *Background which included education and computer skills*
- *Overall goals*
- *Work environment which included communication/collaboration activities*
- *Job duties*
- *Characteristics of the analyst's particular information space, such as raw data sources and formats, analysis tools, analytical products they created and the target audiences.*

The details we gathered are not as extensive as Grudin and Pruitt recommend (2002). We selected them to capture the salient aspects of users for our research efforts. If target security visualization users were, for example, frequently telecommuter working from home then more detailed personas might include details about their home environment and work habits.

The temptation exists to ignore the narrative format and to create a “laundry list” of user requirements based on the user data gathered in the previous step and then to use this as the requirements list. However, there is good evidence that the narrative

Fig. 1 “Foundation Document” excerpt from the persona Frank

Frank Kreuse
Cyber Analyst, EOBU (a defense industry supplier)

Background:
Frank has a B.S. in Information Systems. He specialized in technical writing and has CISSP certification. His training, experience and certification establishes him as an expert computer user.

Work Environment and Information Management:
Frank has access to sophisticated data capture tools but better analytic tools are still lacking. Much of the data that he works with is generated by state-of-the-art network scanning tools. However, he primarily relies on Excel to keep track of the data that is of interest to him so that he can perform analyses such as “what if” queries.

Duties:
Part of Frank’s duties is to produce risk analysis reports based on test results derived from system assessment tools such as Tivoli Netview and DISA SRR.




form is more effective at keeping the user in the software design and development loop. Psychological evidence demonstrates that a narrative is more engaging and more inspiring than a rote list, boosting motivation in designers, developers, and even clients to use the requirements more effectively (Grudin et al., 2006; Nieters et al., 2007). The persona also becomes an efficacious and convenient way to quickly convey a set of requirements among the project team or with other groups. The partial document in Fig. 1 is part of foundation document for one of our personas, “Frank, the Cyber Analyst”.

Since the creation of personas enabled us to have a systematic method for representing user groups, we gained richer understanding of analysts that came from making the requirements explicit. We became more cognizant of the significant differences among different analyst types such as information assurance analysts vs. policy analysts. The bounds of our problem space were defined by the multiple personas we developed to account for the breadth of user types we needed to support; and these insights would not have been as easily or naturally gained with non-narrative, non-personal lists of user features.

3.1.3 Step Three: Personalize the Persona(s)

The personas become more real to the team members the more they know about them. Grudin and Pruitt (2002) argues that photos, especially candid, more realistic photos (from freely available stock) lend to the “personability” of the personas. The combination of name and picture helps the team to remember and refer to them. The personas we created to match our three user types were each given a name and an associated photo. They were introduced to the First Look team by sharing the persona foundation documents at team meetings. Everyone on the team then had the opportunity to review the personas and provide feedback. As the project continued with technical discussions and research, we (almost naturally) began to refer to

Table 1 Three personas created for the First Look project

	<p>Name: Rob McCormick Organization: Food and Drug Administration Role: Consumer Safety Officer, Office of Food Additive Safety, Center for Food Safety and Nutrition</p>
	<p>Name: Terry Whitter Organization: Customs & Border Patrol Role: Intelligence Research Specialist, Office of Intelligence</p>
	<p>Name: Frank Kreuse Organization: Government contractor Role: Cyber Analyst</p>

personas by name as we discussed aspects that would affect them to reinforce their utility after their introduction and make clear the role they would play as team members. Our three personas were “Frank, the Cyber Analyst”, “Terry, the Intelligence Researcher”, “Rob, the Consumer Safety Officer”, represented below in Table 1.

3.1.4 Step Four: Consulting Personas in the Design Decision-Making Process

Once personas are created, they can be “consulted” with when design decisions arise particularly during the development phase. In order to facilitate this process, two matrices can be created: (1) a data-to-features matrix and (2) a persona-weighted feature matrix. To generate these matrices, we created multiple scenarios for our personas to explore requirements. These scenarios stemmed from specific tasks and situations where the persona analyst must deal with tracking information change in a large information space. For example, Frank, as a cyber analyst, was run through a scenario where he had to validate the emergence of a threat pattern using disparate data sources. How Frank performed in the scenario was constrained by the persona characteristics that he was given, i.e. his job description, the resources that he had available and time constraints.

These scenarios, tightly coupled with the personas, allowed us ultimately to create a weighted “feature matrix” (sample show in Fig. 2) that let us explore the importance of individual system features relative to the other features based on persona needs. This matrix was then used to explore initial design directions and for creating a design map that will be used to inform our prototype design.

The data and the features inferred from the persona and scenario data are tabulated and used to inform the persona-weighted feature matrix. Figure 3 is an example of how these requirements for each personal can be weighted to aid in decision

User Data			Features
Frank Cyber Analyst	Terry Intelligence Researcher	Rob Consumer Safety Officer	
"Frank is not always the person required to respond to intrusions. ...he must be aware of and be able to access information about how each incident is being handled by the team. ...network monitoring information comes from a variety of tools so Frank needs to know where the information came from."	"Terry must perform her analysis in a highly collaborative environment and the data she receives from the Intelligence support staff comes from many varied sources. ... at times she must do quality control on the information she receives from the newer support staff by making sure they are coming from credible sources."	"Rob works for an agency that utilizes antiquated computer and information systems. One of the daily challenges that Rob faces is keeping track of the many alerts he receives about food safety incidents to be investigated. He relies on where the alert came from to determine priority for investigation."	1. Functional: Need a summary of the metadata about the information being used in analysis.

Fig. 2 "Data-to-features" matrix excerpt

Features	Persona Weights			Weighted Priority
	Frank (weight = 40)	Terry (weight = 40)	Rob (weight = 20)	
1. Functional: Need a summary of the metadata about the information being used in analysis.	2	2	2	200
2. Non-functional: Need information change alerts to be rapidly customize-able.	2	1	0	120
weight = percentages totaling to 100% or on a scale such as 1-5 score = -1 harms persona 0 does not matter to persona if the feature is there or not +1 helpful to the persona +2 is a must-have feature for the persona				

Fig. 3 Persona-weighted feature matrix excerpt

making in the design process (Adlin et al., 2007). For each of the features identified in the data-to-feature matrix, an inference is made by the designer or development team to determine whether such a feature is helpful to the persona, harmful or neither. The personas themselves are also assigned a weight according to the priority of the visualization project. For project First Look, our focus was on designing primarily for the cyber analyst, Frank rather than the policy analyst, Rob. Therefore, Frank was assigned a heavier weighting than Rob. Consequently, the project development process will focus primarily on features which Frank finds important. Also, when design conflicts arise between catering towards Rob's need vs. Frank's, the assigned weighting should provide direction as to which tradeoffs to make in the design.

Our choice to use a feature matrix is partly based on Meuller's (Adlin et al., 2007) recommendations for deciding which attributes, based on which users, are crucial for inclusion in the design. Also, both matrices can be combined and used as a handy reference card throughout the development process, similar to the one created and used at Yahoo! Media with success. Designers and developers were able to refer to the card as they were in the process of actually coding the prototype.

This is particularly important as the experience of software engineering tells us that critical design decisions often occur later on realities of development unfold (Dix et al., 2004). Again this points to the utility of the personas method to enable the user to be ever-present throughout the duration of the project.

By comparing the personas we were able to identify which features would need to vary across the range of personas and which were held in common. This informed important decisions about system architecture. In fact, we concluded that a stand-alone software tool would actually hinder our users by forcing them to change their workflow and have yet another tool to learn when our goal is to provide rapid, usable orientation. Thus, we explored options such as creating extensions to existing tools and providing services that could be combined into custom applications. The varying time constraints and use environments of the diverse personas led us to explore multimodal interface options and various blends of push and pull technologies, as illustrated by Fig. 4. (It should be noted, however, that the task of generating actual designs from the functional and non-functional requirements identified is often more dependent on the designer’s inspiration than other factors.)

Personas helped to reveal some natural connections between technical resources and user needs through functional and non-functional requirements. At other times, one or more of the project members asserted opinions about certain features that needed to be included while others dissented. In these situations, discussing the feature in light of the persona’s need helped to bring project members to an agreement regarding the design decision. Assuming the role of a persona can illuminate the problem in a new light. When no personas seem applicable it may highlight the need for additional unforeseen requirements and an opportunity to further define the problem space.

Using these three matrices to inform design decisions throughout the development lifecycle can provide a means to coherently connect design decisions made with user requirements data.

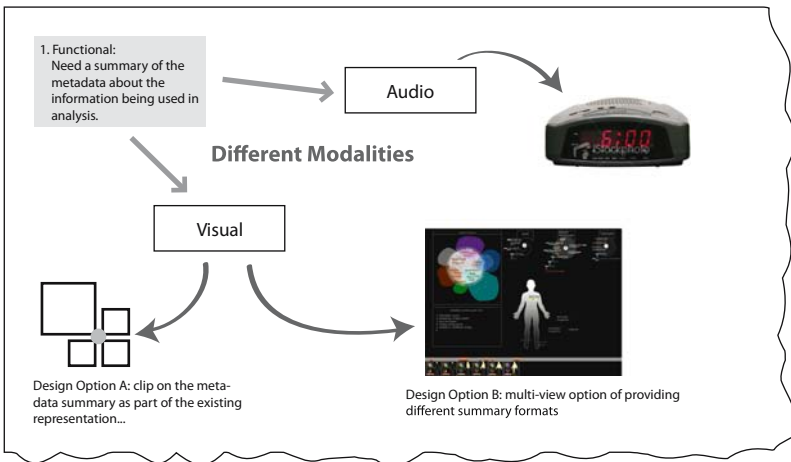


Fig. 4 “Feature-to-design” map excerpt

3.1.5 Step Five: Evaluate Results and Persona Retention

In this step, the personas are used to evaluate the resulting system. In the First Look project we have still been in the design/prototyping stage and have not yet used the personas for this last step of evaluation. However, we intend to use the weighted feature matrix and design map for this purpose. For our research-oriented project, we also plan on using personas as a perspective in cognitive walkthroughs, a baseline in heuristic evaluations, and possibly as guidelines for prioritizing quality assurance. The literature provides numerous examples of the successful use of personas for evaluating the software in a variety of ways (Adlin et al., 2007).

We plan to retain the personas foundation document and other related documents to make them available as part of a library for other projects with similar goals and user populations. Projects with no resources for requirements gathering can benefit from the rich representation of users and requirements, or they may provide a starting point for bootstrapping efforts on related problems.

3.2 Discussion

We have found that in this small research project the use of personas was crucial to design processes, and continues to be useful throughout the implementation. One challenge we encountered that has been experienced by others (Adlin et al., 2006; Holtzblatt and Beyer, 1999; Klee and Goodwin, 2001; Spool, 2006) is gaining acceptance of personas as additional members of the team. It is not surprising that real-world team members need time to learn how to work with their persona colleagues. For our smaller core team, the foundation documents and consistent invocation were sufficient. As we offer the personas as resources to more people and other projects we may investigate other options, such as Yahoo! Media's persona party, though this approach seems more appropriate for much larger teams who are already facing cohesion challenges. However, as it did for the First Look project, personas can serve as a common rallying point for project engagement which may have been otherwise lacking.

4 Application to Security Visualizations

In the case study we presented here, personas were useful for logically and coherently grouping requirements, designing with real user needs in mind, and giving all team members a shared understanding of project goals and priorities through a memorable narrative. The personas method is appropriate for capturing and utilizing user requirements particularly for security visualization system design vis-à-vis other security problems. Unlike other security problems such as sensor data capture or anomaly detection, a security visualization system must be designed with the

human user in mind because analysts will be interacting directly with the interface to help with their tasks. Thus, making sure the user needs are appropriately addressed is particularly critical and the personas method helps to ensure that this is the case throughout the duration of the security visualization project.

Though our specific research project was not geared solely to computer security alone, Frank, our cyber analyst persona, gave us insight into the processes in common across analysis domains as well as the unique properties of the cyber analyst. For example, like other profession information analysts, Frank required tools to help manage the information being gathered to support his hypothesis of threat pattern development. However, unlike other analysts, Frank did not have to cope with new information arriving at irregular intervals. He did not have a need for indicator level alerts to prompt him when new information had arrived, a feature required by both the intelligence and policy analysts.

Also, our project goals led us focus on how Frank coordinates with other staff and integrates information from email, alerts, and social connections with changing data about current system performance to protect the network. We acknowledge that for many applications within security visualization, there are multiple user types and a project with a different focus may capture different requirements in a given persona. Having a suite of personas to represent the range of functionality needed may help security visualization designers better understand and articulate the degree of flexibility needed for each functional/non-functional requirements identified.

Projects specifically targeting cyber-security may need to create sub-type personas to more precisely cover the range of intended users such as system administrators, cyber or information assurance analysts, and network administrators, among others. Variations within the user types can be significant enough to warrant finer-grained representations of the user population that can inform design (Conti et al., 2005; D'Amico and Kocka, 2005; Goodall and Lutters, 2004; Yurcik et al., 2003). The use of sub-types of users could be exploited in the same way that we exploited our three user types: each of them can help define features, and give a weighting to those features to inform and direct user-centered design alternatives.

Relevant details could also vary. Applications interested in remote system monitoring might embed in personas information about the home environment or experience and skill with mobile devices. A visually rich application supporting interaction with massive data might also have a persona that, among other things, represents the substantial color-blind user population.

As a developing field, security visualization encompasses a broad range of research efforts, prototypes, and commercial products. With so much at stake, we must ensure that these systems are usable and target the right problems by effectively involving users. Security visualization applications are highly varied, and while the literature provides examples of effective persona use for large-scale software projects, our case study demonstrates that they can be adapted to smaller scale projects, including research, to effectively be centered on the user in the process.

5 Conclusion

Security visualization applications intended to help users deal with massive data must be more than just aesthetically pleasing or purely functional. They must be share the users' priorities and be usable or their benefits are compromised. Once requirements are gathered, personas offer an adaptable way to keep the user involved in all of the design, development, and evaluation that will affect usability and adoption of the application. The accessibility of the personas method allows all team members from developers to quality assurance testers to utilize the personas, and thereby sharing a common understanding of the goals and priorities. They can enable even collaborators, clients, and other projects with similar needs to easily share this improved understanding.

Our experience also demonstrates the adaptability of the persona method to research and efforts that cannot prioritize expensive requirements-gathering approaches. We do not contend that personas is the sole method for achieving user-centered design, but rather that they particularly enable the user to remain in the process as a focal point throughout the life-cycle of the project. In other words, they help to ensure we are solving problems that are important to the users in truly usable ways.

References

- Adams A, Sasse MA (1999) Users are not the enemy. *Communications of the ACM* 42, 12, pages 40–46. doi: 10.1145/322796.322806
- Adlin T, McGrane K, Pruitt J, Rosenstein A, Goodwin K, Muller M, Hynes C (2006) Panel: Putting Personas to Work. CHI '06 Proceedings
- Adlin T, Pruitt J, Grudin J (2007) The Persona Lifecycle: What Personas Are, Why They Work, and How to Create and Use Them. CHI '07 Course Notes, San Jose, CA
- Bowles JB (2006) Better software reliability by getting the requirements right. *Reliability and Maintainability Symposium*. doi: 10.1109/RAMS.2006.1677359
- Conti G, Ahamad M, Norback R (2005) Filtering, fusion and dynamic information presentation: towards a general information firewall. *IEEE International Conference on Intelligence and Security Informatics*
- Cooper A (1999) *The Inmates are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*. Macmillan
- D'Amico A, Kocka M (2005) Information assurance visualizations for specific stages of situational awareness and intended uses: lessons learned. *Proceedings of the IEEE Workshops on Visualization for Computer Security*. doi: 10.1109/VIZSEC.2005.13
- Dix A, Finlay J, Abowd G, Beale R (2004) *Human-Computer Interaction*, 3rd Ed. London: Prentice-Hall
- Erbacher RF, Walker KL, Frincke DA (2002) Intrusion and misuse detection in large-scale systems. *IEEE Computer Graphics and Applications* 22, 1
- Goodall J, Lutters W (2004) The work of intrusion detection: rethinking the role of security analysts. *10th Americas Conference on Information Systems*
- Grudin J (2006) Why personas work: The psychological evidence. In Pruitt J, Adlin T (eds). *The Persona Lifecycle*. Amsterdam: Elsevier

- Grudin J, Pruitt J (2002) Personas, participatory design and product development: an infrastructure for engagement. The Participatory Design Conference '02
- Grudin J, Pruitt J, Adlin T (2006) The Persona Lifecycle: Keeping People in Mind throughout Product Design. Los Altos, CA: Morgan Kaufmann. <http://research.microsoft.com/~jgrudin/publications/personas/PersonasChapter.pdf>
- Holtzblatt K, Beyer H (1999) Contextual design: using customer work models to drive systems design. Extended Abstracts on Human Factors in Computing Systems. doi:10.1145/632716.632801
- Jarzombek, SJ (1999) The 5th Annual Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium (JAWS S3), Proceedings
- Klee M, Goodwin K (2001) Personas and Goal-Directed Design: An Interview with Kim Goodwin. http://www.uie.com/articles/goodwin_interview/. Accessed on 10 September 2007
- Nielsen J (1995) Scenario-based design: envisioning work and technology in system development book contents: Scenarios in discount usability engineering, pages 59–83. New York, NY: Wiley
- Nieters J, Ivaturi S, Ahmed I (2007) Making personas memorable. In CHI '07 Extended Abstracts on Human Factors in Computing Systems. doi: 10.1145/1240866.1240905
- Pruitt J, Grudin J (2003) Personas: Practice and Theory. Conference on Designing For User Experiences '03. doi: 10.1145/997078.997089
- Seffah A (2003) Learning the Ropes: Human-centered Design Skills and Patterns for Software Engineers' Education. ACM Interactions '03
- Spool J (2006) Yahoo's Approach to Keeping Personas Alive. <http://www.uie.com/brainsparks/2006/05/18/yahoos-approach-to-keeping-personas-alive/>. Accessed on 10 September 2007
- The Standish Group International Inc. (1994) The CHAOS Report
- The Standish Group International Inc. (2001) The CHAOS Report
- Young R (2002) Recommended Requirements Gathering Practices. CrossTalk: The Journal of Defense Software Engineering. <http://www.stsc.hill.af.mil/crosstalk/2002/04/young.html>. Accessed on 10 September 2007
- Yurcik W, Barlow J, Rosendale J (2003) Maintaining perspective on who is the enemy in the security systems administration of computer networks. ACM CHI Workshop on System Administrators Are Users, Too '03
- Zurko M (2005) User-centered security: stepping up to the grand challenge. Computer Security Applications Conference (ACSAC) '05

Measuring the Complexity of Computer Security Visualization Designs

X. Suo, Y. Zhu, and G. Scott Owen

Abstract We present a novel method to measure the complexity of computer security visualization designs. The complexity is measured in terms of visual integration, number of separable dimensions for each visual unit, the complexity of interpreting the visual attributes, and the efficiency of visual search. Visualization developers can use this method to quickly evaluate multiple design choices in the early stage of their design before any user study can be conducted. To demonstrate this method, we have conducted complexity analysis on two open source security visualization tools – TNV and RUMINT.

1 Introduction

Evaluation is an integral part of the information visualization design process. For example, visualization developers often need to make a design choice among multiple design options. In a group setting, different developers may come up with different designs for the same set of data. In these situations, developers need methods to assess and compare different designs. The most common evaluation method is user study, which can be used to measure task completion time, the number of errors, and user satisfaction. However, user studies are often time consuming, difficult to control and can only be conducted after a prototype has been developed. In addition, the results of these studies often do not explain what causes usability problems.

In this paper, we propose an alternative evaluation method – complexity analysis. Following this method, developers systematically evaluate a set of factors that influence the efficiency of processing visual information. Here the complexity is measured in terms of visual integration, number of separable dimensions for each

X. Suo, Y. Zhu, and G.S. Owen

Department of Computer Science, Georgia State University, Atlanta, Georgia, USA, e-mail: yzhu@cs.gsu.edu

visual unit, the complexity of interpreting the visual attributes, and the efficiency of visual search. Based on well established psychological theories on human cognition, this method allows visualization developers to quickly compare the complexity of multiple visualization designs and is particularly useful during the early design stage before any user study can be conducted.

Comparing with other heuristic visualization evaluation methods that deal with a broader range of usability issues, the proposed method focuses on complexity. However, the complexity of a visualization design may greatly influence common usability issues, such as task completion time, error rate, learnability, and user satisfaction. Therefore it would be beneficial to correlate the proposed complexity analysis with other heuristic methods as well as user studies. For example, the results of the complexity analysis can help generate hypotheses for user studies to verify.

The paper is organized as follows. In Sect. 2, we discuss related work. In Sect. 3, we present our method, using open source security visualization tool TNV as an example to demonstrate the steps of complexity analysis. As an additional case study, we apply our method to another open source security visualization tool RUMINT. In Sect. 4, we discuss our future work. Section 5 is the conclusion.

2 Related Work

Assessing the effectiveness of the visualization design is a complicated issue. There is no doubt that formal user study is and should be the primary form of evaluation. However, as Tory and Moller (2005) pointed out, formal user studies have their limitations. Controlled user studies require lots of time and resources and are hard to manage. As a result, Tory and Moller have argued that expert review can be a valuable alternative to formal user studies.

Expert review is a heuristic evaluation method, and there have been a number of heuristics for evaluating information. For example, Tufte (2001) has proposed some heuristics for evaluating the visual display of quantitative information, such as the data-ink ratio. Shneiderman's "Visual Information Seeking Mantra" (Shneiderman, 1996) is sometimes used as a heuristic for evaluation. Amar and Stasko (2005) have proposed a knowledge task-based framework for the design and evaluation of information visualizations. This is a high level framework that covers a broad range of issues.

Overall, the current heuristics for evaluating information visualization are far from comprehensive and not well organized. To address this issue, Zuk et al. (2006) have performed a meta-analysis of existing heuristics with an aim to develop a set of appropriate heuristics for the evaluation of information visualization. As a first step, they attempt to build a hierarchical structure to organize the many heuristic rules of evaluation. Their proposed information visualization heuristic structure includes the following groups: perceptual, cognition, usability, etc. Drawing from three sets of heuristics (including Shneiderman's and Amar and Stasko's heuristics), their analysis suggests that different heuristics are needed to evaluate different aspect of a visualization design.

Our complexity analysis is a type of heuristic evaluation of information visualization. It can be seen as a specific type of expert review with a focus on evaluating the complexity. The specific guidelines and procedures presented in this paper is a new addition to the heuristics of information visualization evaluation. Our work is compatible with the analysis by Zuk et al. (2006). The analysis of visual integration, the number of separable dimensions for each visual unit, and the complexity of interpreting the visual attributes belong to the cognition group in their heuristics hierarchy, while the analysis of visual search efficiency belongs to the perception group.

The uniqueness of our proposed method is its focus on the complexity of visualization design. The complexity of visualization design is rarely mentioned in previous heuristic evaluation methods. Only a few visualization researchers have touched on the issues of complexity in visual display, but none of them have dealt with it in a systematic way. For example, Bertin (1983) and Trafton et al. (2000) have used the number of dimensions as a measure for the complexity of visual displays, and considered visualizations with more than three variables to be complex. Brath (1997) has proposed a heuristic method to measure the effectiveness of the mapping from the data dimension to the visual dimension by classifying the visual mappings into one of the four categories. In terms of analyzing complexity, our evaluation method is more systematic than these previous methods and considers many more factors.

The proposed complexity analysis is based on a number of psychological theories, including Guided Visual Search theory (Wolfe and Horowitz, 2004), Gestalt theory (Wertheimer and King, 2004), and cognitive load theory (Clark et al., 2006). According to the cognitive load theory, there are three types of cognitive load: intrinsic cognitive load, extraneous cognitive load, and germane cognitive load. The mental effort to comprehend a data visualization is part of the extraneous cognitive load, which is a major factor that influences the task performance. The proposed visualization complexity analysis is an attempt to measure the extraneous cognitive load of visualization comprehension.

It should be noted that the proposed complexity analysis is not a direct measure of a visualization design's usability or task performance. Although we hypothesize that the complexity of a visualization design is likely to have a significant impact on task performance and usability, the exact relationship between complexity and usability or task performance is not well understood and will be the focus of our future work. It also means that our method should be combined with user studies and other heuristic methods that directly measure usability and task performance.

3 Technical Approach

The processing of a data visualization depends on a host of psychological processes, including information read-off, integration, and inference (Trafton et al., 2000). The main goal of the proposed complexity analysis is to systematically evaluate the

Table 1 Overview of the complexity analysis

Complexity analysis steps	Step 1	Step 2	Step 3	Step 4
Purpose	Divide a data visualization into hierarchical layers	Analyzes the efficiency of visual integration	Analyze the efficiency of reading visual units	Analyze the efficiency of visual search
Theoretical basis	(Trafton et al., 2000; Zhou and Feiner, 1997)	(Wertheimer and King, 2004; Trafton et al., 2000)	(Clark et al., 2006; Garner, 1974)	(Wolfe and Horowitz, 2004)
Outcome	N/A	Visual integration complexity tree	Visual mapping complexity tree	N/A
Data structure	N/A	Maximum number of visual integrations	<ul style="list-style-type: none"> • Number of different types of visual units • Number of separable dimensions per visual unit • Complexity scores for visual units 	Target-distracter difference score and visual search complexity scores for color, motion, size, and orientation

major factors that influence the efficiency of information read-off and integration. Table 1 gives an overview of the complexity analysis process.

The complexity analysis is carried out in the following steps:

- (1) We divide the visualization into five layers: workspace, visual frame, visual pattern, visual units, and visual attributes.
- (2) Next, we analyze the efficiency of integrating visual elements. We build a tree structure that depicts how visual frames are organized in each workspace, and how visual patterns are organized in each visual frame. We call this a visual integration complexity tree (Fig. 2) because it shows how a reader might mentally integrate visual frames and visual patterns. The number of nodes on this tree is the maximum number of visual integrations that a reader might perform.
- (3) We then analyze the efficiency of interpreting visual units. For each type of visual unit, we identify the visual attributes that are used to encode data parameters. Each of the encoded visual attributes is called a dimension. For each encoded visual attribute, we estimate the complexity of mapping the visual attribute to its corresponding data parameter. The outcome is a visual mapping complexity tree (Fig. 6).
- (4) Finally we analyze the efficiency of visual search. We evaluate the target-distracter difference for four attributes (color, motion, size, and orientation) that are important for efficient visual search.

In the following sections, we will discuss the complexity analysis in more detail, using TNV as an example to demonstrate our ideas. TNV (version 0.3.7) is a computer network traffic visualization tool developed by Goodall et al. (2005). Later we will apply our method to RUMINT, another open source network security visualization tool developed by Conti (2005, 2007). The dataset we used to evaluate both tools is called *cigi.pcap*, available at <http://wiki.ethereal.com/CIGI>.

3.1 Hierarchical Analysis of Data Visualization

In order to systematically analyze a visualization design, we divide data visualization into five hierarchical layers: workspace, visual frame, visual patterns, visual units, and visual attributes. Each layer is a component of the previous layer. That is, a workspace is one or more visual frames that are designed for a specific purpose. A visual frame is a window within a workspace and contains multiple visual patterns. A visual pattern is a set of visual units that are readily perceived as a group; and they are identified based on four Gestalt laws: proximity, good continuation, similarity, and common fate. Some examples of visual units include point, line, 2D shape (glyph), 3D object, text, and image. Each visual unit is defined by seven visual attributes (Bertin, 1983): position, size, shape, value, color, orientation, and texture.

The hierarchical analysis of data visualization is inspired in part by Zhou and Feiner (1997), who used a hierarchical framework to construct visualizations. Our hierarchical analysis is also influenced by Trafton et al. (2000), who propose a three layer hierarchical framework – information read-off, integration, and inference – to describe the cognitive process of graph comprehension. In our hierarchical framework, visual units and visual attributes are related to the information read-off layer, while visual patterns and visual frames are related to the visual integration layer.

In the case of TNV, the workspace is the entire visual interface, while the four visual frames are marked by red boxes (see Fig. 1).

3.2 Visual Integration

Larkin and Simon (1987) point out that a main advantage of visualization is that it helps group together information that is used together, thus avoiding large amounts of search. In complex problem solving, the visual units need to be integrated (Trafton et al., 2000), which adds to the extraneous cognitive load.

In this study, the cognitive load of visual integration is estimated by building a visual integration complexity tree (Fig. 2). For each visual frame, we identify the visual patterns in that frame based on four Gestalt laws: proximity, good continuation, similarity, and common fate. The number of nodes on the visual integration complexity represents the upper bound of visual integration a reader might perform. The visual integration complexity for TNV is shown in Fig. 3.

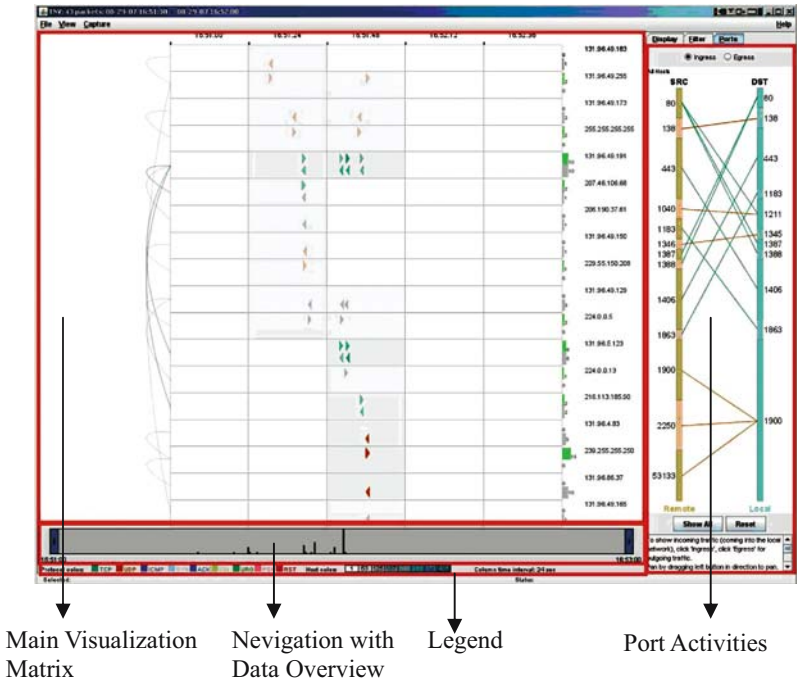


Fig. 1 Four different frames of TNV

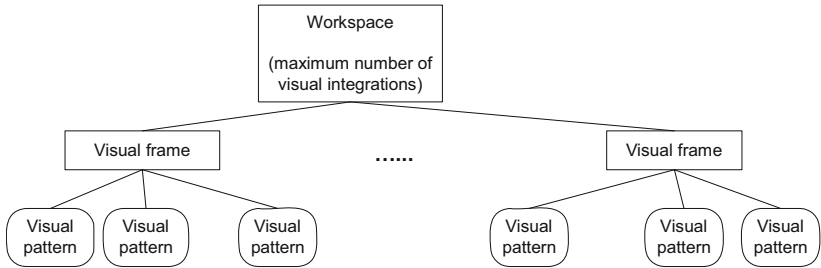


Fig. 2 Visual integration complexity tree

3.3 Separable Dimensions for Visual Units

In a visualization design, different data parameters are mapped to different visual attributes of different visual units. Each encoded visual attribute is called a dimension. Readers need to identify and remember these visual mappings, thus adding to the extraneous cognitive load. Psychological studies have showed that human eyes can see only three variables at the same time, and the difficulty of using a graph is determined by how many fixations are required (Kosslyn, 1985). Some researchers have suggested using the number of dimensions as an indicator for the complexity

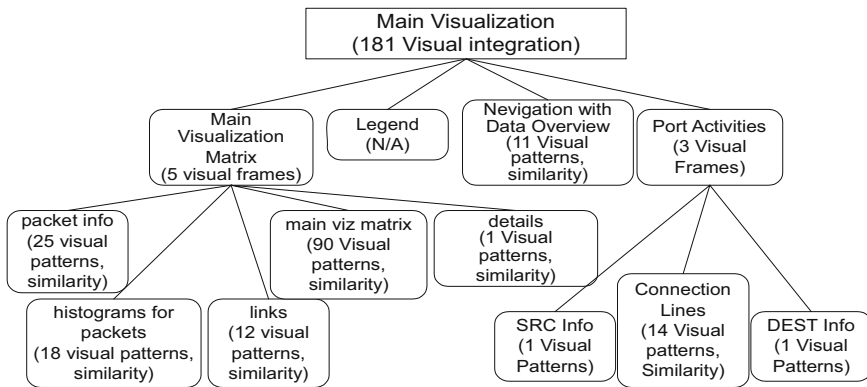


Fig. 3 Visual integration tree for TNV. Each of the child nodes contains the number of visual patterns and the Gestalt laws they are identified with. Each parent node contains the number of visual frames. The number of visual integrations is calculated by multiplying the number of visual frames and the sum of the visual patterns

of visualization (Kosslyn, 1989; Brath, 1997; Trafton et al., 2000). However, we believe that it is important to consider the difference in visual processing of integral and separable dimensions.

Garner (1974) has discussed the difference between integral and separable dimensions and how they are processed differently by human. In short, integral dimensions are processed together, while separable dimensions are processed individually. As a result, integral dimensions are likely to be processed faster than the separable dimensions. Garner has also provided a number of guidelines for identifying integral and separable dimensions. For example, in perceptual classification, stimulus sets defined by integral dimensions are classified primarily in relation to similarities; sets defined by separable dimensions are classified in relation to dimensional structure. Also in perceptual classification, dimensional preferences exist only for separable dimensions.

Based on the guidelines in (Garner, 1974), we have identified the following integral and separable dimensions:

- X and Y coordinates are integral dimensions. (Here we only consider two dimensional displays.)
- Color and value are integral dimensions.
- Shape, size, and orientation are each considered as a separable dimension.

It should be noted that the guidelines in (Garner, 1974) are rather abstract. Different people may have different interpretations and come up with different groupings of integral dimensions. Much work needs to be done to develop more specific guidelines for identifying integral and separable dimensions in visualization design.

In the case of TNV, there are five different dimensions in the main visual frame (Fig. 4), and three different dimensions in the port visualization frame (Fig. 5).

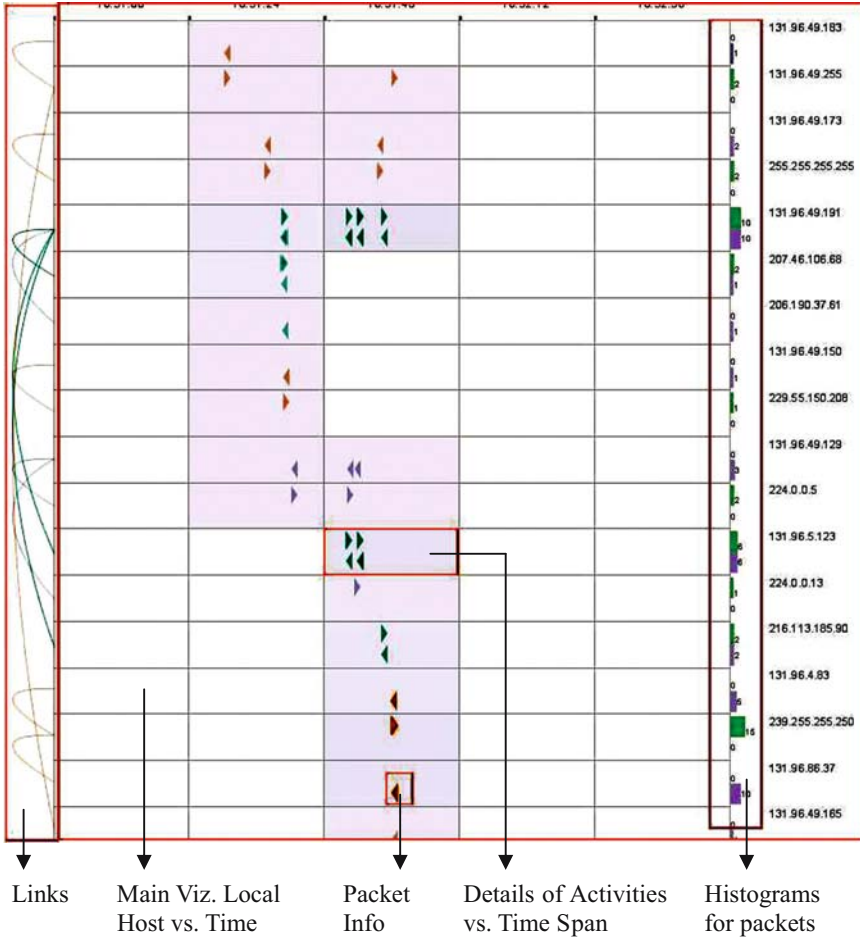


Fig. 4 Five different dimensions in TNV main visualization matrix. (a) Histograms for packets: categorized based on their shape/size/color. (b) Details of activities; categorized based on color/shape. (c) Main visualization matrix: categorized based on X–Y coordinate; in this case, they are local host vs. time. (d) Package information triangles: categorized based on shape/size/color. (e) Links: categorized based on shape/size/color

3.4 Interpreting the Values of Visual Attributes

Readers need to interpret the values of each visual attribute, which is another source of extraneous cognitive load. In our analysis, for each separable dimension, we assign a score for the complexity of interpreting the values of the visual attribute based on the following criteria:

For integral dimensions, the visual attributes are considered together. For example, X and Y coordinates are considered together and a single complexity score is assigned to both coordinates. In the end, each separable dimension has a complexity

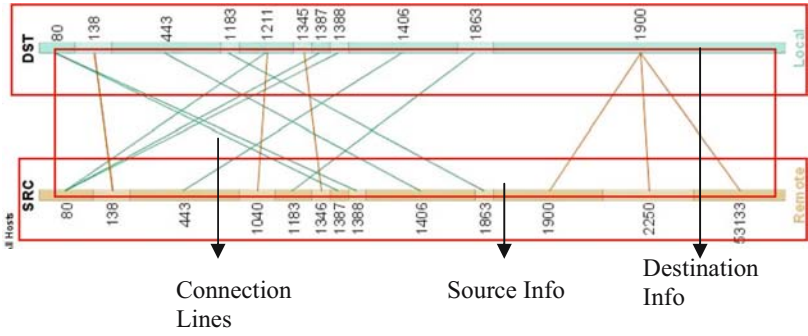


Fig. 5 Three different dimensions in port visualization. (a) Source and destination information: categorized based on the coordinate (vertical axis). (b) Connection lines between the two axes: categorized based on shape/color

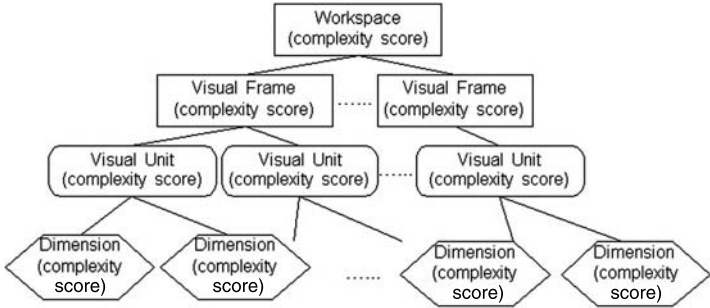


Fig. 6 Visual mapping complexity tree

score. The complexity score for a type of visual unit is the sum of complexity scores of its separable dimensions. The complexity score for a visual frame is the sum of scores of different types of visual units it contains, and so on. The structure of a typical visual mapping complexity tree is shown in Fig. 6.

The visual mapping complexity tree for TNV is shown in Fig. 7.

3.5 Efficiency of Visual Search

One of the main advantages of visualization is that they can support efficient visual search. According to Wolfe and Horowitz (2004), target-distracter difference is the key to efficient visual search, and there are four major factors that affect the target-distracter differences – color, motion, size, and orientation. Target-distracter difference indicates how a target stands out from the background, which can be any other surrounding objects or the neighboring background colors.

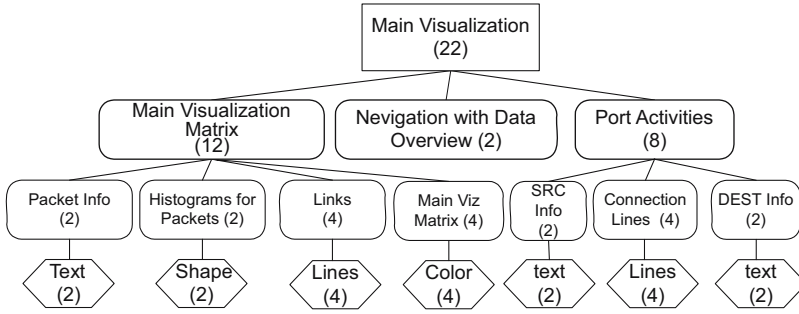


Fig. 7 Visual mapping complexity tree for TNV, each number below the visual units are the complexity scores based on Table 2. Every parent node's score is calculated as the sum of its children's scores

The target-background difference is determined differently for color, motion, size, and orientation through the following equations.

$$c = \left(\left| \frac{\sum_{i=1}^n T_r}{N} - D_r \right| + \left| \frac{\sum_{i=1}^n T_g}{N} - D_g \right| + \left| \frac{\sum_{i=1}^n T_b}{N} - D_b \right| \right) / 765, \quad (1)$$

$$M = \frac{\left| \frac{\sum_{i=1}^n T_f}{N} - D_f \right|}{F} \quad (2)$$

$$s = \frac{\sum_{i=1}^n \frac{|T_s - D_s|}{s}}{N} \quad (3)$$

$$o = \frac{\sum_{i=1}^n \frac{|T_o - D_o|}{N}}{180}. \quad (4)$$

Equation (1) is for calculating the target-background difference in color. Here T_r , T_g , and T_b are the R/G/B values of target color, D_r , D_g , D_b are the R/G/B values of the distracter whose color is the closest to the target color. N and n are the number of color components represented in the visualization. Number 765 is the distance between the two most distant colors.

Equation (2) is for calculating the target-background difference in motion. Here T_f and D_f are the speed of motion for target and distracter, respectively. F is the speed of the fastest moving item in the visualization. N and n are the number of moving items.

Equation (3) is for calculating the target-background difference in size. Here T_s and D_s are the sizes of target and distracter items, respectively. S is the total area of visual frame that contains the target and distracters. N and n are the number of visual entities.

Table 2 Complexity scores for interpreting the meaning of visual units

Complexity score	Criteria
5	Very difficult to interpret. There is no legend. A typical reader has to memorize the mapping between the value of the visual attribute and the value of the corresponding data parameter
4	More difficult to interpret. A typical reader needs to frequently refer to a legend to interpret the value of the visual attributes
3	Somewhat difficult to interpret. A typical reader needs to refer to a legend from time to time
2	Relatively easy to interpret. A typical reader only needs to refer to a legend occasionally
1	Easy to interpret. This is based on common knowledge. There is no need to memorize or refer to a legend

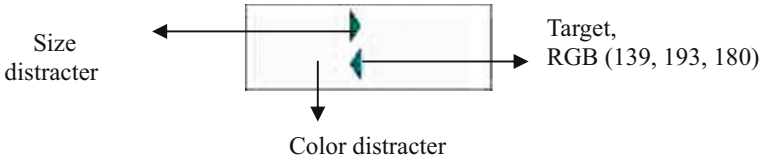


Fig. 8 Part of the main TNV visualization frame

Table 3 Target-distracter difference scores for TNV

	Color	Motion	Size	Orientation
Target-distracter difference scores	0.2850	N/A	0	1

Equation (4) is for calculating the target-background difference in orientation. Here T_o and D_o represents the orientation of the target and distracters respectively. N and n are the numbers of distracters.

In the case of TNV, we calculated the target-distracter difference for the color, size and orientation of the visual units in the main visualization frame (Fig. 8). The result is shown in Table 3.

3.6 Case Study with RUMINT

RUMINT is an open source network security visualization tool developed by Conti (2005, 2007). Figure 9 shows a RUMINT thumbnail view after it finishes capturing 949 packets from the test dataset.

Figures 10 and 11 are the visual integration complexity tree and visual mapping complexity for RUMINT, respectively. Since RUMINT is a rather complicated visualization tool with many frames and dimensions, we have to simplify the trees

Fig. 9 RUMINT thumbnail overview

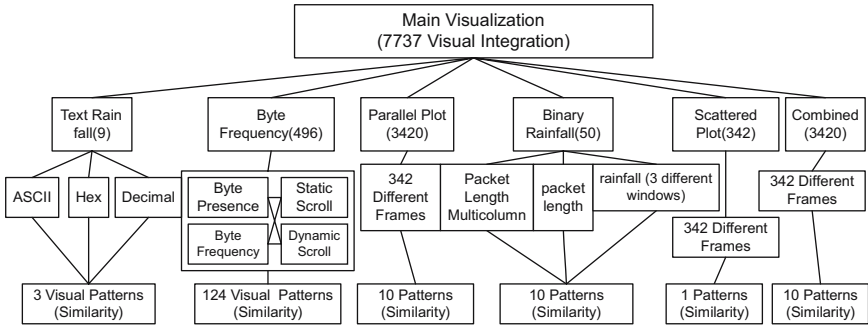
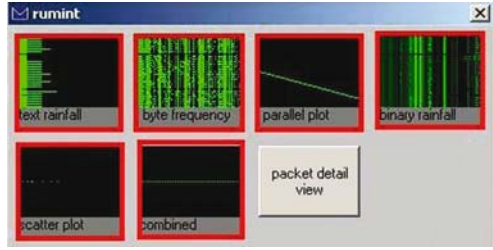


Fig. 10 Visual integration complexity tree for RUMINT

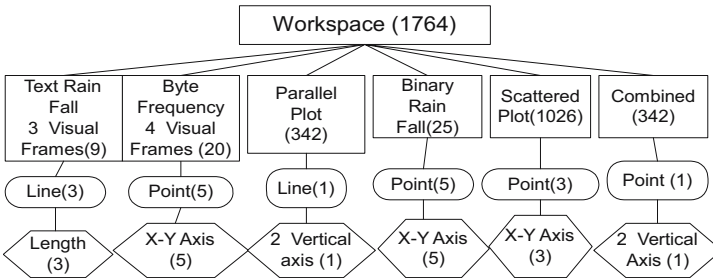


Fig. 11 Visual mapping complexity tree for RUMINT

in order to fit them into this paper. Table 4 shows the target-distracter difference scores for two visual frames of RUMINT.

It is necessary to point out that the case studies discussed above is not an attempt to directly compare TNV and RUMINT. Because the visual interfaces of these two software tools are quite different, it is hard to make meaningful comparisons between their complexity scores. For example, the two software systems may have the same complexity scores for different reasons. In other words, our proposed complexity analysis is primarily for comparing different design choices in the same visualization tool. It is not meant to compare visualization tools with drastically different visual interfaces.

Table 4 Target-distracter difference scores for two visual frames in RUMINT

Visual frames	Visual search guidance			
	Color	Motion	Size (pix ²)	Orientation
Overview	0.33333	n/a	0.43064	0
Byte frequency	0.33333	n/a	0.32169	0

4 Future Work

In the future, we plan to continue refining our complexity analysis methods. For example, the process of identifying integral and separable dimensions for visualization designs need to be further clarified and improved. The current equations for calculating the target-distracter differences are based on our intuition and need to be verified and revised through user studies.

Another focus of our future work is to study the relationship between the visualization complexity and usability. It will be interesting to see how the various complexity parameters affect the task completion time, error rate, learnability, and user satisfaction. We will need to correlate our complexity analysis results with formal user studies. We plan to use our complexity analysis to generate hypotheses of usability, and then design specific user studies to test these hypotheses.

Although the proposed complexity analysis method is developed in the context of computer security visualization, it can be easily extended to other information visualization areas. In particular, we plan to apply our complexity analysis method to bioinformatics visualization.

5 Conclusion

In this paper, we have presented a systematic methodology to measure the complexity of information visualization. Here the complexity is measured in terms of visual integration, number of separable dimensions for each visual unit, the complexity of interpreting the visual attributes, and the efficiency of visual search. These measures are based on well established psychological theories. Together they indicate the amount of cognitive load involved in comprehending a particular visualization design. The underlying hypothesis is that by reducing visualization complexity, we can improve the usability of the visualization.

The proposed complexity analysis is particularly useful during early design phase before any user studies can be conducted. It is intended to help developers quickly compare different design choices for a visualization tool. We have demonstrated this method by evaluating the complexity of two computer security visualization tools – TNV and RUMINT.

The proposed complexity analysis is not intended to be a comprehensive assessment of the usability of a visualization design. Rather it is focused on measuring the

complexity of visualization designs in terms of visual cognition. Therefore it should be combined with user studies as well as other heuristic evaluation methods.

Acknowledgements The authors would like to thank Hsiu-Chung Wang for her contribution to the early works of this research.

References

- Amar RA and Stasko JT (2005) Knowledge Precepts for Design and Evaluation of Information Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 11(4): 432–442.
- Bertin J (1983) *Semiology of Graphics*, University of Wisconsin Press.
- Brath R (1997) Metrics for Effective Information Visualization. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis)*, IEEE.
- Clark R, Nguyen F, Sweller J (2006) *Efficiency in Learning: Evidence-Based Guidelines to Manage Cognitive Load*: Pfeiffer.
- Conti G (2005) Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries. In *Proceedings of Workshop on Visualization for Computer Security (VizSEC)*.
- Conti G (2007) *Security Data Visualization: Graphical Techniques for Network Analysis*: No Starch Press.
- Garner WR (1974) *The Processing of Information and Structure*: Lawrence Erlbaum.
- Goodall JR, Ozok, AA, Lutters WG, Rheingans P, Komlodi, A (2005) A User-Centered Approach to Visualizing Network Traffic for Intrusion Detection. In *Proceedings of Conference on Human Factors in Computing Systems (CHI) extended abstracts*, ACM.
- Kosslyn SM (1985) Graphics and Human Information Processing: A Review of Five Books. *Journal of the American Statistical Association* 80(391): 499–512.
- Kosslyn SM (1989) Understanding Charts and Graphs. *Applied Cognitive Psychology* 3: 185–226.
- Larkin JH, Simon HA (1987) Why A Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11: 65–99.
- Shneiderman B (1996) The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the IEEE Conference on Visual Languages*, IEEE.
- Tory M, Moller T (2005) Evaluating Visualizations: Do Expert Reviews Work? *IEEE Computer Graphics and Applications* 25(5): 8–11.
- Trafton JG, Kirschenbaum SS, et al. (2000) Turning Pictures into Numbers: Extracting and Generating Information from Complex Visualizations. *International Journal of Human-Computer Studies* 53(8): 827–850.
- Tufte ER (2001) *The Visual Display of Quantitative Information*, Graphics Press.
- Wertheimer M, King D (2004) *Max Wertheimer and Gestalt Theory*: Transaction Publishers.
- Wolfe JM, Horowitz TS (2004) What Attributes Guide the Deployment of Visual Attention and How do They Do It? *Nature Reviews/Neuroscience* 5: 1–7.
- Zhou MX, Feiner SK (1997) Top-Down Hierarchical Planning of Coherent Visual Discourse. In *Proceeding of International Conference on Intelligent User Interface (IUI)*, ACM.
- Zuk T, Schlesier L, et al. (2006) Heuristics for Information Visualization Evaluation. In *Proceedings of the Working Conference on Advanced User Interface (AVI)*, ACM.

Integrated Environment Management for Information Operations Testbeds

T.H. Yu, B.W. Fuller, J.H. Bannick, L.M. Rossey, and R.K. Cunningham

Abstract Network testbeds are indispensable for developing and testing information operations (IO) technologies. Lincoln Laboratory has been developing LARIAT to support IO test design, development, and execution with high-fidelity user simulations. As LARIAT becomes more advanced, enabling larger and more realistic and complex tests, effective management software has proven essential. In this paper, we present the Director, a graphical user interface that enables experimenters to quickly define, control, and monitor reliable IO tests on a LARIAT testbed. We describe how the interface simplifies these key elements of testbed operation by providing the experimenter with an appropriate system abstraction, support for basic and advanced usage, scalable performance and visualization in large networks, and interpretable and correct feedback.¹

1 Introduction

Network testbed research has advanced significantly in recent years. Testbeds provide a controlled setting for studying various aspects of a networked environment: hardware, protocols, algorithms, applications, social behaviors, and so on. Public testbeds, such as Emulab (White et al., 2002) and PlanetLab (Peterson et al., 2006), achieve a level of configurability, repeatability, and realism that is well suited for most networking research. DETER (Benzel et al., 2006) is an Emulab cluster with enhanced containment mechanisms designed specifically for cyber-security research.

T.H. Yu, B.W. Fuller, J.H. Bannick, L.M. Rossey, and R.K. Cunningham
MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420, e-mail: tamara@ll.mit.edu,
bfuller@ll.mit.edu, john.bannick@ll.mit.edu, lee@ll.mit.edu, rkc@ll.mit.edu

¹ This work was sponsored by the Defense Advanced Research Project Agency (DARPA) and the Department of Defense 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.

Since 1997, Lincoln Laboratory has been developing software that generates realistic traffic for testbed networks by simulating users (Lippmann et al., 2000), eventually packaging the software as a tool called LARIAT, or Lincoln Adaptable Real-time Information Assurance Testbed (Rossey et al., 2002). The main distinction between LARIAT and other testbeds is that LARIAT generates a virtual user world that overlays on top of the experimenter's own testbed. LARIAT supports a wide range of software, hardware, and network topologies. It applies various models of user behaviors to directly exercise applications specified by the experimenter, resulting in highly realistic usage of the applications and the underlying operating systems, hosts, and networks.

LARIAT can be used for a broad range of information operations (IO) tests. It is particularly useful for security research and evaluation because it accurately represents vulnerabilities that occur as a result of flaws in the design and implementation of services, protocols, and applications by directly running the software used on the modeled network.

While LARIAT excels in providing host-level realism for tests, early versions of LARIAT were complicated to set up and use. In these first versions, experimenters configured test parameters via direct interaction with a database, manually set up user accounts and applications on each host, parsed log files to determine if the configuration scripts ran correctly, and manually verified that each traffic generator was running. This tedious process limited the size and complexity of networks that could be tested. These early experiments made it clear that efficient testbed management is important.

We have found that certain key tasks of IO experiments – test specification, test control (including software deployment, troubleshooting and validation), and test monitoring – are particularly problematic for experimenters. To address these tasks, we developed the Director, a graphical user interface for managing LARIAT testbeds. In this paper, we present two contributions: (1) an interface that greatly simplifies these key tasks and thereby significantly improves the usability of LARIAT, and (2) a demonstration of how several interface and visualization techniques can be used in large-scale testbed management software.

The remainder of this paper describes the Director in detail. Section 2 surveys related work. Section 3 describes the technical approach of our work. It first presents some background knowledge on the general LARIAT framework in Sect. 3.1, then identifies key design goals in Sect. 3.2, and finally presents the solutions in Sect. 3.3. Section 4 discusses lessons learned and future work. Section 5 closes with a summary.

2 Related Work

There have been extensive studies on testbeds for networking research and development. Solutions span simulation, emulation, overlay, and live networks, providing researchers a wide range of options for experimentation.

Simulation tools such as ns (Bajaj et al., 1999) and Simnet (Kamara et al., 2005) require minimal hardware and are easy to setup, control and use to obtain verifiable results. They efficiently simulate the normal operation of a variety of protocols and arbitrary network topologies, but cannot be used to explore flaws in protocol implementations or unusual application behavior without re-programming the flawed implementation or behavior. To create conditions that closely mimic live networks while maintaining experimental control, researchers turn to emulation tools.

Emulab (White et al., 2002) is a widely used network emulator. It allows experimenters to run the system under test on physical hosts with real operating systems and application software. Testbed assets can be rapidly configured to create any network topology. Hosts can be easily set to a target state for the experiment and reset to a clean state when the experiment is done. Furthermore, Emulab leverages tools like Dummynet (Rizzo, 1997) and ns to introduce synthetic network conditions and traffic between physical nodes. The mix of real resources and simulation provide both depth and breadth for the experimental environment. Emulab traffic is generated at the host level and is protocol-based. It does not model real applications and interactions between users, which can be critical to many security tests.

DETER (Benzel et al., 2006) is an Emulab cluster focused on information security concerns and repeatability of experiments. DETER is unique in that it is an open resource that allows for the running of malicious code. However, the platform's public nature limits the scope of experiments that can be run on the testbed. For strong assurance of security and confidentiality, coupled with significantly lowered hardware cost in recent years, many still prefer private, isolated testbeds.

Unlike Emulab, which operates on local dedicated clusters, PlanetLab (Peterson et al., 2006) is a global platform that runs on an overlay network of nodes on the Internet. It provides a testing environment with live network conditions for large-scale network services such as distributed hash tables (DHTs) (Rhea et al., 2005). PlanetLab users contribute computing assets to the platform in exchange for access to the collective resources. PlanetLab manages the nodes on behalf of their owners and allocates slices of those nodes to the users in a safe, fair manner. While PlanetLab is invaluable for the network research community, it is not for experiments that require direct access to and full control of testbed assets or those that require strict containment.

Many tools seek to effectively manage testbed systems. Plush (Albrecht et al., 2006) provides a framework for deploying applications in distributed environments such as PlanetLab. In Plush, experimenters define applications under test in XML. Based on the definition, Plush controls allocation of available resources, deployment of software, configuration of nodes, and execution of the application. The Plush framework is extensible: it integrates third-party tools where appropriate for distribution of software and monitoring of hosts. The current interface is designed primarily for application developers with area expertise on the definition and deployment tools in use.

The Experiment Workbench (Eide et al., 2007) is a system for managing experiments in Emulab. Using the Workbench, experimenters apply scientific process to testing by running experiments in controlled iterations. Each iteration is comprised

of setting or restoring the test environment to a pristine state, setting or changing parameters, performing the experiment, collecting data, and archiving any material (e.g., code, scripts, and configuration reports) pertaining to the experiment. This allows for easy verification of test results and clear comparison across runs. The Experiment Workbench provides a graphical user interface for easy access and control of the experiments.

SEER (Schwab et al., 2007) is the testbed management tool that is most similar to the Director in environment, goals, and implementation. It provides a user interface for DETER, enabling experiment definition and deployment of traffic generation scripts on the DETER testbed. In contrast, the Director focuses on the definition and deployment of virtual users for testbeds of arbitrary complexity.

Managing a testbed is similar to administering a network. In both cases, the operators have to monitor activities on many machines and detect anomalies, i.e., errors or attacks. Because of this similarity, two network traffic analysis tools are included in this discussion.

Network Eye (Ball et al., 2004; Fink et al., 2005) is a network traffic monitoring tool that provides visualization of the inbound and outbound traffic of a “home” network, allowing users to easily detect any unusual patterns in foreign connections. It further correlates those connections with host processes to aid forensic investigation of suspected attacks. While the aggressor–defender model does not apply to testbed management, Network Eye’s techniques for summarizing network status, highlighting anomalies, and mapping low-level observations to high-level activities are relevant and valuable.

TNV (Goodall et al., 2005) is also a visualization tool designed for network traffic analysis. TNV creates timelines of network events by hosts, on top of which it overlays host connections. This allows analysts to view and correlate sequences of events and reconstruct scenarios. The tool provides additional coordinated views of detailed information such as ports and packet contents. The Director has a similar aim of displaying virtual user activities and communications over time for test verification and monitoring purposes.

3 Technical Approach

3.1 LARIAT Overview

LARIAT emerged from the 1998–1999 DARPA off-line Intrusion Detection System (IDS) evaluations (Lippmann et al., 2000). In the first evaluation, Lincoln Laboratory used a 16-node testbed network to emulate thousands of hosts modeling an Air Force base communicating with the Internet. Virtual and real users exercised real applications to generate traffic and launch existing and novel attacks on the testbed. The corpora of network traffic created from the testbed were used to measure the IDSs attack detection and false-alarm rates, resulting in the first formal, repeatable, statistically significant evaluation of IDSs. While the traffic corpora proved

valuable, network characteristics have changed and new devices, operating systems, and applications have been developed.

To provide researchers the flexibility of having their own testbeds, we created LARIAT. In preparation for LARIAT deployment, the testbed staff build the network(s) under test, install the operating systems and applications on the hosts, and deploy network or host-based defensive tools if necessary. On top of this basic testbed, LARIAT deploys virtual hosts and virtual users, creates an emulated Internet, and puts data content on servers to transform the generic network into the specific, desired environment. Driven by Markov models, which are based on patterns of activity derived from real user and network behavior (Boothe-Rabek, 2003), the virtual users exercise applications, access server contents, and interact with other virtual users.

LARIAT tests run on the topology available. The testbed topology generally varies from test to test and may even change during a test, such as when hosts migrate from network to network as a real user of a laptop might. The testbed may include widely disparate hardware – desktops, laptops, and other network devices with different architectures and manufacturers. Individual hosts may be running a wide range of operating systems, including Linux, Unix, and Windows, in multiple configurations. LARIAT supports tests in most of such customized, heterogeneous network environments.

In the LARIAT framework, a host may operate as a client, providing a platform for a virtual user or group of virtual users, or as a server, providing information at the request of one of the clients. Linux and Unix hosts can be configured in one of two modes: single-host, which operates much like a normal desktop, or multi-host, which sources traffic from multiple (10–10,000) IP addresses to emulate multiple hosts. LARIAT Windows servers do not support multi-host virtualization. Similarly, on a Windows client, only one user can act at a time and interact directly with applications (Boothe-Rabek, 2003).

Virtual users communicate via a range of protocols and applications, including e-mail (POP, SMTP, and Exchange) via Microsoft Outlook and mail, file sharing (SMB) via Windows Network and Samba, and web (HTTP, HTTPS, and FTP) via Internet Explorer and Mozilla. They can be assigned different roles, such as supervisors, secretaries, developers, or system administrators. The role assigned to a user dictates their daily usage pattern, the applications used, the way in which they interact with the applications, the types of contents and services accessed, and the pattern of collaboration with other users.

On a LARIAT testbed, servers host data and provide services that need to be accessed by the virtual users. LARIAT automatically configures many Linux and Windows servers, including Apache, Internet Information Services (IIS), Very Secure FTP Daemon (vsftpd), CVS, and Certificate Authority (CA) Server. A substantial repertoire of data contents, such as sample web sites, is provided and can be automatically deployed on the appropriate servers. LARIAT also sets up accounts that allow users to log onto remote hosts via SSH or Telnet. Finally, LARIAT supports emulation of an Internet cloud hosting thousands of web sites and the root

Domain Name Servers, creating the appearance that an isolated testbed is connected to the Internet.

The available options in LARIAT provide an enormous breadth of possibilities. The approach of modeling users of real applications allows for testing of open source and commercially available software, open and proprietary protocols, and the performance of security tools using attacks against actual applications. The challenge is to enable all of these options in a manner understandable to the experimenter.

3.2 Design Goals

LARIAT is a powerful but complex system. When designing a user interface for LARIAT, we must address the following challenges:

- *Appropriate system abstraction.* A key goal of the user interface is to provide functionality without burdening experimenters with implementation details. The user interface should help experimenters design the test they want to run for the network they have and the users and traffic profiles they want to model.
- *Basic and advanced usage.* It is important to identify a set of features common to all tests and make them obvious and easy to use, so that experimenters can focus on correctly designing and configuring the custom components necessary for their particular tests.
- *Scalable performance and visualization.* The user interface must perform efficiently on small and large testbeds. Since LARIAT supports modeling individual and groups of users and individual hosts and enterprise networks, status needs to be provided across a wide range of scales, and interface elements must be able to depict both overviews and details.
- *Interpretable and correct feedback.* The user interface should provide useful feedback on all tasks performed, on all system components under control. Accurate reports of testbed status and detailed troubleshooting information need to be collected in a centralized display tool.

3.3 Interface and Visualization

3.3.1 Director Overview

The Director is a Java-based graphical user interface that serves as the control center of the testbed. It provides a gateway to testbed resources as well as the database, where configuration, status, and log information is stored. It allows experimenters to design and execute tests in a high-level environment. The home screen in Fig. 1 conveys the overall workflow on a LARIAT testbed. Items in the center column depict the basic steps of running a test: describing the *Testbed Environment*, defining the

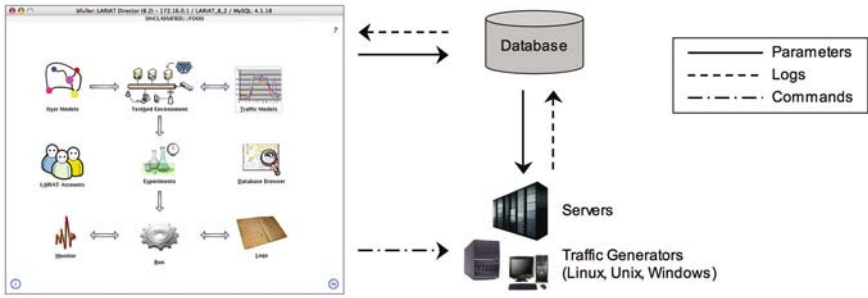


Fig. 1 Data and message exchanges between the Director (left) and other testbed components (right)

Experiments, and performing experiment Runs. Items on the sides provide additional control for advanced users to refine User Models, adjust Traffic Profiles, and enable access control with LARIAT Accounts. As shown in Fig. 1, the Director coordinates actions on the testbed, collects and processes logs, and displays status to the experimenter. The testbed Monitor and detailed Logs provide further aids in troubleshooting.

The Director is specifically designed to address several key tasks of IO testing: test specification, testbed control, and testbed monitoring. In Sects. 3.3.2–3.3.4, we will describe the visualization components of this interface in detail, focusing on how they meet our design goals in helping experimenters perform these tasks.

3.3.2 Test Specification

A test specification consists of three components: the testbed environment (including both the network topology and the virtual environment overlay), the models that control user actions and shape traffic, and the parameters that define a run.

In LARIAT, the testbed environment is organized hierarchically. At the top is the site, i.e., the testbed facility. Within a site are one or more networks. Networks are broken up into network segments, which in turn contain the physical hosts. Each host has a configuration that describes what the machine represents in the context of a test (e.g., a personal desktop, a public shared machine, an entire organization with many virtual hosts, or a server). Multiple configurations can be specified for a host. This lets experimenters quickly change settings by selecting an alternate configuration. LARIAT automatically generates virtual hosts and virtual users for the machines, which are positioned at the bottom of the testbed hierarchy.

Figure 2 illustrates the hierarchical organization of a LARIAT testbed in the Director. On the left side, the experimenter can see that within the testbed MIT Lincoln Laboratory there is a network called Internet.world.net, which is responsible for emulating the Internet. This network contains one logical subnet with several hosts: INT-PC-010, hive, page, melo, and spew. The real host spew emulates a set

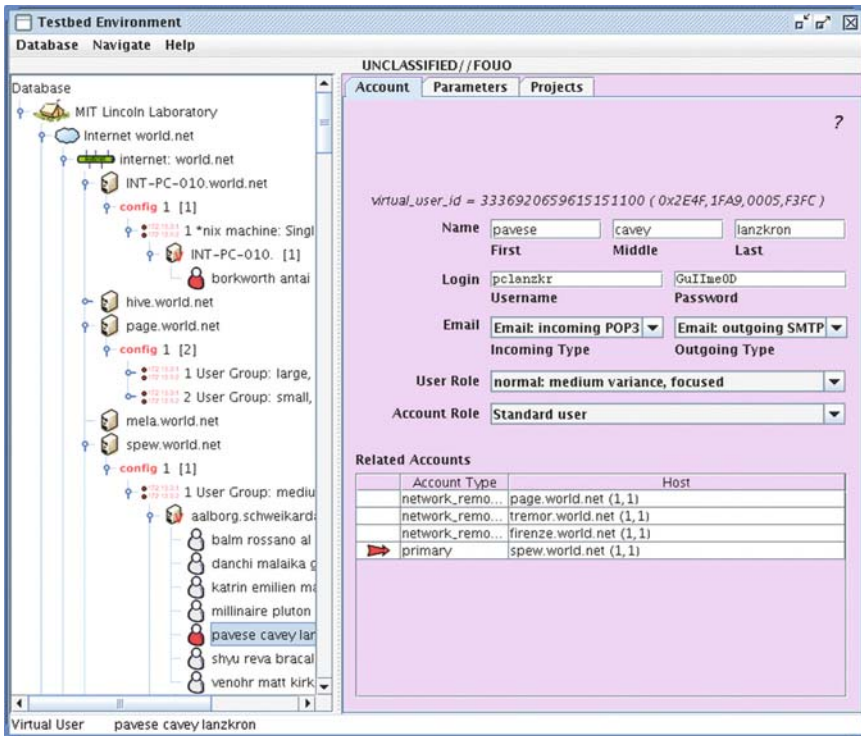


Fig. 2 The testbed environment is displaying part of the hierarchical tree with a virtual user selected

of virtual hosts. Virtual users are assigned accounts on these virtual hosts. In the display, the account of Pavese Cavey Lanzkron is selected, showing his login username and password, his Email client's supported protocols, his user role and account type. *Related Accounts* shows that the user has additional accounts elsewhere on the network. Under the *Parameters* tab, the experimenter can change the user's behavior by adjusting parameters such as the workday hours, the probability of a mistyped command, and the likelihood to access projects. Finally, *Projects* provide the experimenter a mean of modeling collaboration among users, who may share documents and exchange emails frequently.

The testbed environment presents a natural and coherent view of the testbed. The tree structure provides an abstraction of the relations among numerous testbed components of all types. By expanding and collapsing nodes, it allows the experimenter to manage the level of complexity of the testbed configuration exposed, reduce visual clutter, and focus on specific areas of interest. The tree is highly scalable. Even in a large network with hundreds of machines and hundreds of thousands of users, experimenters can still systematically navigate their way around and drill down to any parts necessary. Furthermore, similar to a filesystem browser, the dual-panel interface allows experimenters to quickly navigate the tree on the left and edit

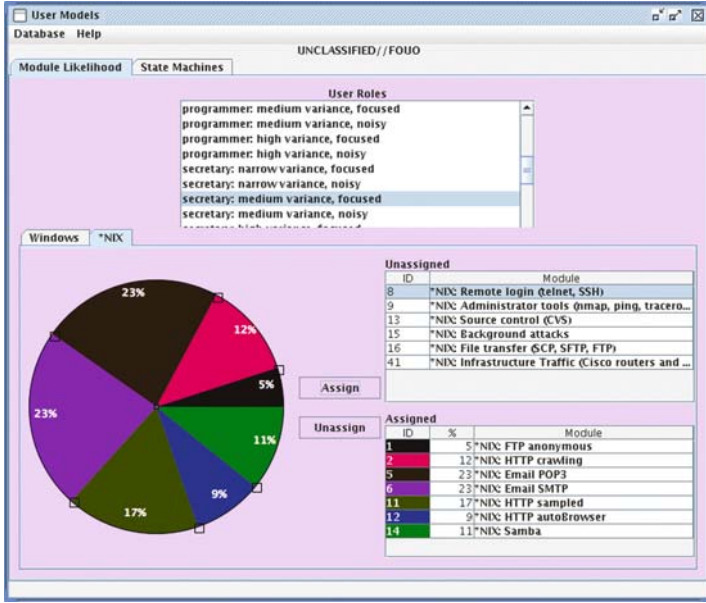


Fig. 3 Experimenters can fine-tune the traffic profiles

the properties of the selected node on the right. Experimenters can add, remove, and search for nodes. Compared to alternative representations such as network graphs, this interface is more compact for viewing large testbed setups and more streamlined for the typical testbed design workflow, which is heavy in edits.

For a basic LARIAT setup, the testbed environment is all that an experimenter needs to configure. LARIAT provides a default set of models that generate reasonable traffic. However, if customization is necessary, the Director also provides more advanced controls. For example, Fig. 3 displays an interface for defining models of virtual users' computer activities as split among different applications and protocols, where each pie segment represents an application or protocol. From this interface, experimenters can quickly gauge the expected traffic composition and adjust it by dragging the handles on the edge of the pie. There are tools for defining other aspects of the traffic models, such as application state machines and daily traffic rates (not shown here). These graphical representations help experimenters understand and modify virtual user behaviors.

Another way to customize the communication patterns on the testbed is through network flows. Network flows are predominately used to represent security policies of organizations. They describe what types of inbound and outbound traffic are allowed in a network as well as any exceptions. Figures 4 and 5 show two complementary views of the same flows on a testbed.

In the graphical view (Fig. 4), directed edges represent traffic flows among network nodes. In this example, most of subnets can communicate with each other, forming the well-connected cluster in the lower left corner. The cluster on the right

Fig. 4 A graphical view of network flows

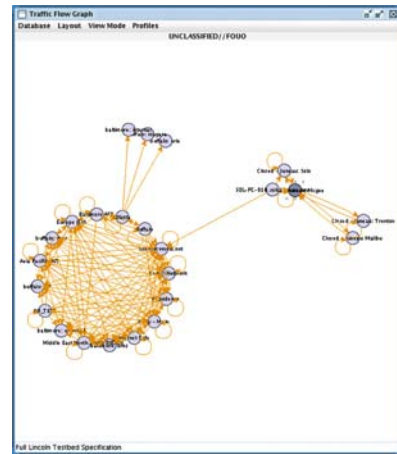
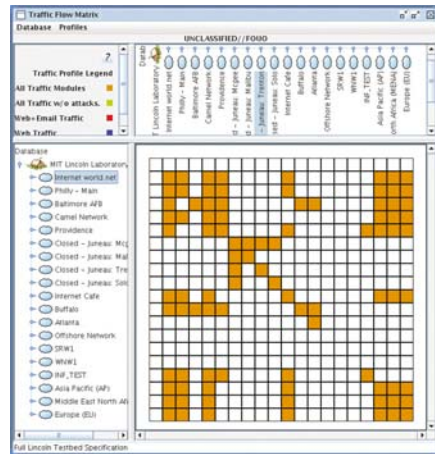


Fig. 5 A matrix view of network flows



represents a small community that talks among themselves but not with anyone outside. An interesting exception here is the host *SOL-PC-010.solo.juneau.gov*, which can go to *Internet.world.net*. The graphical view is an intuitive way of visualizing the flows. Major traffic patterns are easily identifiable, as are certain outliers. However, the graphical view does not scale well. When the number of nodes increases (e.g., when there are more host-level rules), the number of edges could grow quadratically, making the view unreadable very quickly. The graphical view is also relatively difficult to edit.

The matrix view (Fig.5) solves some of these problems. The traffic flows are presented in an adjacency matrix, in which rows and columns correspond to nodes in the testbed's network topological tree. The cell (r,c) is colored if there is a rule that allows the node in row r to send traffic to the node in column c . Unlike the graphical view, in which the layout could change drastically when some of the flows

change, the matrix view places nodes in an organized and fixed manner. This makes flow editing much easier. The ability to collapse parts of the tree allows this view to work efficiently with large testbeds and lots of flow rules. The main disadvantage is that when a node is collapsed, the lower-level flows are completely hidden. This could result in misinterpretations. We plan to address this in future versions.

In general, the directed graph is intuitive but hard to manipulate; the matrix is space-efficient and easy to edit but poor in showing the “big picture.” They can be used to complement each other. Similar dual-representation has been shown in *MatrixExplorer* (Henry and Fekete, 2006). We adopt this representation because it does not force experimenters to create and verify flows strictly in one view or the other, but allows them to choose the best view for the task at hand.

3.3.3 Testbed Control

Once an experiment is designed, the experimenter has the challenge of realizing the test design on the physical testbed. Setting up hosts individually through manual configuration and ad hoc scripts works in the short term for small testbeds, but the solution is neither repeatable nor scalable. The Director streamlines software deployment and configuration by introducing a well-defined sequence of tasks with precise commands for any host and test configurations. An example is the *Traffic Preparation:All* task selected in Fig. 6, which gives hosts configuration information needed for the experiment. Tasks are organized into phases. In this case, the selected task belongs to the phase *Validation*, which contains tests to ensure the network is properly configured and that hosts will generate traffic as expected. Phases make up the final workflow that the experimenter can step through. In each step, the experimenter can simply select a task, select the hosts to run the task, start the task, and monitor the progress from the Director (Fig. 6).

The Director allows experimenters to execute tasks on testbed hosts in parallel. Based on the task, the host’s operating system, and the connectivity of the network, the Director automatically determines how to connect to the host (via SSH, Telnet, etc.) and gives the appropriate command to start the task. The Director uses a thread pool to handle connections to multiple hosts concurrently. To support rapid configuration in large networks and to meet our scalability design goal, the Director can delegate tasks to others. Instead of connecting to all hosts, the Director can connect to one or more group leaders and ask them to either perform the task for all or relay the task order to their group members. Delegation greatly reduces the processing burden on the Director, allowing it to quickly satisfy the experimenter’s requests.

As the hosts perform the task, they send status updates to the database. The Director periodically queries the database and displays the progress to the experimenter. The task status display is organized around the hierarchical network topological tree. Each row corresponds to a node. The display shows the overall status for each host followed by pie charts that demonstrate progress in various aspects of the task. The outcome of each step, or wedge, is color-coded: green means okay, yellow means warning, and orange, red, and magenta mean errors at different levels of severity.

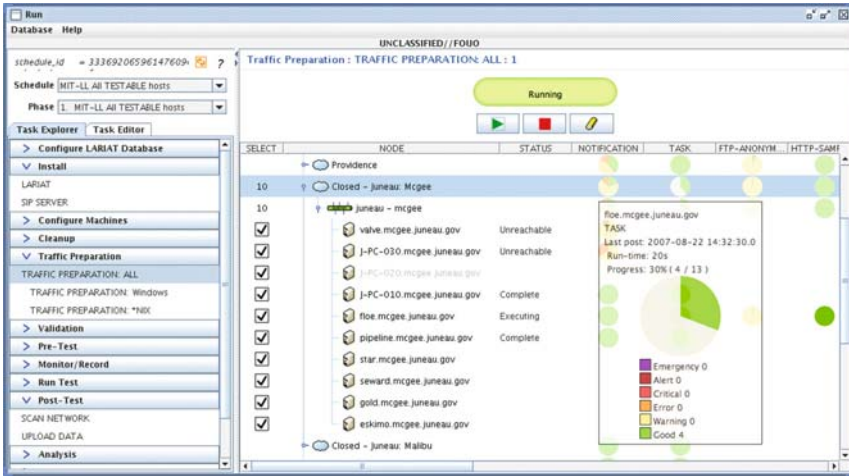


Fig. 6 An interface for running and monitoring tasks on the testbed, which displays the workflow on the left and the task status on the right

Small pie charts are effective in providing the overview at a glance. The extent of problems on the range can be assessed by the amount of red shades shown. The pattern of the colors may also suggest the source of the problem, e.g., whether it is local or at a gateway. If the experimenter wants more details, he or she can mouse over a small pie chart, which would pop up a magnified version. The experimenter can drill down even further by double-clicking the small pie chart to view a set of relevant logs useful for troubleshooting.

In addition to showing host-level status, the task status display aggregates status up to the network segment, network, and site levels. Aggregate status is displayed as semi-transparent to differentiate from host status. The need for aggregation arises from the difficulty of debugging large testbed networks. For testbeds with hundreds of nodes, there is too much information when viewing all hosts at once: experimenters would have to keep scrolling and they cannot focus on a subset of hosts. With aggregation, experimenters can collapse some networks and concentrate on the rest. The faintly displayed aggregate status would not compete for the experimenters attention, but at the same time ensures that any warning or error conditions are still shown. This meets our fourth design goal providing overall status of testbed tasks with the ability to pinpoint specific problems and receive verbose information about errors.

Finally, the Director’s task framework is designed to be highly extensible. Instructions for tasks are encoded in database tables and normally abstracted away from experimenters by the interface. However, experimenters have the option of creating custom tasks by adding them to the task tables. Then they can run these custom tasks the same way as normal tasks from the Director. Some examples include tasks to start traffic capture and archive test results.

3.3.4 Testbed Monitoring

While executing tasks, the Director is effective in providing status and troubleshooting information; to meet our goal of interpretable feedback, it is necessary to give information during an experiment as well. Given the vast number of events that occur during a test, this data must be presented in a way that gives the experimenter useful information without being overwhelming. During an experiment, the Director provides status at both the host and user level: a host health monitor (Fig. 7) tracks machines' CPU, disk, and memory usage, and a traffic event viewer (Fig. 8) displays user-driven activities in a timeline format.

The host health monitor (Fig. 7) is a visualization tool designed to allow experimenters to quickly detect and mitigate problems related to system resource usage. The visualization has two parts: a heatmap on the left, which indicates the overall condition of hosts on the testbed, and a set of charts on the right, which provides detailed historical information on a selected host. The heatmap has three columns, which represent CPU, disk, and memory usage. Each row corresponds to a host. The shade of each cell indicates the condition or activity level of a given type of system resources on a given host: green means low usage, black means medium usage, and red means high usage. As the number of hosts increases, the display tries to shrink the row height to fit all hosts. When the number of hosts gets so big that they cannot all be presented in the display in a legible manner, adjacent hosts are

Fig. 7 A dynamic heatmap that represents the health status of testbed hosts, along with details for the selected host

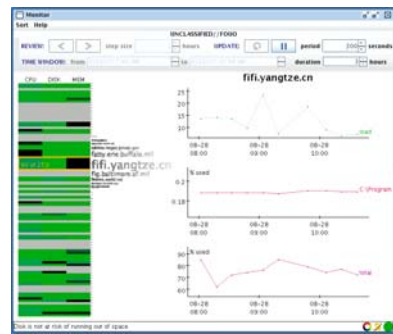
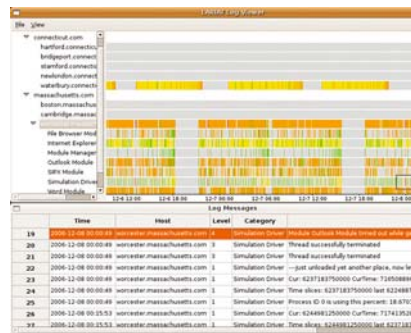


Fig. 8 A timeline of virtual user traffic events expanded by traffic type on testbed hosts



aggregated into one row that displays the worst conditions in the group. In this way, an experimenter can monitor the full testbed at a glance and would never miss hosts in distress even when statuses are aggregated. This display is also highly dynamic. When the experimenter moves the mouse over the heatmap, the rows near the mouse will expand to show individual hosts' status. The experimenter can click on a host to lock the display and then examine the charts on the right more closely. These charts display the recent history of the selected host's health condition, including detailed measurements of load, disk space, swap and paging activities. Together, the heatmap allows experimenters to quickly detect problems and the charts help them diagnose the problems, making this an effective diagnostic and monitoring tool for testbeds.

In any large, complex network, failures are inevitable. Generally, users report these errors to administrators, who then diagnose and correct the problems. LARIAT, at a minimum, must be able to reproduce the error reporting part of the process; that is, it should provide an experimenter with detailed status of the virtual users as they log their actions and results. Figure 8 shows how these logs are presented to the experimenter. Each traffic generator has events displayed in a timeline. The events are painted as vertical bars. The colors of the bars indicate the outcomes of the events: green for okay, yellow for warning, and orange for error. At the high level, the display shows when virtual users generate traffic and whether the traffic is okay. For example, the users on <http://www.waterbury.connecticut.com> are working approximately 8-hour shifts with 4-hour breaks in between. There are a lot of warnings and some errors toward the end of the each shift. The experimenter can zoom in to examine the events more closely. He could also break down the traffic events by applications. For example, traffic on <http://www.worcester.massachusetts.com> is generated from the *File Browser*, *Internet Explorer*, *Outlook*, *Words* among others. In this case, email seems to have a lot of failures, while web browser seems okay. To investigate the problems further, the experimenter can retrieve log messages for the selected events, as shown in the lower part of the display.

The primary use of these monitoring tools is to help experimenters verify that tests are running as expected (at least from the traffic generation perspective) and detect any system or configuration error. LARIAT does not monitor the system under test. Nevertheless, the outcomes of traffic events are directly affected by test events. Therefore, the results shown in these displays must be interpreted in the context of the experiment. For example, traffic failures may be the result of activating defensive tools that block critical pathways. While it is not a complete solution, these monitoring tools can provide valuable situation awareness during test.

4 Future Work

LARIAT and the Director have been used extensively over the last several years at dozens of sites. As with any interface design, the Director must balance expressiveness with ease of use. For example, a network is better represented as a graph

than a tree. However, a graph with thousands of nodes is hard to understand and even harder to manipulate. The simplicity of a tree is attractive, but there are network devices that cannot be easily classified into one network or another. The tree structure is also unsuitable for describing the link layer. We are considering a hybrid approach to provide the best of both worlds.

Over the last decade, we have seen IO testbeds grown significantly, but experimenters are not given any more time to configure their testbeds. Most experimenters already have tools for designing network topologies integrated into their workflow (e.g., Visio). It would save a lot of time if the Director could import data from these tools, so that the experimenters do not have to enter the network information again. The data entry problem can also be alleviated by auto-discovering the topology. However, it is unlikely that all of the information needed by LARIAT can be discovered without modifying existing tools. To this end, it would be useful to let experimenters set properties for multiple hosts simultaneously in a context sensitive environment, so that these missing pieces of information can be filled in quickly.

Furthermore, as testbeds grow, situational awareness becomes even more important. The Director visualizes both host and user-level events, but it is still missing other important aspects of traffic generation, such as the spheres of communication and the paths of traffic. For example, in addition to reporting that a user is browsing a file share, it would be informative to correlate it with other events, such as this user's past activities and other users' actions in this file share. Through correlation, the visualization can start painting a more complete picture of the networks in a test. The challenge is to scale any solution to hundreds of thousands of users, both in designing the visualization and in engineering the system.

5 Conclusions

In previous sections, we described the design and implementation of the Director. The Director dramatically reduces the time and effort required to describe and perform a test in LARIAT. This improvement is due in part to intelligent defaults and the ability of the Director to automatically create the virtual user environment. This allows experimenters to very quickly define their test network at a level they are comfortable with and the Director will populate the virtual user world on top of that topology. For experimenters that desire more fine-grained control, the Director supports editing of user and traffic models, as well as various parameters of instantiated users. As test networks expand to include hundreds of thousands of virtual users, the Director has employed many techniques to address scalability issues. For example, to avoid making too many simultaneous connections, the Director may request a few hosts to relay task orders to others if possible. Task results are posted to a centralized database, where the Director provides an at-a-glance status of the task, showing successes, warnings, and failures. The goal of this work was to help experimenters define and execute tests more accurately and quickly on LARIAT testbeds. We have achieved this goal by providing a unified, easy-to-use environment for controlling

testbeds at both the network level and the user level. The Director acts in concert with other LARIAT software to enable experimenters to create virtual components, configure networks and hosts, and monitor status and activities easily and reliably.

Although the capabilities of LARIAT are unique, the usability challenges are common among large testbed systems. We found Ben Shneiderman's Visual Information Seeking Mantra "overview first, zoom and filter, then details-on-demand" (Shneiderman, 1996) to be a reliable approach in dealing with structural complexity and information overload. We chose visual representation structures that were intuitive and scalable. The hierarchical testbed tree and the dynamic heatmap both handle aggregation well. They allow experimenters to drill down to a specific area without losing sight of the context. Elements are rendered using visual cues such as hue, intensity, and shapes to help experimenters make high-level assessments of the testbed and to direct their attentions to problems. The consistent application of the overview-to-details model, the selection of scalable representations, and the attention to visual features made the Director a successful interface for LARIAT and for IO testbeds. We hope our experience will provide insights for future development in testbed management and interface software.

Acknowledgements The authors wish to thank all contributors to the LARIAT project. We would also like to thank Chris Connelly, Sanjeev Gupta, Kyle Ingols, Shawn O'Shea, Reed Porada, Douglas Stetson, and Seth Webster.

References

- Albrecht, J., Tuttle, C., Snoeren, A.C., Vahdat, A.: PlanetLab Application Management Using Plush. *SIGOPS Oper. Syst. Rev.* **40**(1), 33–40 (2006)
- Bajaj, S., Breslau, L., Estrin, D., Fall, K., Floyd, S., Haldar, P., Handley, M., Helmy, A., Heidemann, J., Huang, P., Kumar, S., McCanne, S., Rejaie, R., Sharma, P., Varadhan, K., Xu, Y., Yu, H., Zappala, D.: Improving simulation for network research. Tech. Rep. 99-702b, University of Southern California (1999)
- Ball, R., Fink, G.A., North, C.: Home-centric visualization of network traffic for security administration. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 55–64. ACM Press, New York, NY, USA (2004)
- Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K., Ostrenga, R., Schwab, S.: Experience with DETER: A testbed for security research. Second IEEE Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom 2006), March (2006)
- Boothe-Rabek, J.C.: WinNTGen: Creation of a Windows NT 5.0+ Network Traffic Generator. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (2003)
- Eide, E., Stoller, L., Lepreau, J.: An Experimentation Workbench for Replayable Networking Research. *Proceedings NSDI*, Apr (2007)
- Fink, G.A., Muessig, P., North, C.: Visual correlation of host processes and network traffic. In: *VIZSEC '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, p. 2. IEEE Computer Society, Washington, DC, USA (2005)
- Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Preserving the big picture: Visual network traffic analysis with tn. In: *VIZSEC '05: Proceedings of the IEEE Workshops on*

- Visualization for Computer Security, p. 6. IEEE Computer Society, Washington, DC, USA (2005)
- Henry, N., Fekete, J.D.: Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics* **12**(5), 677–684 (2006)
- Kamara, S., Davis, D., Ballard, L., Caudy, R., Monrose, F.: An extensible platform for evaluating security protocols. In: ANSS '05: Proceedings of the 38th annual Symposium on Simulation, pp. 204–213. IEEE Computer Society, Washington, DC, USA (2005)
- Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A.: Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings **2**, (2000)
- Peterson, L., Bavier, A., Fiuczynski, M.E., Muir, S.: Experiences building planetlab. In: USENIX '06: Proceedings of the Seventh Conference on USENIX Symposium on Operating Systems Design and Implementation, pp. 25–25. USENIX Association, Berkeley, CA, USA (2006)
- Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., Yu, H.: Opendht: a public dht service and its uses. In: SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer communications, pp. 73–84. ACM Press, New York, NY, USA (2005)
- Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM Comput. Commun. Rev.* **27**(1), 31–41 (1997)
- Rossey, L.M., Cunningham, R.K., Fried, D.J., Rabek, J.C., Lippmann, R.P., Haines, J.W., Zissman, M.A.: LARIAT: Lincoln adaptable Real-Time Information Assurance Testbed. Aerospace Conference Proceedings, 2002. IEEE **6**, (2002)
- Schwab, S., Wilson, B., Ko, C., Hussain, A.: Seer: A security experimentation environment for deter. In: DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007, pp. 2–2. USENIX Association, Berkeley, CA, USA (2007)
- Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. *vol 00*, 336 (1996)
- White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, pp. 255–270. USENIX Association, Boston, MA (2002)

Visual Analysis of Network Flow Data with Timelines and Event Plots

D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd

Abstract This paper describes Isis, a system that uses progressive multiples of timelines and event plots to support the iterative investigation of intrusions by experienced analysts using network flow data. The visual representations have been designed to make temporal relationships apparent, allow visual classification of events with dynamic brushing, and enable users to organize their visualizations to reveal traffic structure and patterns by reordering rows. Isis combines visual affordances with SQL to provide a flexible tool for investigation. We present an annotated case study using anonymized data of a real intrusion that demonstrates the features of Isis.

1 Introduction

For the last 2 years we have been researching visual tools to aid network security investigations by working with the principal network analyst for the electrical engineering and computer science departments. Analysts use a variety of data sources to resolve incidents, including system logs, intrusion alerts, network flows, and full packet traces. Our tools target network flow data. They make use of a data source that produces 0.5–3 million flows per hour with daily accumulations in the tens of millions of events. Developing strategies for storage, retrieval, and display at these data volumes heavily influenced our design.

During incident handling, time is of the essence for analysts because they need to quickly isolate intrusions and because the sequencing of events in time is critical in reconstructing the methods and patterns of intrusion. As the analyst looks through time he frequently *pivots* to change the IP on which he is currently focused. To support this iterative search process, our tool, *Isis*, provides analysts with two linked visual representations of temporal sequences of network flow traffic – the *timeline*

D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd
Department of Computer Science, Stanford University

and the *event plot*. In both representations, IP addresses act as categorical values for the vertical axis with time on the horizontal axis. Timelines show an aggregate value over all events, while event plots reveal the patterns of individual events. These are examples of Bertin's reorderable matrices where rows are rearranged to reveal visual structure in the data through juxtaposition (Bertin, 1983).

The two representations support a common set of affordances to help the analyst look for temporal relationships in the data. The different levels of aggregation in the displays provide different opportunities to visually classify events with dynamic brushing and to create visual structures from the events. Small multiples of timelines are compact, which provide orientation, navigation and history at the expense of detail. In contrast, event plots provide enormous detail about the exact sequence of events and permit reordering of IP addresses to show their relationships. However, their unaggregated nature means they do not use space efficiently.

The rest of the paper is organized as follows. First, we describe the network flow data and how it has been structured in a database to permit the analyst to issue efficient queries. Next, we detail the evolution and design rationale of our system, *Isis*. We describe how an analyst interacts with our system in practice, by following an annotated case study of a typical intrusion incident. We conclude with a discussion of related work, and the benefits and limitations of our approach.

2 Network Flow Data

The data for our visualizations are the routed ICMP, UDP, and TCP network flows, captured by a sensor at the network gateway, and stored in a relational database. Each flow summarizes the time and duration of a network connection at the transport layer. Packet counts and bytes are stored, but the contents are not. To be useful for the analysis of network incidents, flows must be organized for fast searches over the tens of millions of daily flows. We use a MySQL database to store the flows, which provides the analyst with a flexible and familiar interface for specifying queries. Initially we directly mapped flow record fields to table columns, but this resulted in unsatisfactory performance. More importantly, it did not match up well with the analyst's needs so we modified the schema as described below. Because flows are relatively modest in size compared to the traffic they summarize, both the flow sensor and the database repositories can be housed on inexpensive commodity hardware.

2.1 Flow Sensor

Flow records can be uni-directional or bi-directional. Bi-directional flows collapse the two uni-directional flows of a conversation into one record, with separate fields for the port, packet and byte counts. We are using the open source Argus flow system (<http://www.qosient.com/argus>) configured to create bi-directional flows for routed

ICMP, UDP, and TCP traffic. Each flow is defined by the 5-tuple key of protocol, source/destination IP, source/destination port. The orientation of a flow is determined by the srcIP in the packet that created the flow. For long-running connections, a flow record is generated every 60 s, but any connection which is idle for more than 300 s will be dropped. It is re-established as a new flow if subsequent packets are seen.

2.2 Database Repository

Many incidents begin with a report of anomalous behavior involving a local IP address. As a result, the analyst will issue a query for all the traffic associated with that focus IP. Once the analyst locates a suspect IP that contacted the focus, he will often want to retrieve the suspect IP's communication with other local IPs. We call this a *pivot* from the focus IP. The problem is that a SQL table constructed with columns directly mapping raw flow record fields will have srcIP and dstIP columns. Obtaining all the traffic for a single IP would require queries with expensive OR or UNION clauses, because the investigation process is easier to reason about in terms of local and remote addresses.

To improve query performance, we transform all the src/dst fields in flow records into local and remote columns in the database table. To preserve the critical orientation information of the src/dst relationship, we add a column which specifies the role played by the local IP in each flow. Because we capture some local traffic, there can be flows with two local addresses. In this case we arbitrarily choose the destination as the local IP. Since using local and remote designations makes a query about a flow's destination port more complex, we include it as a convenience column.

The database schema also incorporates metadata reflecting aspects of the structure of the network. Our local network is subdivided along lines reflecting the administrative and technical groups. Since analysts are responsible for these logical subnets, called VLANs (for virtual LANS), the database allows queries to be restricted by VLAN. A similar segmentation is made for remote IPs because the Internet is divided into autonomous system numbers (ASNs) which define responsibility for IP address ranges. Since an ASN is roughly equivalent to an internet service provider (ISP) and resolving network incidents is usually done at the ISP level, the database also associates ASNs with an IP address.

3 The Investigation Process

Network security incidents can be triggered in a variety of ways, by an automatic alert generated by intrusion detection systems (IDS), by an e-mail complaint from an administrator at a remote network, or even by a user noticing that a machine has started behaving oddly.

Once a report has been received, analysts assume a variety of roles (D'Amico et al., 2005) while engaging in an iterative process of hypothesis generation and evaluation: *Triage* to decide whether a report merits investigation; *Escalation* to determine method of compromise and its extent; *Correlation* to compare this incident with those of the past; *Threat analysis* to search for attacker identity and motivation; *Incident response* to recommend or implement a course of action; and *Forensic analysis* to gather and preserve evidence.

The network flows collected by the sensor show details of the times and extent of communications among machines and so are valuable for triage and escalation. If sufficiently fast access to historical data is available, they may also be used for correlation analysis. During an investigation, analysts are trying to identify flows that comprise an intrusion out of a vastly larger set of flows. By providing filtering, sorting, and compact visualization of the flows, Isis can help the analyst to build a mental model of the network activity so that he can distinguish intrusion flows from normal flows.

As one might expect, we observed that an analyst typically begins an investigation focused on the IP thought to be compromised. The analyst would inspect all of its traffic looking for sources of possibly malicious traffic. The analyst would then pivot to focus on the suspect IPs and inspect their traffic. If that traffic indicated additional possible compromises, he would pivot again. This iterative analysis process may be described as follows:

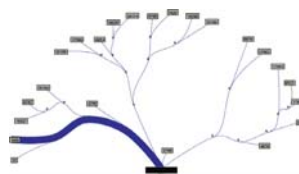
1. Select a IP to focus on and look at its traffic by querying the database.
2. Inspect the composition of the traffic to determine it is related to the intrusion.
3. Compare traffic to other flows to discover attributes and temporal correlations.
4. If there is uninteresting data, refine the query by adding filters, and repeat #2.
5. If an interesting IP is found, pivot to it, changing focus, and repeat #1.

4 Flow Maps

Initially we tried to use flow maps, a type of node-link diagram (Phan et al., 2005), to display network traffic. Each computer was mapped to a node and the width of an edge encoded the amount of traffic between the two computers. Flow maps were used to minimize the amount of edge clutter that would occur had we created an edge for each flow. However, we found that it was difficult to pivot with node-link diagrams. From an initial graph, the analyst could pivot on a partner IP node, and the system would add new nodes and edges to the pivot node. Unfortunately, repeated pivots would add too many nodes and the display would become too cluttered.

Analysts also found the intermediate nodes used to cluster outgoing edges in flow maps to be meaningless because the clustering was based on position of the destination node, and not on an attribute of the data. We tried data driven layouts, such as mapping a node's position to the geographic location to its ASN, this was not an effective use of screen space. We also tried creating meaningful intermediate nodes that would break traffic into different components, such as port nodes.

Fig. 1 A flow map where nodes are radially mapped by the first time they contacted a focus node at the center



Unfortunately, this had the disadvantage of replicating the same IP in multiple nodes and forced the analyst to visually collect traffic from disparate locations.

After using the flow map prototype with analysts on several intrusions, we realized that it was critical to make the temporal relationships of flows apparent. For example, analysts wanted to look for correlations between a login and the beginning of malicious traffic. The problem was that an edge only indicated that two IPs had communicated, but not when they communicated. Unfortunately, it was difficult to add temporal cues to node-link diagrams. Animation was largely ineffective as analysts had a hard time tracking movement and recalling trajectories (Tversky et al., 2002). We had some success using an analog clock metaphor seen in Fig. 1. Nodes were positioned by the first time that they communicated with a focus node at the center, but we later realized this layout was not flexible enough.

5 Progressive Multiples of Timelines and Event Plots

After our experiences with the flow map prototype, we decided that our visualizations should:

- Make temporal relationships apparent
- Allow users to classify events by their attributes and inspect event details
- Support an iterative analysis process
- Allow users to create visual structure in order to reveal traffic patterns

We developed a display where time is mapped left-to-right along the x -axis. Since the principal element of an investigation is the traffic to and from an IP address, we segmented the y -axis into rows, where each query for the traffic of a focus IP is visualized as a timeline. An example can be seen in Fig. 2. The analyst can brush the timeline to inspect its data attributes or get details on demand. If they find suspicious connections, they can pivot on those connections to create new timelines. New rows are added sequentially to the display, which preserves the steps taken by the investigator. This history allows the analyst to backtrack and revisit earlier queries. Analysts can also reorder rows to compare the results of different queries. We describe this technique as progressive multiples (Phan et al., 2006), a combination of progressive disclosure and small multiples (Tufte, 1990) of timelines.

Timelines are useful for overviews of network traffic, but when the investigation has narrowed to a small time window, the analyst needs to see unaggregated flows. Initially we provided the flow data as a sortable table, but analysts found it hard to

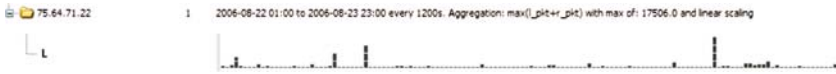


Fig. 2 Timeline using an aggregation expression that shows the maximum total packets to and from the local IP that is suspected to be compromised

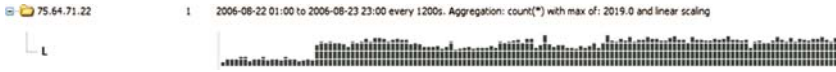


Fig. 3 Timeline using an aggregation expression that shows the total number of connections to and from the local IP that is suspected to be compromised

extract temporal patterns from text. In response, we developed the event plot. An example can be seen in Fig. 9. For each IP on the y-axis, each of its flows is shown as a mark on the x-axis. The event plot has a sidebar to support inspection of traffic in several dimensions: port, ASN, VLAN, locality, and role. Brushing an IP in the main window highlights entries in the sidebar that appear in that IP’s row. Likewise, brushing an entry in the sidebar highlights IPs that contain that value. The event plot allows the analyst to color marks by traffic type, to reorder the rows to reveal traffic patterns, and to adjust the horizontal spacing of the marks.

We describe the features of timelines and event plots by presenting them in the context of the investigation of an anonymized version of a real intrusion.

6 A Case of Mysterious IRC Traffic

After looking at a routine summary of network activity, an analyst noticed that there was a high level of IRC traffic to a server in northern Europe from a local machine, 75.64.71.22. Since hackers often use an IRC channel to control a bot on a compromised host, she decided to look at the machine’s traffic by providing the tool with a focus IP, time window, aggregation function, and filter string.

The aggregation and filter are specified as SQL expressions. To ensure the entire query result fits on the screen events are binned and shown as bars. The height of the bars is controlled by the aggregation expression, which can be any expression that returns a non-negative scalar value, such as the min, max, or average of the number of packets. Figure 2 shows the result of a query of 75.64.71.22 with the aggregation: $\text{max}(\text{l_pkt} + \text{r_pkt})$ which maps the height of each bar to the flow in that bin with the largest total number of packets.

The aggregation expression count^* will count the rows satisfying a query and creates a timeline where the bar heights are proportional to the number of connections in each bin. In network incidents, the existence of a connection is often as important as its size or duration so it is a common for an analyst to use this aggregation to begin exploring traffic. Figure 3 shows the result of a query of using count^* to get an overview of the traffic spanning the last day.

To discover the extent of the IRC traffic, the analyst looks at the port distribution by asking for the tearoff window seen in Fig. 4. After sorting the ports by the aggregate value, the analyst sees that port 6667, an IRC port, has approximately 63,000 flows. The analyst can see how activity on the port is distributed in the timeline by brushing its cell. If they see a value of interest, they can create new timelines that filter traffic on these ports.

Brushing the cell containing port 6667 results in the timeline seen in Fig. 5. Orange highlighting is overlaid on all of the existing timelines. The height indicates the proportion of traffic using port 6667. Orange marks underneath the bins indicate the presence of port 6667 traffic, even if the height of the orange may not be visible in the bar. As a result, the analyst can see that IRC traffic has been present over the whole time period.

The analyst can bring up tearoff menus for several dimensions: partner IPs, ports, role, ASN, locality, or Vlan. From the IP tearoffs the analyst can also pivot to new IPs. Figure 6 shows a brush from the same tearoff menu for port 6666, which is another IRC port. This reveals a different connection to an IRC server that is correlated with the spike in activity. Since the goal is to determine the first incidence of IRC traffic, the analyst makes a mental note to come back and look at the traffic on port 6666 at a later time.

Figure 7 shows a progression of queries the analyst generates to discover the beginning of the IRC traffic. To determine when the IRC traffic began, she limits the

Fig. 4 A tearoff menu that shows the distribution of traffic by destination port

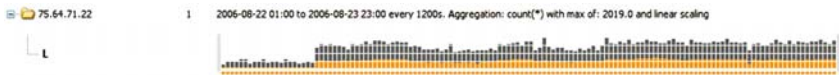
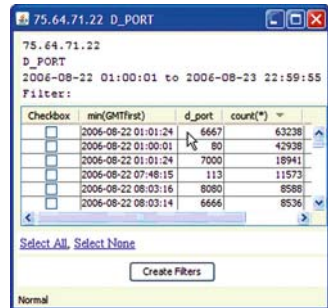


Fig. 5 A brushed timeline where the orange highlighting indicates the presence of connections to IRC servers on port 6667 that extends over the whole period



Fig. 6 A brushed timeline where the orange highlighting indicates connections to IRC servers on a different port, 6666

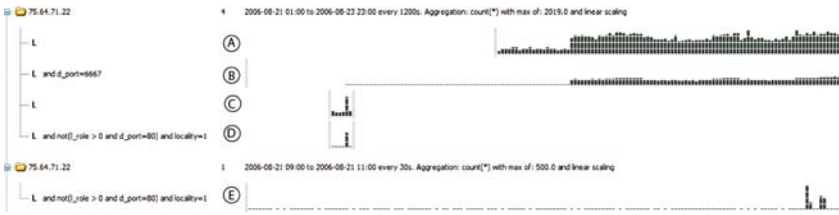


Fig. 7 A progression of queries generated in the timeline display showing how the user has narrowed the time window from a 2-day period to 2-h period

<i>l</i> _weekday	<i>l</i> _hour	GHfirst	duration	locality	<i>l</i> _role	proto	<i>l</i> _asn	<i>l</i> _vln	inet_rstoa(<i>l</i> _pn)	<i>l</i> _port	<i>r</i> _asn	<i>r</i> _vln	inet_rstoa(<i>r</i> _pn)	<i>r</i> _port	<i>d</i> _port	<i>l</i> _pkt	<i>l</i> _byte	<i>l</i> _lbyte	<i>r</i> _pkt	<i>r</i> _byte	<i>r</i> _lbyte
2	2	2006-08-21 09:00:02	0.002	1	-3	1	32	71.75.64.71.22	8	26101	24288.64.94.230.32	8	0	2	196	128	2	196	128		
2	2	2006-08-21 09:00:13	5.568	1	3	6	32	71.75.64.71.22	25	750	20280.121.130.70.75	8	0	2	196	128	26	8620	7814		
2	2	2006-08-21 09:00:27	0.002	1	-3	1	32	71.75.64.71.22	8	26101	24288.64.94.230.32	8	0	2	196	128	2	196	128		
2	2	2006-08-21 09:00:45	7.4	1	3	6	32	71.75.64.71.22	25	750	63744.212.249.195.196	25	25	72	5002	1090	138	91904	84516		

Fig. 8 An example of the sortable drill down table that is generated when the analyst looks at the raw data

traffic to port 6667 and changes the time window to be another day earlier, which produces row B, and shows the beginning of the IRC traffic. To look for the source of the compromise, she queries for all traffic in a 2-h window centered on the beginning of the IRC traffic, resulting in row C. Since the focus IP is a web server, the analyst removes all web traffic served by the focus and restricts the traffic to be non-local, as local traffic is usually benign. This is done using the filter “not(*l*_role>0 and *d*_port=80) and locality=1”, resulting in row D.

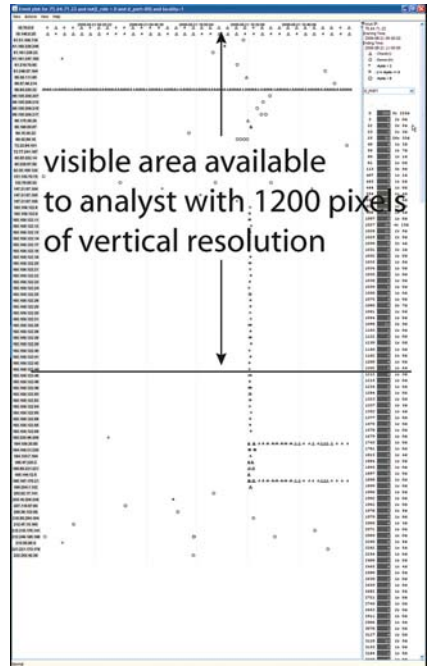
Timelines in the same folder share a common bin size and aggregate, which makes it easy to juxtapose rows and compare query results. Since the bin size of the larger folder is constrained by the earlier queries, the analyst moves row D to new folder on row E. The bin sizes are recalculated, and changes the binning to 30 s. Each folder has different bin sizes and aggregates so that the system can display a wider dynamic range of data.

Now that the analyst has narrowed her focus to a 2-h window, she wants to see the individual flows. The analyst can create a drill down table to look at the data, which has approximately ~1,600 flows, as seen in Fig. 8. Each row in the drill-down table describes a flow between the focus IP and partner IP.

Since the analyst is looking for temporal patterns, she moves from the table view to the event plot seen in Fig. 9 which provides a direct mapping from table rows to marks. Marks are placed along the *x*-axis according to a table row’s timestamp and the event plot is initially ordered by numeric IP address. To facilitate visual classification and inspection of traffic, the system provides several affordances that allow the user to modify the size, shape, and color of the mark based on its flow attributes. To obtain additional detail about a mark, the user can hover over the mark, and a tooltip will display the full information for that flow.

Each mark is mapped to one of three bin sizes (first quartile, middle quartiles, last quartile) depending upon the total number of packets exchanged between the two IP

Fig. 9 An event plot generated of a two-hour time window containing ~1,600 events. Notice that many events lie off-screen, due to limited resolution



addresses. From the sidebar, the analyst can adjust the thresholds for the mark size to reduce visual clutter or highlight outliers.

The shape of the mark initially shows the role of the focus IP. If the connection was initiated by the focus IP then it is the client and the mark is a triangle. Otherwise, the focus IP is a server and the mark is a circle. Being able to see when a computer switches from being a client to being a server is often a useful heuristic for localizing intrusions. Since too many encodings can interfere with preattentive search, the analyst can choose to view all marks as squares. To isolate the initial intrusion, she looks for an SSH connection that occurs before the start of traffic to the IRC servers. She can do this by using the sidebar, which provides a breakdown of the traffic into its constituent parts by the same dimensions as the tearoffs in the timeline display: ASN, Port, Vlan, Locality, and Role. She can control which dimension to inspect with the drop down menu.

Hovering over port 22 (SSH) in the sidebar causes the system to highlight all the rows in the event plot that contain flows with that destination port, revealing SSH server connections from two IPs as seen in Fig. 10.

The system allows users to brush marks as well as the entries in the sidebar. Hovering over a mark highlights the values in the sidebar that appear in the IP's connections and highlights bins in any timeline containing that IP. For instance, if an IP uses both port 22 and port 80, Isis will highlight those ports when the mouse is over that row. Hovering over an entry in the sidebar will highlight all the rows that contain that value and the bins in the timeline display that contain that value.

Fig. 10 Brushing the sidebar on port 22 highlights two rows containing suspicious SSH connections

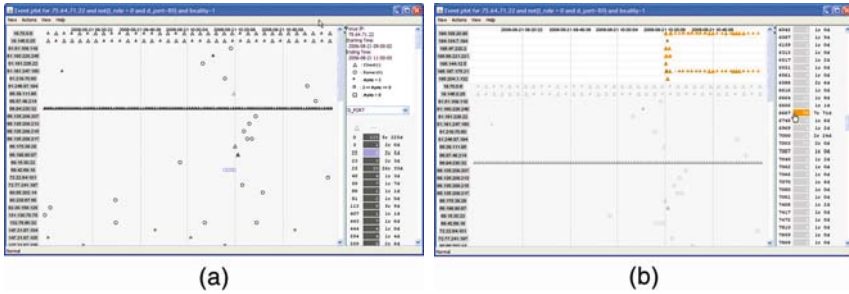
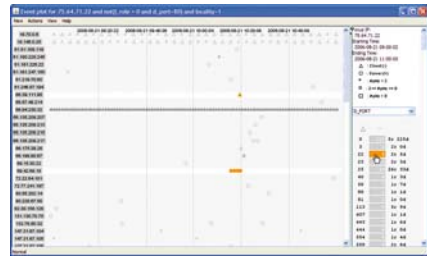


Fig. 11 (a) The left event plot shows the results of coloring port 22 a shade of blue. (b) The right event plot shows how the analyst has brushed port 6667 and used the “collect a brush” action to bring the IPs with flows on port 6667 to the top of the display so they can be made visible

To keep permanent track of port 22, the analyst colors its sidebar entry a shade of blue, seen in Fig. 11a. The system allows the user to color marks by any of the sidebar dimensions. The color space for each dimension is global, so marks in other event plots with port 22 will now also be colored blue. A mark is only colored by one dimension at a time to avoid conflicts in displaying a mark that has had multiple attributes colored.

To look for the IRC traffic, the analyst brushes over port 6667, but does not see any highlighting. The problem is that the number of rows that need to be displayed can exceed the physical limitations of the screen. Note that Fig. 9 is 1,920 pixels high, but currently she has about 1,200 pixels, or 60% of that vertical resolution. When a user brushes a dimension, off-screen rows are logically highlighted but are not visible. To address this, the system allows users hovering over a dimension to “collect a brush”, that is to force all brushed rows to move to the top of display. This also allows her to easily group IP addresses by their different attributes in order to organize the display visually. In Fig. 11b, we see her collect the brushed rows for IRC port 6667.

At this point the analyst has a good idea of the intrusion sequence, but the display does not yet reflect this structure. After coloring the marks for port 6667 in red, she now searches for the prior download of the IRC tools themselves. The analyst knows that once an intruder logs in via SSH, he will usually download a set of exploit tools via a web server or ftp. The analyst selects web traffic on port 80 in the sidebar and colors it yellow. The system allows her to reorder rows to create visual structure out of the events. Figure 12 shows the results of the analyst moving the rows containing

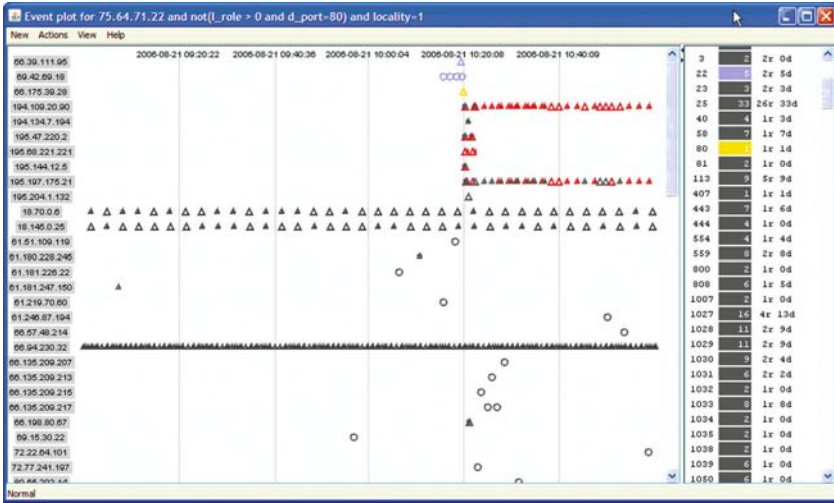


Fig. 12 An event plot where the analyst has colored SSH connections on port 22 in blue, web traffic on port 80 in yellow, and IRC traffic on port 6667 in red. She has also reordered the rows so that the flow of events can be more easily read in a left-to-right and top-to-bottom manner

SSH traffic (port 22) and Web traffic (port 80) so that the image can be read top-to-bottom, left-to-right. The juxtaposition allows her to compare the behavior of different IP addresses or group IP addresses by their role in an intrusion.

Up to this point the analyst has been working with the event plot scaled with a traditional continuous time axis. The advantage of continuous time is that the horizontal distance between marks is linearly related to the number of seconds between those flows. This allows her to get a sense of how events are clustered or distributed in time. Since intrusion events often occur in a small time interval, a continuous view of time can result in overplotting and reduced visibility.

To reduce the effects of overplotting, the system can show events in an ordinal space. This distributes them evenly across the *x*-axis such that the distance between the marks does not encode the number of seconds between events. However, because understanding the ordering of events is often more important than knowing the precise time at which they occurred, the ordinal view can be very useful. To make more space for the events, the analyst switches the view to be ordinal, as seen in Fig. 13. This reveals a gap between the SSH connections and the onset of IRC connections suggesting that there may be interesting events in the off-screen flows.

Scrolling down, the analyst discovers that the spacing is due to the presence of proxy scans from computers presumably controlled by the intruder which are mapping the compromised computer. Figure 14 shows an ordinal event plot where the rows have been reordered by the analyst to show the proxy scans.

Now that the analyst has established a sequence of events for the intrusion, she wants to precisely adjust the spacing of the marks so as to create a narrative. The system allows her to interactively edit and adjust the location of the temporal grid-lines to segment the event plot and make it easier to read visually. This allows her to

Fig. 13 A plot with the same data as Fig. 12, changed from continuous to ordinal time so as to evenly space marks across the x-axis

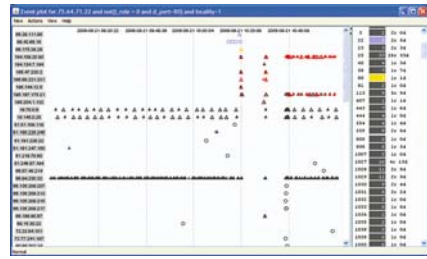
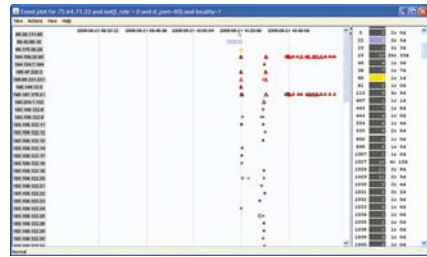


Fig. 14 An ordinal event plot where the rows have been reordered to show a string of proxy scans from machines presumably controlled by the intruder



create multiple regions with different horizontal distortion, similar to the technique offered by table lens (Rao and Card, 1994).

The analyst completes the narrative in Fig. 15 by adjusting gridlines to allocate more space to the events that make up the intrusion. The SSH connection in blue circles indicates a remote login from 69.42.69.18. The analyst concludes that it is the IP that is likely to be responsible for the initial intrusion and that IRC bot tools were downloaded from 66.175.39.28 by the flow in yellow. The remaining connections in red are the client connections to the IRC servers. The gray connections are the proxy scans. At this point, she will want to return to the timeline display to pivot upon the intruder IP to determine if other computers have been affected and begin this process a new.

This case study has demonstrated how an analyst is able to investigate intrusions using a combination of small multiples of timelines and event plots. Both representations make temporal relationships apparent; make it easy to visually classify events; and to create visual structure. The two displays are complementary: timelines provide overviews, navigation through pivoting, and support iteration, whereas event plots allow the analyst to classify events and create visual structure through the reordering of rows.

7 Related Work

There are many examples of applying visualization to improve monitoring and situational awareness of a network. Lakkaraju et al. describe a tool called NVisionIP, which visualizes flows using three levels of granularity: a galaxy view, which shows the whole network, a small multiples view which shows the information

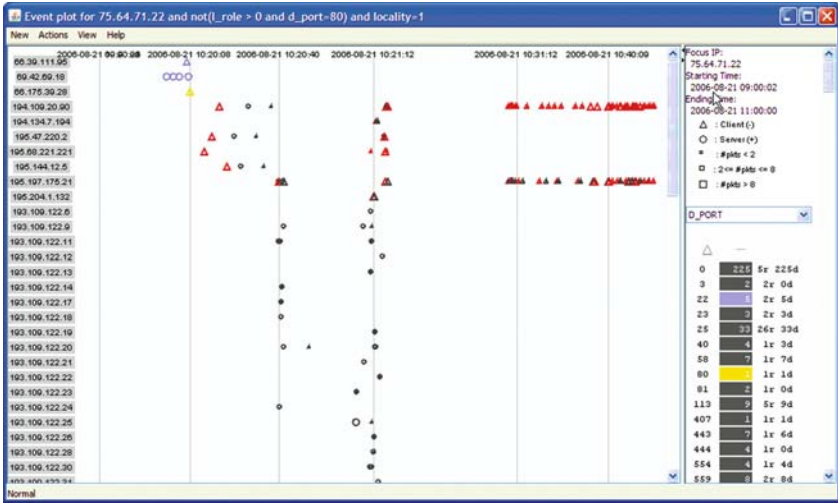


Fig. 15 The narrative event plot of the intrusion after adjusting the ordinal gridlines. The second row shows the blue server flows of the SSH compromise. The next row shows the large yellow triangle of the IRC bot download. Finally there are several rows of client IRC connections as red triangles. The remaining traffic in gray are proxy scans

for a selected set of hosts, and a view which shows the behavior of one machine (Lakkaraju et al., 2004). Similarly, Yin et al. describe VisFlowConnect, which uses parallel coordinates to monitor the state of a network (Yin et al., 2004).

To represent time, some systems have used animation (Lakkaraju et al., 2004; Musa and Parish, 2007) instead of allocating an axis. Among those with spatial layouts of time, PortVis (McPherson et al., 2004) visualizes port activity with time on the y-axis and IDS Rainstorm (Abdullah et al., 2005) which visualizes alarms from an IDS for a large IP space in columns where the x-axis of each column is time. Livnat et al. describe polar layouts with time on the radius (Livnat et al., 2005) while Radial Traffic Analyzer (Keim et al., 2006) uses an angular measure for time.

Isis differs from the preceding visualizations in that its focus is incident investigation using network flows. VIAssist (D’Amico et al., 2007) is a framework also designed for escalation and correlation analysis. It integrates multiple data sources, visualizations, and supports collaboration among analysts. The timelines and event plots of Isis are not found in VIAssist, but could be incorporated into such a framework.

Among temporal event plots with categorical values on the vertical axis are the Gantt and PERT charts used in job shop and project scheduling. These differ in that events are the categorical variables whereas in Isis, events are the marks in our plots. This makes the event plot similar to the medical event chart (Lee et al., 2000) used to track disease progression and treatment. In the area of network security, Goodall et al. describe TNV which maps each IP address to a row to produce a timeline of activity (Goodall et al., 2005). Connections between IP addresses are drawn as lines among rows, in contrast to Isis, which maps an aggregate over connections for a

specific IP address as a bar in a bin on a single timeline, or maps connections into marks along the x -axis of an event plot.

8 Future Work and Conclusions

The size of the datasets being visualized is constantly growing; for example, the traffic at the university's border is an order of magnitude larger than our data source. Given the volume of data, the visualization that runs on the analyst's workstation should offload calculation to a remote database. We have been able to conduct analyses spanning a few days, using queries that run over tens of millions of events and select tens of thousands for display. These queries take under a minute to complete. If an analyst needed to search a few billion rows, we have some preliminary evidence that this could be done by distributing a query across a clustered database server. However, even if the query eliminated 99.99% of the rows, it might still generate a result set of several hundred thousand events and a degree of overplotting that could easily overwhelm event plots with ordinal time. Incorporating methods that abstract flows into larger behaviors by Kannan (2006), Xiao et al. (2006), or Karagiannis et al. (2005) holds promise for reducing clutter as well as adding useful information for the analyst.

The timeline and event plot visualizations were developed for experienced analysts with the goal of revealing temporal patterns through juxtaposition. The two representations are designed with Shneiderman's information-seeking mantra (Shneiderman, 1996) in mind. Timelines provide overview, zooming, and filtering functions while event plots illuminate details. Their development benefited enormously from taking an evolutionary approach. For example, the initial decision to allow the analyst to directly alter the WHERE clause of the SQL query was thought to be a temporary scaffolding to enable rapid prototyping of interface elements. However, we soon realized that selectively exposing SQL in parts of the interface provided flexibility for expert users.

Designing tools for expert users that leverage their domain knowledge remains a difficult problem. Practices that work well for novices or for tools used occasionally can be inefficient and even annoying for those who must use them every day. The ad hoc nature of incident analysis suggests we should design each tool to do one thing well, knowing that the analyst will combine it with others. Network incident response is still more art than science and likely to remain that way for some time to come.

References

- Abdullah, K., C. P. Lee, et al. (2005). "IDS RainStorm: visualizing IDS alarms." IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05): 1–10.
- Bertin, J. (1983). *Semiology of Graphics: Diagrams, Networks, Maps*, University of Wisconsin Press.

- D'Amico, A., K. Whitley, et al. (2005). Achieving Cyber Defense Situational Awareness: A Cognitive Task Analysis of Information Assurance Analysts. Human Factors and Ergonomics Soc. 49th Ann. Meeting.
- D'Amico, A., J. R. Goodall, et al. (2007). "Visual discovery in computer network defense." IEEE Computer Graphics and Applications.
- Goodall, J. R., W. G. Lutters, et al. (2005). "Preserving the big picture: visual network traffic analysis with TNV." IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05).
<http://www.qosient.com/argus> Argus.
- Kannan, J., J. Jung, et al. (2006). "Semi-automated discovery of application session structure." Proceedings of the Sixth ACM SIGCOMM on Internet Measurement.
- Karagiannis, T., K. Papagiannaki, et al. (2005). "BLINC: multilevel traffic classification in the dark." ACM SIGCOMM Computer Communication Review **35**(4): 229–240.
- Keim, D. A., F. Mansmann, et al. (2006). "Monitoring Network Traffic with Radial Traffic Analyzer." IEEE Symposium on Visual Analytics Science and Technology.
- Lakkaraju, K., W. Yurcik, et al. (2004). "NVisionIP: netflow visualizations of system state for security situational awareness." Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security: 65–72.
- Lee, J. J., K. R. Hess, et al. (2000). "Extensions and applications of event charts." The American Statistician **54**(1).
- Livnat, Y., J. Agutter, et al. (2005). "A visualization paradigm for network intrusion detection." IEEE Workshop on Information Assurance and Security.
- McPherson, J., K. L. Ma, et al. (2004). "PortVis: a tool for port-based detection of security events." Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security: 73–81.
- Musa, S. and D. J. Parish (2007). "Visualising communication network security attacks." Information Visualization, 2007. IV'07. 11th International Conference: 726–733.
- Phan, D., L. Xiao, et al. (2005). "Flow map layout." IEEE Symposium on Information Visualization (INFOVIS): 219–224.
- Phan, D., A. Paepcke, et al. (2006). "Progressive multiples for communication-minded visualization." Graphics Interface.
- Rao, R. and S. K. Card (1994). "The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.
- Shneiderman, B. (1996). "The eyes have it: a task by data type taxonomy for information visualizations." IEEE Symposium on Visual Languages: 336–343.
- Tufte, E. R. (1990). *Envisioning Information*, Graphics Press.
- Tversky, B., J. B. Morrison, et al. (2002). "Animation: can it facilitate." International Journal of Human-Computer Studies **57**(4): 247–262.
- Xiao, L., J. Gerth, et al. (2006). "Enhancing visual analysis of network traffic using a knowledge representation." IEEE Symposium on Visual Analytics Science and Technology: 107–114.
- Yin, X., W. Yurcik, et al. (2004). "VisFlowConnect: netflow visualizations of link relationships for security situational awareness." Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security: 26–34.

NetBytes Viewer: An Entity-Based NetFlow Visualization Utility for Identifying Intrusive Behavior

T. Taylor, S. Brooks, and J. McHugh

Abstract NetBytes Host Viewer is an interactive visualization tool designed to show the historical network flow data per port of an individual host machine or subnet on a network over time, using a 3D impulse graph plot. Such visualizations allow network administrators to quickly and effectively diagnose infected or malfunctioning computers by viewing data transmission patterns for each port on the entity. NetBytes has a set of interactive features which help to deal with the problems associated with displaying a 3D graph on a 2D screen. First, NetBytes offers a “selector” mode which allows the user to highlight specific ports (or times) on the graph using a slider and snap buttons. From the selector, the user can launch a set of 2D graphs (Bytes vs. Time and Bytes vs. Ports) to acquire more detailed information about the host with less clutter. Lastly, the user is able to rotate the 3D graph in any direction to mitigate occlusion. The long term objectives of this work include the integration of the NetBytes Viewer with complementary visualizations of the overall network. This application will integrate with a larger network analysis tool and be utilized as a drill-down mechanism.

1 Introduction

Over the years, the Internet has increasingly become host to Trojans, scanners and other viruses which can wreak havoc on private networks. These hostile programs have made it necessary for security administrators to watch over their networks using intrusion detection systems which trace large amounts of packet data. This data must be analyzed for signs of viruses. To help in this analysis, researchers have turned to visualization techniques. Visualizing network data allows network administrators to more quickly discover patterns of hostile activity and diagnose problem

T. Taylor, S. Brooks, and J. McHugh
Dalhousie University, Halifax NS, e-mail: teryl@cs.dal.ca, sbrooks@cs.dal.ca, mchugh@cs.dal.ca

computers. However, current techniques focus on visualizing network traffic as a whole and do not focus on visualizing historical data of individual hosts.

The NetBytes Viewer is an interactive visualization tool designed to show the activity of a network entity over time using a 3D impulse plot. The entity could be a single host (e.g. an email server) and the visualization shows the per port outbound NetFlow volumes for a certain time interval (e.g. hourly) over a specified time period (e.g. a week). The entity could alternatively be a subnet, showing per host behavior or a larger network aggregation showing per subnet behavior. Such a visualization allows network administrators to quickly and effectively diagnose infected or malfunctioning computers by viewing data transmission patterns for each port on the host. For example, an email server would typically have a significant traffic load on port 25 (SMTP) and port 110 (POP). A visualization using NetBytes would show this as “picket fences” corresponding to the ports with the height of each picket (impulse) representing an hourly volume. NetBytes allows the user to monitor a range of ports for significant network traffic. Traffic on ports that should be unused may be a sign of a worm, virus, or other malware. Unusually high volumes on supported services may also indicate a compromise. Using a historical database, NetBytes can help pinpoint the time at which a host was compromised. Tracing back through the original NetFlow data for this time allows the user to determine the source of the compromise. Working forward from that time can identify secondary infections.

This paper will discuss the design and implementation of the NetBytes Viewer and discuss how it can be an effective tool for analyzing network traffic. Furthermore, the document will outline some of the other network visualization tools currently under research as well as discussing the future plans for NetBytes.

2 Related Work

There exist several network traffic visualization tools that enable network administrators to monitor traffic flows on their internal networks. VISUAL (Ball et al., 2004) is a 2D visualization application which shows the traffic for a small to medium-sized network. All internal hosts are represented as cells in a square grid in the center of the screen. External hosts are represented as squares placed outside the internal grid. Square-size denotes the amount of IP activity of the host and lines in the square are used to denote port traffic. Line connections between internal and external hosts denote traffic flow while grid color denotes communication between computers internally. VISUAL allows interactive filtering of specific computers to reduce screen clutter and utilizes animation to show changes in network flow data over time. Detailed information of individual hosts is available in “text” form and includes host IP, IP addresses of all computers that are communicating, ports used and percentage of overall traffic. However, VISUAL is limited in its ability to show individual port traffic for a specific host over time.

VisFlowConnect (Yin et al., 2004) utilizes a set of parallel axes view on a 2D screen. Each point on an axis represents an IP address (of a machine or domain) and connections between points on parallel axes represent network connections and flow of data. Users can drill down in an overall global network view to see a subset of IP addresses in a specific domain or a view of the internal network. VisFlowConnect depicts time by using animation to show connections appearing and disappearing between computers. Furthermore, it has the ability to filter by port, protocol, transfer rate, and packet size. Lastly, VisFlowConnect does show individual host statistics in a text view. Stats include: total number of bytes into and out of the machine in the current time window, as well as, bytes transferred per machine.

NVisionIP (Lakkaraju et al., 2004) presents a visual representation of an entire class-B IP network on a single screen. The overview screen has horizontal and vertical axes in which all subnets of a network are listed along the top axis while the hosts in each subnet are listed on the vertical axis. Each host is colored based on some characteristic of interest which include traffic volume, number of flows or flows on a particular port. Once again animation is used to show traffic collected over a given time period. Users can select a region of the overview screen to launch another window which provides more detailed information about hosts in the selected region. Each host is represented by two bar charts. One chart displays the traffic on a number of well known ports with the other shows traffic on all other ports. Color is assigned to traffic on different ports to make it easier to compare flows of interest. NVisionIP also has a machine view which is launched by clicking on a single host. This launches a window which visualizes byte and flow count for all protocols and ports using 2D bar graphs. Although the drill down mechanism allows users to view data on individual hosts, it does not visualize patterns in the data over time.

The “Spinning Cube of Potential Doom” (Lau, 2004) is another network visualization tool which was built on top of the Bro intrusion detection system. It displays a 3D cube which a user can interactively spin. “Each axis represents a different component of a TCP connection: X is the local IP address space; Z is the global IP addresses space; and Y is the port numbers used in connections to locate services and coordinate communication” (Lau, 2004). TCP connections are displayed as individual dots with color used to distinguish a successful from an unsuccessful connection. Time is again displayed through the use of animation. This application has no drill down mechanisms for showing detailed data for individual hosts.

None of the above applications give detailed information on individual hosts. NetBytes complements these applications by showing detailed time and port information for individual host all on a single screen. It would therefore be an excellent addition to any of these applications.

Portall (Fink et al., 2005) is a network visualization tool which displays TCP connections between hosts and is able to correlate those connections with the processes that generate them. The main application window shows two parallel axes. One axis represents client hosts and processes of the machines that are monitored on the network while the other axis represents server hosts and processes. Client-server connections are displayed using straight lines from one axis to another while transmission volumes are visualized by tool tip popups. Time is represented using

animation with processes and connections appearing and disappearing. Portall is the first visualization tool to try to correlate traffic flow to the processes sending and receiving the traffic. It works well for very small networks but suffers from clutter and occlusion issues as the number of hosts and processes grow on the main display. Having said this, Portall and NetBytes could be good complements to one another as NetBytes can show traffic volumes and history for a host on a single screen while Portall could show from which processes that data is coming from.

Portvis (McPherson et al., 2004) is another visualization tool which incorporates three different displays to view TCP port traffic. The first visualization corresponds to a 2D grid where rows along the vertical axis represent units of time while each column along the horizontal axis represents a range of 2,048 ports. Color in each column is used to represent the level of activity on the ports during a particular time unit. A selector is used to select the unit of time to be displayed on the main visualization. The main visualization contains a 256×256 grid where each point represents one of the 65,536 ports. The location of the port on the grid is determined by breaking the port number into a two-byte (x,y) location. The grid can be magnified in specific areas to provide more detailed information about specific ports. Lastly, the tool contains a port display which displays information on five attributes of a selected port. The 3D display is a set of 5 bar graphs where time is represented on the horizontal axis and traffic volume on the vertical. This visualization tool shows many different types of data at the same time using several different 2D graphs. Portvis takes an interesting approach to visualizing port traffic; however, time-dependent patterns are defined very coarsely (2,048 port buckets) which could make detecting temporal patterns difficult. Furthermore, there may be issues of information overload especially under conditions of extreme port traffic when the port grid is filled with many colors. NetBytes Viewer improves on this by showing time as a third dimension allowing the user to see data trends on ports more quickly with no need for port volume aggregation.

IDS Rainstorm (Abdullah et al., 2005) was designed to visualize IDS alerts for the Stealthwatch anomaly-based IDS system. The main visualization divides the display into a set of columns that represent a contiguous set of IP addresses. The horizontal portion of each column represents a 24 h period of alerts. Colored dots in a row represent the total alarms for the IP addresses at a particular instance in time. The user can use a mouse to highlight an IP address range. Upon clicking on a selection, a secondary window is launched with an enlarged view. The selected IP addresses appear on a vertical axis on the right side of the window. Time appears on a horizontal axis and the alarms appear as large colored glyphs. This works shows how a drilldown mechanism can be used to view specific data in more detail. NetBytes Viewer utilizes drilldown to launch 2D views of multiset data to gain more insight into network traffic patterns.

The work of (Komlodi et al., 2004) is compelling as it places the user at the center of the design process itself, and offers a great deal of flexibility over the visualization. The user is able to visualize IDS alerts in both 2D and 3D windows and set the mappings between variables and visual attributes of glyphs such as position, size,

opacity and color. In addition, they derive a set of new guidelines for the design of information visualization tools for intrusion detection.

In “Situational Awareness and Network Traffic Analysis” (McHugh et al., 2004), the authors look at various techniques for analyzing and visualizing network traffic. Although the authors did not develop any interactive applications as part of the study, they do show a set of graphs which help to visualize network traffic at aggregate and individual host levels. One such graph is a 3D chart of an individual host which displays bytes transferred from a host per port per hour over a specified range of time. It allows the user to easily see patterns in port traffic which allow for quick diagnosis of anomalous behavior of the machine. NetBytes extends this 3D graph visualization by making it interactive so that patterns can be analyzed much more in-depth.

3 Technical Approach

The NetBytes Viewer was designed to visualize processed NetFlow data. Application volume data is obtained from multisets (bags) produced by the SiLK Analysis Tools from NetFlow data and can represent flow, packet, or byte counts.

NetFlow is an open protocol developed by Cisco for collecting IP traffic information. It “is an abstraction that provides a level of detail that is less than that from packet headers, but greater than session summaries. Flow records capture source and destination addresses, protocols, ports (for TCP and UDP), traffic volumes (packets and bytes), and start and end times” (McHugh et al., 2004). Such records can be generated using a NetFlow enabled Cisco router on a border network then collected and analyzed using a set of data analysis tools called SiLK. SiLK has the ability to perform many operations on the data including bagging (counting) flows, bytes or packets for specific unique key identifiers such as source ports, destination ports, source IP addresses, or destination IP addresses. This bagging technique was used to manipulate raw NetFlow records which were subsequently placed in a relational database and selected for visualization using relational queries allowing for multiscalar visualization over a wide range of entities.

The NetFlow data used in this project was collected from a private office network. The data shown in this paper is from a network email server with significant IP traffic on ports 25 (SMTP), 110 (POP), and 443 (https used for email web access).

3.1 *NetBytes Viewer User Interface*

NetBytes offers a comprehensive set of different views to make it possible for a network administrator to thoroughly analyze the traffic for an individual host (or aggregate subnet) on a network. It does this by showing the traffic activity per port on an hourly basis over a specified time period. This allows the administrator to

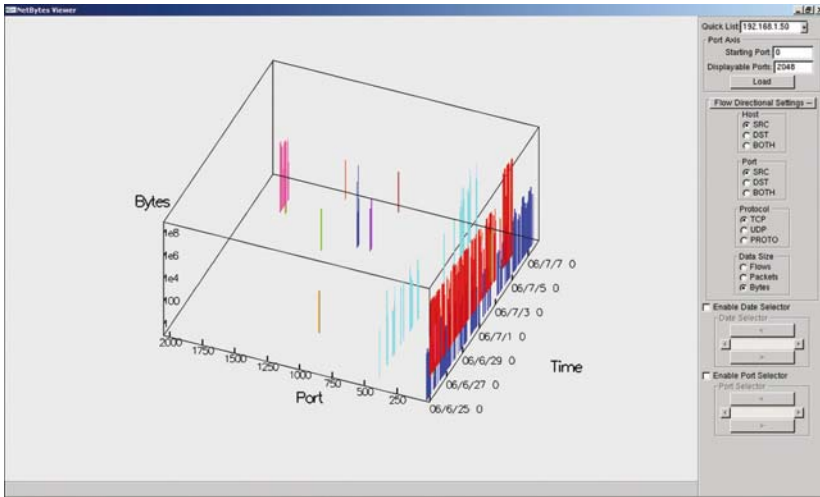


Fig. 1 NetBytes host overview representation of a 3D impulse graph (bytes vs. ports vs. time)

see patterns of anomalous traffic activity (e.g. traffic spikes or scanner activity) on suspicious ports. Furthermore, NetBytes enables the admin to pinpoint the time at which the anomaly occurred so that he/she may determine how the host became infected.

The initial overview screen (shown in Fig. 1) displays a global view of a single host's traffic structured as an orthogonal 3D impulse graph. Along the Z axis of the display is the port list for the machine (ranging from 0 to 2,048 in Fig. 1) while time (at an hourly granularity) is display on the X axis. The third dimension represents the magnitude of traffic (in flows, packets, or bytes) seen by the host (or subnet) in an hour.

Time was displayed as a third dimension to avoid the need for animation. Animation requires users to use short-term memory to remember trends and patterns in the data which could be missed or forgotten during playback. Furthermore, animation can cause issues of change blindness for the individual (Rensink et al., 1997). By contrast, with time on the third dimension, all the information is displayed on the screen, allowing the user to view and retrieve important information quickly. Color is used to specify all the data over time for a specific port. This helps to differentiate between the ports in the host viewer.

On the right-hand side of the main viewing window there is a sub window which contains a set of controls allowing the user to interact with the graph for a more detailed analysis of the data. A list box of IP addresses is used to select the host or subnet entity to be displayed in the main graphic window. NetBytes Viewer also gives the user control over setting which port data is displayed via a set of text boxes. Radio button groups allow for the data to be filtered based on data direction, protocol and volume measure (i.e. bytes, packets or flows). Sliders are also provided to select data points as will be described further in the next section.

3.2 User Interaction

Rendering a 3D graph on a 2D surface can cause several difficulties including occlusion, loss of depth perception and loss of head parallax (Ware, 2004). To deal with these issues, the overview screen is equipped with a number of interactive features. First, to help eliminate occlusion and perception issues, the 3D graph can be rotated around its center axis in all directions by holding down the left mouse button and dragging the view in any direction. This allows the user to view the data from all angles which facilitates the recognition of patterns and can also be used to disambiguate the 2D projection of the 3D graph. Furthermore, the user can drag the graph into two separate 2D orthogonal views (Volumes vs. Ports and Volumes vs. Time) as shown in Fig. 2. The “Volumes vs. Ports” view is useful to see the maximum network traffic per port while the “Volumes vs. Time” graph is useful for investigating peak traffic times.

Another issue with the 3D graph is that it can be difficult to perceive depth especially when there are several ports on the screen with associated data points. To help

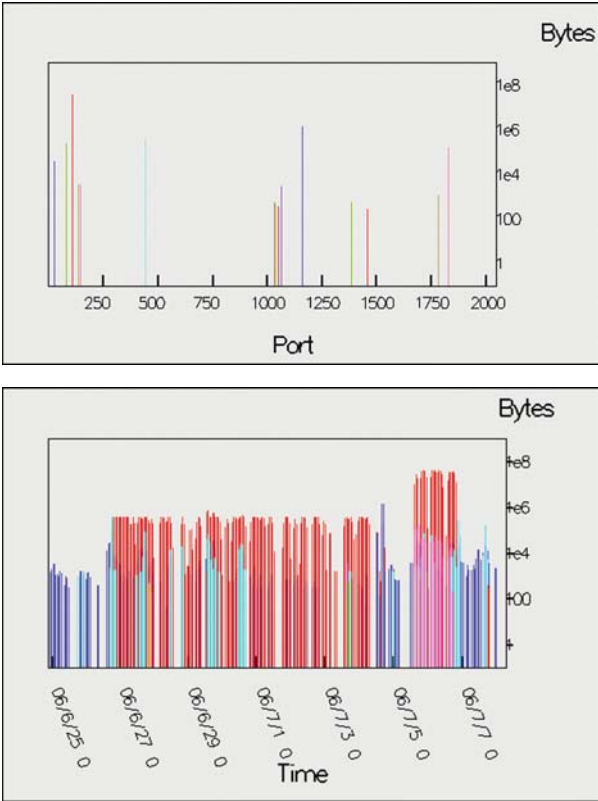


Fig. 2 Rotating the 3D graph into two separate 2D orthogonal views “Bytes vs. Ports” (top) and “Bytes vs. Time” (bottom)

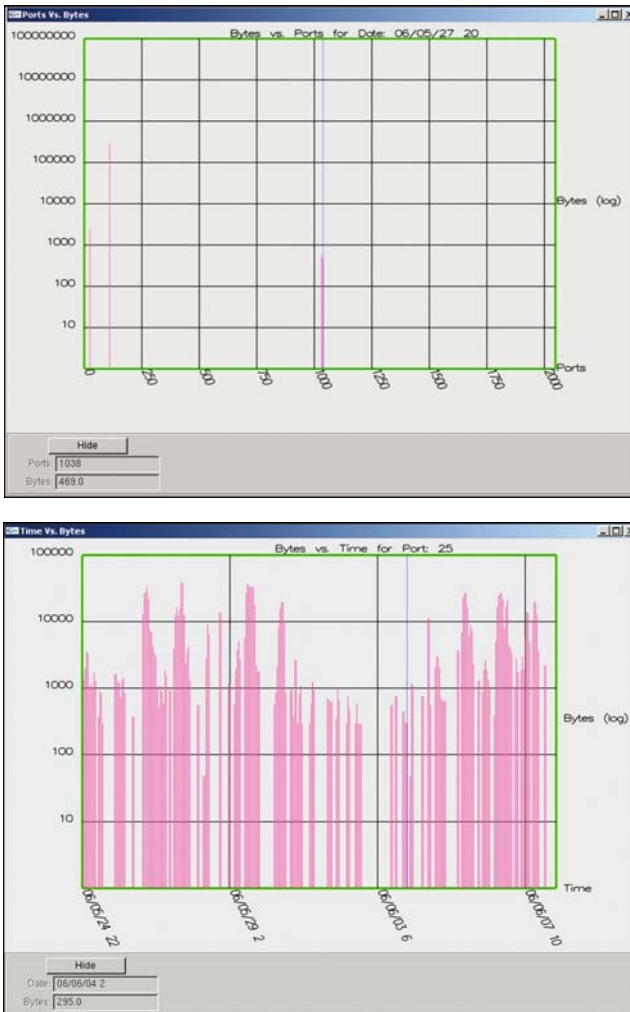


Fig. 3 “Bytes vs. Time” and “Bytes vs. Ports” graphs launched when application is in selection mode

the user deal with these issues, the NetBytes Viewer allows the user to look at data in more detail by launching a set of auxiliary 2D graphs (Volumes vs. Ports) and (Volumes vs. Time) as shown in Fig. 3. These 2D graphs are restricted to either a single time or a single port. This further reduces occlusion and perspective issues and increases the user’s ability to interact with the data. The 2D Volume vs. Ports graph, as shown in Fig. 3 (*top*), can be launched by clicking on the **Enable Date Selector** selection box, while the Volume vs. Time graph (Fig. 3, *bottom*) can be launched by clicking on the **Enable Port Selector** selection box. Clicking on these boxes will not only launch the graphs in a separate window, but also enable the corresponding slider in the main 3D view. Using the slider, the user can select a point

on the time axis (if the date selector is enabled) or the user can select a point on the port axis (if the port selector is enabled).

When the **Port Selector** slider is enabled, a green semi-transparent highlight band is displayed along the port axis and a slider is enabled at the bottom of the screen as shown in Fig.4 (*top*). When dragging the slider from side to side, the highlight band moves back and forth along the port axis and the corresponding 2D graph is updated in real-time with the time and volume data for the newly selected port. Clicking on the arrow buttons on either side of the slider causes the highlight band to jump to the next port which has data values. A similar highlight band (in blue) is displayable on the time axis when the **Date Selector** slider is enabled as shown in Fig. 4 (*bottom*).

It is worth noting that alternate approaches were considered for implementing the selection feature. Initially we allowed the user to click on a port number on the

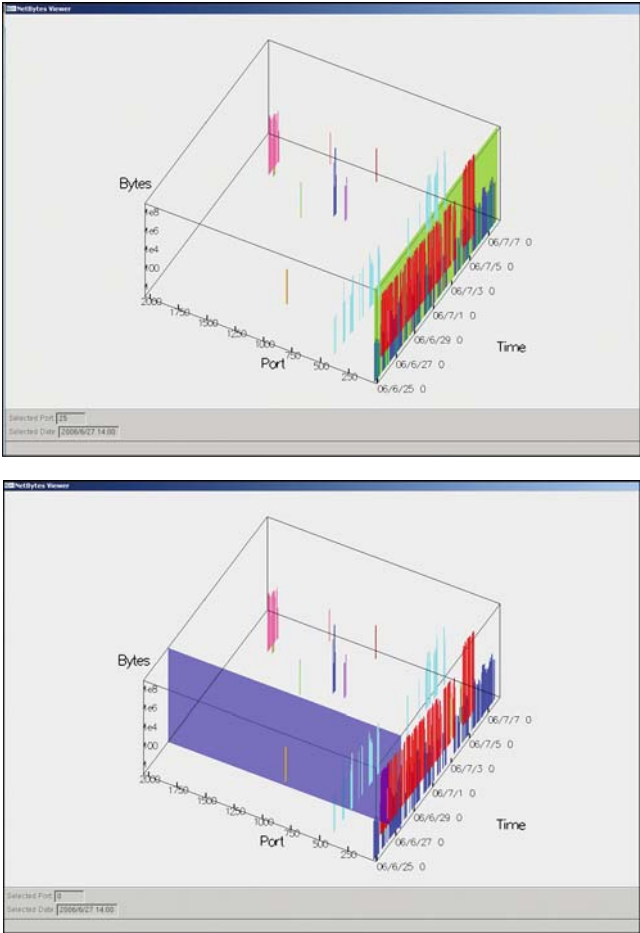


Fig. 4 Overview screen in port selector mode (*top*) and date selector mode (*bottom*)

port axis to highlight the data; however, with a large port range (around 2000+) this became impractical as it was difficult to accurately click on a port. A slider with snap buttons was more effective as the user is able to drag the slider to the vicinity of the intended port and then use the snap buttons to accurately select one.

The user can interact with the 2D graph using his or her mouse to highlight data points. Corresponding data values are displayed in the bottom left hand corner of the window, as can be seen in Fig. 3. These values are updated in real-time. Axes with a large number of data points can make it difficult to precisely highlight these values using the mouse; therefore, the **UP** and **DOWN** arrows were enabled on the window. Moving the cursor to a region on the graph and pressing the **UP** arrow selects the closest data point to the right-hand side of the cursor. Furthermore, pressing the **DOWN** arrow selects the closest data point to the left side of the cursor.

3.3 Implementation Details

NetBytes was written in C++ using the OpenGL graphics library (Woo et al., 1999). OpenGL was chosen for its power and flexibility in developing 3D graphics. OpenGL is a high performance graphics package written in C++ which is important when dealing with large amounts of data. Furthermore, it is platform independent which makes it useful for both Windows and UNIX users. Our aim is to release all tools as open source software on a variety of platforms.

NetFlows are gathered from a network border using the SiLK collection utilities and then processed with the SiLK bag tools. Afterwards, data is loaded into an SQL database by a small automated process. NetBytes uses this database when rendering its graphs. Using an intermediate database allows the application to capitalize on the more powerful SQL query language. It also ensures that the viewer can be used with other data analysis tools other than SiLK.

3.4 Case Studies

The goal of NetBytes Viewer is to enable administrators to quickly analyze the flow of traffic into and out of a host or subnet to determine if any malicious activity is occurring. A good example of where NetBytes Viewer proved useful was at a small networking research laboratory. The SiLK tools were installed to collect Netflow traffic at a border between the internal and external networks. As part of the research facility, the administrators house computers for external companies. Data was collected between February and March 2006 yielding some interesting results.

One host in particular produced fascinating visualizations indicating signs of systems being hijacked. Using NetBytes Viewer, evidence that the host was compromised is seen immediately as shown in Fig. 5 (*top*). As can be seen, there is

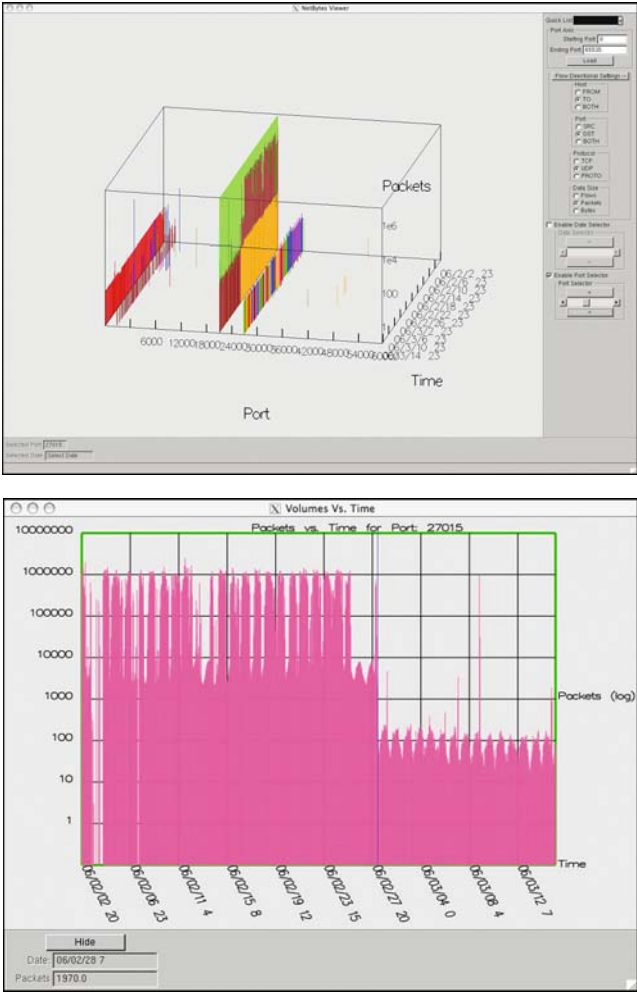


Fig. 5 Host compromised by game server (top) and port 27,015 (bottom)

a significant amount of incoming UDP traffic on ports 26,900 (yellow band) and 27,015 (brown band). These ports are typically utilized as game server ports for games like Half Life. The machine had been compromised and turned into a game server. Using the port selector feature, the port traffic was investigated in further detail in the 2D Packets vs. Time graph also shown in Fig. 5 (bottom). Many worms and viruses utilize high numbered ports to transfer data. NetBytes Viewer makes it easy to see the traffic on these higher ports to identify intrusion. Also of interest is the consistent traffic volumes on port 123 (red band in Fig. 5) which suggests this system is likely an NTP server for the network. Changing the filter parameters on the right control window allows the user to investigate the outgoing UDP traffic as

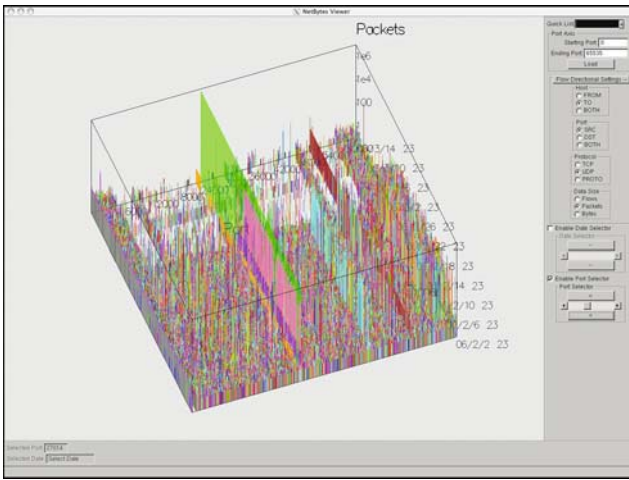


Fig. 6 Outgoing traffic for compromised host

shown in Fig. 6. Since the figure shows outgoing traffic, there is plenty of ephemeral port activity including two very prevalent bands of consistent data flow. One data flow is on port 27,014 (pink band that is highlighted by the port selector) and is likely the outgoing traffic for the game server running on the machine. The brown band maps to port 54,020 which seems to be typically used for client applications like FTP, messaging etc. on Linux servers.

Since NetBytes Viewer shows temporal data as well as traffic volume data on the same graph, administrators are able to see large data spikes in the context of normal traffic volumes. Excessive traffic volumes could be an indication of a denial of service attack or large file upload or download. Another SiLK data set from a private company showed a large spike in TCP traffic one evening during a two week collection period. This spike was clearly visible when using NetBytes Viewer using a volume-protocol-time graph. With a small amount of further investigation it was found that such a spike was the result of a planned back up of one of the company's data servers.

NetBytes Viewer is also good for looking at cyclic patterns in a network's traffic. For instance, in a company where workers would typically work from 9 to 5 Monday to Friday, one would typically expect the HTTP traffic to be relatively high through the week and almost non-existent on evenings and weekends. With NetBytes Viewer, network administrators can see these patterns by producing visualizations of entire subnets and networks and drilling down to see specific port traffic. If patterns do not exist, administrators can use the application on smaller subnets and hosts to track down the offending hosts. Ultimately, NetBytes allows the user to monitor how much data traffic is on his/her network over time and using this information can track down malicious behavior.

4 Future Work

The current NetBytes' feature set provides a good basis for an interactive host visualization; however, there are many more features that we intend to implement. A simple addition would be a mechanism for selecting a range of dates so that the user has more control over what is displayed in the 3D view. One way to do this might be to provide the entire date range on the selection slider, and then allow the user to highlight a range which would dynamically alter the corresponding axes on the graph. Another option is to provide a set of date pickers along the right-hand side of the main window, where the user could enter in the range values.

Another useful visualization technique that we intend to incorporate is non-linear distortion (Leung and Apperley, 1994). NetBytes is able to display all 65,000 ports as some viruses and Trojans are active on higher ports. But placing such a large port range on the screen at once causes the graph to become very large which can make it much more difficult to analyze. Since data tends to be sparse in higher port range, this range (2,048–65,000) could be distorted (compressed) to show all the data but in far less space. One way to do this might be to group ports into 256 partitions and collapse all the traffic for a partition into 1 aggregate impulse.

The application also requires a mechanism for displaying all the remote hosts that interact with the current host as well as the amount of data transferred per hour between the hosts. A possible approach might be to create another 3D graph which depicts “volumes vs. destination hosts vs. time”. Another option might be to launch the 2D graph of “volumes vs. ports” and then list the remote hosts on a per port basis off the 2D graph.

Other features that will be explored include:

- The ability to filter based on protocol (UDP, TCP, ICMP, etc.), magnitude of bytes transferred per hour, and individual remote host IP addresses. The latter filter would be excellent to view what data is being transferred per port between two machines on a network.
- The 3D graph could be made bidirectional which enables the user to view network data on both the source and destination ports of a particular host.
- Standardize the scaling values across 2D graphs so that they can be more easily compared.
- Ability to generate PDF files or jpegs of the visualizations.
- Ability to snap the 3D graph to a 2D orthogonal view.
- Use flooding techniques to display accurate Byte values on the 3D graph.
- Integrate NetBytes more closely with the SiLK NetFlow analysis tools.

The NetBytes viewer will also become a component of a larger network level visualization tool which shows connections between entities as well as entity behavior. This tool will support drill down to follow links between entity levels and to identify the sources or destinations of the volumes contributing to a single impulse. A user study will also be conducted to analyze the impact of the visualization tool on a network administrator's ability to do his/her job. Such a study is invaluable in gaining feedback on the tool's usefulness.

5 Conclusions

In conclusion, NetBytes provides a useful technique for visualizing network traffic flows to an individual host using a 3D Volume vs. Ports vs. Time graph. Furthermore, it provides a set of interactive features enabling network administrators to discover anomalous traffic patterns that could indicate signs of a virus or Trojan. As a result, it offers an effective drill down mechanism for a larger network analysis visualization tool. Many features will be added to NetBytes to improve its ability to effectively help in the fight against network intrusion. This work can therefore be viewed as the initial stage of a comprehensive network visualization project.

Acknowledgements This work was supported by NSERC and the Canadian Foundation for Innovation.

References

- Abdullah K, Lee C, Conti G, Copeland J A, Stasko J (2005) IDS RainStorm: visualizing IDS alerts. Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC), 1–10
- Ball R, Fink G A, North C (2004) Home-centric visualization of network traffic for security administration. Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, 55–64
- Fink G A, Muessig P, North C (2005) Visual correlation of host processes and network traffic. Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC), 11–19
- Komlodi A, Rheingans P, Ayachit U, Goodall J R, and Joshi A (2004) A user-centered look at glyph-based security visualization. IEEE Workshops on Visualization for Computer Security, 21–28
- Lakkaraju K, Yurcik W, Bearavolu R, Lee A J (2004) NVisionIP: netflow visualizations of system state for security situational awareness. Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security, 65–72
- Lau, S (2004) The spinning cube of potential doom. Communications of the ACM, 47(6):25–36
- Leung Y K, Apperley M D (1994) A review and taxonomy of distortion-oriented presentation techniques. ACM Transactions on Computer–Human Interaction, 1(2):126–160
- McHugh J, Gates C, Becknel D (2004) Situational awareness and network traffic analysis. Cyberspace Security and Defense: Research Issues, 209–228
- McPherson J, Ma K, Krystosk P, Bartoletti T, Christensen M (2004) PortVis: a tool for port-based detection of security events. Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security, 73–81
- Rensink R A, O’Regan J K, Clark J J (1997) To see or not to see: the need for attention to perceive changes in scenes. Psychological Science 8(5): 368–373
- Ware C (2004) Information Visualization: Perception for Design. Morgan Kaufman, Los Altos, CA
- Woo M, Neider J, Davis T, Shreiner D (1999) OpenGL Programming Guide: The Official Guide to Learning OpenGL. OpenGL Architecture Review Board, Addison-Wesley Professional
- Yin X, Yurcik W, Li Y, Lakkaraju K, Abad C (2004) VisflowConnect: netflow visualizations of link relationships for security situational awareness. Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security, 26–34

Visual Analysis of Corporate Network Intelligence: Abstracting and Reasoning on Yesterdays for Acting Today

D. Lalanne, E. Bertini, P. Hertzog, and P. Bados

Abstract This article proposes to go beyond the standard visualization application for security management, which is usually day-to-day monitoring. For this purpose, it introduces a pyramidal vision of the network intelligence and of the respective role of information visualization to support not only security engineers, but also analysts and managers. The paper first introduces our holistic vision and discusses the need to reduce the complexity of network data in order to abstract analysis and trends over time and further to convert decisions into actions. The article further introduces the analysis tasks we are currently tackling. The two following sections present two different ways to overview network data concentrating on specific dimensions of network security: user and application centric firstly, and alarm and temporal centric secondly. Finally this article concludes with the limitations and challenges introduced by our approach.

1 Introduction

Most of the visualization tools designed and implemented so far for the domain of corporate network security generally support day-to-day monitoring of network activities or high level security dashboards. However, numerous other user profiles and needs are related to the administration and analysis of a computer network in a company, and there is an increasing need to analyze and take decisions on this resource and its related information. In other words, not only the security team and the system/network engineers are nowadays interested in reflecting on the network

E. Bertini and D. Lalanne
DIVA/DIUF University of Fribourg CH-1700 Fribourg, Switzerland, e-mail: e.bertini@unifr.ch, d.lalanne@unifr.ch

P. Hertzog and P. Bados
NEXThink S.A. Parc Scientifique, PSE-B CH-1015 Lausanne, Switzerland, e-mail: p.hertzog@nexthink.com, p.bados@nexthink.com

topography, users and applications, but also higher level decision-makers such as the security architect, the chief security officer, the helpdesk, legal department or even employees farther away from the raw network resources, such as the business managers, the chief information officer and finally the chief executive officer.

In this position paper, we propose to consider network intelligence as a central resource of the company and consider the role of interactive visual tools for supporting not only daily monitoring but also other administrative activities related to a corporate network, which generally requires analysis over a longer period of time.

Most of the data related to a corporate network can be represented with the tuple *who* (users), *where* (hosts), *how* (applications/ports), *what* (alarms) and *when* (specific time). From our experience, the *when* parameter has a privileged role and can be taken into account at various levels of detail: day(s) perspective for monitoring or tracking activities; weeks, months for a deeper analysis in time; months, quarters or years for trending. In this article, we will emphasize on the analysis aspect since it is the middle layer standing between the security team and the management. We believe this middle layer can particularly be helpful for increasing knowledge and incrementally defining policies.

The various time levels mentioned above imply various user tasks and different levels of detail to be supported. Figure 1 represents our understanding of network administration, from day-to-day monitoring by system administrators to analysis and further trending by managers. Of course users can have various roles and are not stereotyped to one activity. To make it simpler, we define three main layers corresponding to three major user tasks:

1. *Monitoring*. This task is already well supported by visualization tools and allows tracking abnormalities on a corporate network and finding the related causes in order to take immediate actions to preserve the sake of the network.
2. *Analyzing*. This task stands in the middle layer of our pyramidal view. We believe this layer is particularly crucial and must be supported by visualization tools. There are numerous analysis tasks to support such as segmenting user types and applications through visual clustering, visualizing alerts over time as an

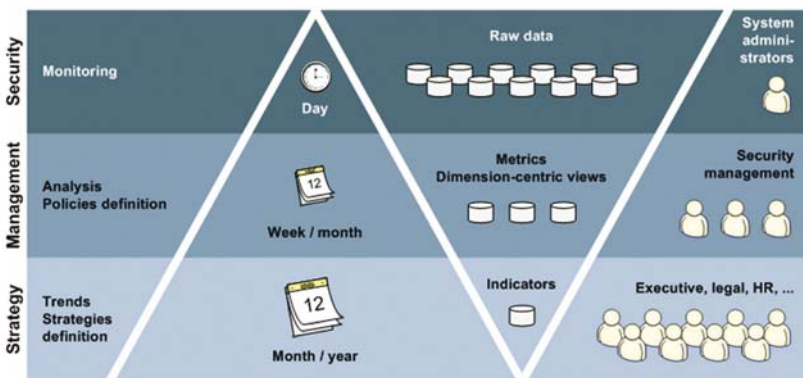


Fig. 1 Various levels of abstraction, various time granularities (day-to-day, month, year), and various roles of users, with different needs, are involved in network administration

entry point to understand relationships between network data, and also finding correlations, grouping similar elements, eliciting outliers.

3. *Trending*. Based on simple indicators, such as performance metrics, volume, license prices, etc. the management can observe its corporate IT evolution over time and can compare its own company with others in the field. The role of such kind of tools is the support for taking decisions and defining novel strategies, to be further converted in policies by its team members.

The pyramidal view in Fig. 1 also indicates that the level of details of the network data is inversely proportional to the time range of observation, mainly for two reasons: the increasing amount of data to handle when considering larger periods of time, and the level of details necessary to take high level decisions. Because there is a large amount of raw data, aggregation is necessary in order to reduce complexity when shifting from day-to-day monitoring to larger periods of time. A top manager will need to see how simple indicators evolve during a full year, whereas an analyst who needs to apply network policies will need to analyze activities within a month, in order to adapt management strategies and decisions into actionable policies.

2 Background

In the recent years the use of visualizations as a means to monitor corporate networks and detect potential threats has grown in interest, and a good number of systems have been developed. Visualizations can be useful in network security, not only for monitoring, but also to analyze evolution in time of large quantities of data. When mapping data to visual features, it is possible to perceive complex patterns at a glance and to reduce the burden associated to the reasoning activity. Visualization works as an external memory, offloading cognitive resources and considerably increasing the efficiency and effectiveness of analysis (Allen et al., 2000; Card et al., 1999).

One way to classify the existing security visualization systems is according to the nature of their data source. Tools such as visFlowConnect (Yin et al., 2004), nVisionIP (Lakkaraju et al., 2004), RUMINT (Conti et al., 2005) or TNV (Goodall et al., 2005) manipulate network flows or the results of packet inspection. Applications like MieLog (Takada and Koike, 2002a) or Tudumi (Takada and Koike, 2002b) use logs collected directly on the endpoints. And finally, RainStorm (Abdullah et al., 2005), SnortView (Koike and Ohno, 2004), STARMINE (Hideshima and Koike, 2006), VisAlert (Yarden et al., 2005), or other visualization-based tools that use an hybrid approach (Hertzog, 2006), visualize alarms directly generated by IDS. Regardless the type of data utilized by the system, and thus the kind of supported tasks, all these systems share the same principle of using visualization to increase *situational awareness* and to make more effective and simpler the detection and comprehension of abnormal behaviors. Even if these systems also support the analysis of threats and the formulation of possible solutions, their main focus remains the monitoring task. The quality of a security visualization system is measured in its

ability to convey all the necessary information and to provide administrators with a sense of control of their network. For this specific reason, most of these systems further share a time perspective limited to day-to-day monitoring.

While ensuring a daily network safety is of paramount importance, we want to investigate the idea of taking into account longer time spans and to consider the analysis of a network not only in the perspective of monitoring but also of strategic analysis: understanding relationships between data, managing a segmented population of users and applications, observing the evolution of various indicators (overall risk, volume, number of licenses, etc.), and finally devising adequate network policies. A similar approach is presented in (D'Amico and Kocka, 2005) where different types of analyses, job functions and uses of visualization are described. The need to go beyond the day-to-day monitoring task is also acknowledged in the domain of *computer forensics* where the data collected over long time periods is necessary to deeply analyze some potentially criminal behaviors (Allen et al., 2000) but this domain does not cover the needs we try to address here: the main objective in forensics is to follow the paths of some criminal acts to find evidence of them; here we propose to look into the data to increase knowledge and take informed decisions.

As soon as we shift from the daily monitoring paradigm toward the extended time analytics paradigm, we face the problem of data explosion. Security visualization can profitably draw ideas from other fields of computer science like data warehousing, data mining, and visualization, where the problem of coping with large quantities of data to visualize trends and patterns has been largely investigated. Data reduction and summarization are particularly pertinent here. As an example, in business intelligence there is a long tradition of methods and techniques conceived to cope with millions of transactions accumulated everyday (Ramakrishnan and Gehrke, 2000). With such a volume of data produced at a constant rate it is mandatory to decide what to retain and what to discard, and also at what level of abstraction data must be represented. An interesting initial study going in this direction is used by Hierarchical Network Maps (Mansmann and Vinnik, 2006) a visualization tool for monitoring data traffic, where OLAP data cubes are used to represent data hierarchically and at different level of details. In data mining there are also several methods that might be useful to our purposes to: help reducing the amount of data, produce more abstract descriptions of the data, and discover hidden information (e.g., dimensional reduction, sampling, clustering, rules induction, classification) (Han and Kamber, 2000). Visualization then has as well an established set of tools and techniques to deal with large quantities of data and/or to produce effective visual abstractions such as: pixel-based visualizations (Keim, 2000) and visualization from data cubes (Stolte et al., 2002).

3 On the Need to Support Visual Analysis

Figure 2 details the middle layer presented in the introduction and particularly emphasizes on the gap between the usual daily network monitoring and the very high level of decision making. Our claim is that this gap is currently not well bridged

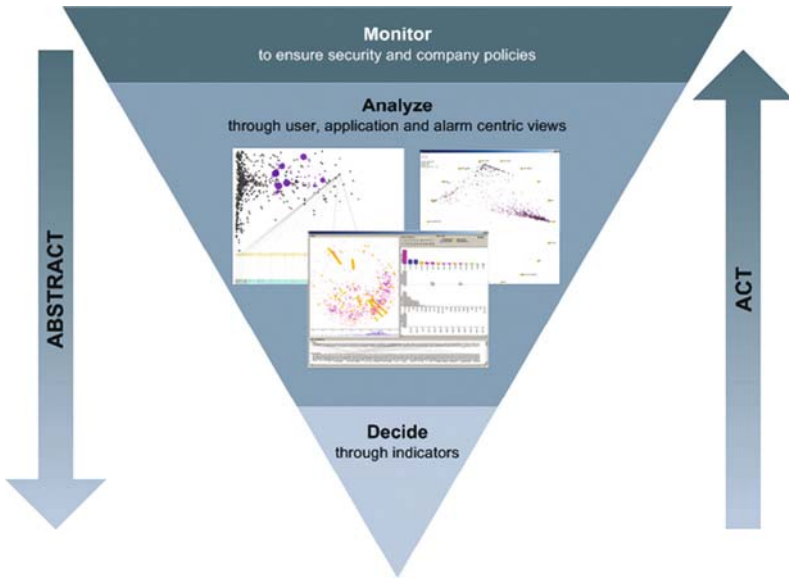


Fig. 2 Visual interactive tools are critical not only to support analysts in abstracting indicators to help decision makers defining strategies but also to convert high level strategic decisions into actionable policies

by any interactive tools, neither in the bottom-up direction, nor in the top-down. The bottom-up bridge should support the analysis of raw data in order to abstract indicators for the top management to easily take decisions, whereas the top-down bridge should ensure that the causes of the discovered trends are elicited, and the taken decisions are, for instance, converted by the analyst into actionable network policies.

Monitoring network data over time can be useful to observe how the network evolves according to taken actions (“*what if I add this policy, or dispense this awareness program to my users?*”). Visualization tools can help observing evolution and development in order to compare network status before and after specific intentional actions. Although it is hard to infer a strict causality between a specific action and the resulting network evolution, correlations between actions and impacts on the network can be clearly brought to light. Furthermore, we believe giving analysts the opportunity to annotate the network at specific times, or to label applications or users, could be of great value for assessing the impact of network policies or management strategies (control volume, optimize number of licenses, etc.). In a longer term, we could envision supporting prediction of network evolution on simple indicators, such as network overall performance based on customizable features.

To wrap it up, our belief is that abstracting network data in time and reducing complexity for the sake of understanding open various opportunities to increase knowledge, support high level reasoning, devise policies and strategies, and monitor

specific events, performance and risk over time in order to finally increase the overall security level. Our pyramidal view further sustains the idea of supporting collaboration between layers and people within an IT service, not limited to security.

3.1 Types of Analyses

In our view the analytical process can have a strong impact both on the lower and higher layers, interacting with each with different tools and purposes. Two types of analyses can take place.

- *Explorative (from monitoring to deciding)* – its main purpose is to find something useful or interesting without having a completely formulated and explicit goal. We see this type of analysis originating from the middle of the pyramid, thus mainly performed by security analysts as a way to inform administrators (below) and managers (above).
- *Explicative (from decision to action)* – its main purpose is to explain trends or behaviors of interest observed in the system. This is the type of analysis that is currently performed at the monitoring level to: (1) understand what is the real danger represented by the event; (2) understand the origin of the event. The same type of analysis however can be scaled to longer time spans and adopted by the higher levels of the pyramid. As an example, current trending tools, commonly found in dashboards, permit to see trends (mainly variations in time) but fail to explain why such trends take place. Being able to explain them might increase the ability to devise clever policies.

Explicative analysis takes requests from the outside to return explanations. Explorative analysis stems from an inner and less focused effort to analyze what happens in the other layers and to inform them when some useful knowledge is produced. Both types of analyses are useful and implemented only in few systems. We believe that the approach we propose can support both these tasks and increase the distribution of knowledge across all the layers within an organization or company.

3.2 Analysis Tasks

In our attempt to understand the kind of analytical tasks that can be performed in network security as soon as we abstract away from the view of “pure” network monitoring we have devised a list of possible tasks. This is certainly not exhaustive but is useful to understand what kind of knowledge might be generated. The following tasks are also the ones supported by the software tools and prototypes we have developed so far and from which we have gained most of our experience. In describing these tasks we consider that the following data on network’s traffic is available: source and target hosts, applications, ports, and user IDs and that some

form of alert system is in place. Following are the tasks we have isolated from our own experience:

- *Segmentation (who does what)* – the network can be seen as a place with actors (users) who exploit some resources and generate traffic and events. In this context one is interested to know who does what to segment the population or the resources according to the traffic or events they generate. As an example, in our SpiralView (presented below), it is possible to see which users, with what resources, generate some specific types of alarms: these elements are “segmented” in terms of alarm types. Many other methods of segmentation can be imagined.
- *Correlation, clustering, and outlier detection (building profiles)* – what is really difficult from the perspective of a network analyst is to summarize in few elements what are the typical behaviors/habits taking place in a network. As an example, it is certainly true that there will be groups of users who use the same set of applications in more or less the same manner and that spotting them would be useful to build user profiles and thus to devise specific policies for specific groups. In this task we consider all types of analyses permitting to identify homogeneous groups of resources that explain some relevant behaviors observed in the network. Another example would be to see if there are any emerging patterns between source hosts and target hosts, i.e., if there are groups of source hosts who usually connect to the same set of target hosts, and so on. It is worth to note that in the effort to find consistent groups able to expose some patterns, we often find outliers, i.e., elements behaving as no other elements in the network. Often these are at least as informative as groups, typically exposing malfunctions, potential threats or poor network management.
- *Alerts as entry point to the whole population (normal vs. abnormal behavior)* – if a system is equipped with some sort of alert generation system (e.g., an IDS) it is true that the whole network traffic can be split into two wide categories: resources involved at least once in suspicious behaviors and the others. If we look at the network through this lens we can recognize interesting opportunities. One is to use the resources involved in suspicious behaviors as an entry point to the whole population. One can isolate suspicious behaviors originating, e.g., from a group of alarms and see if there are other similar behaviors which do not generate alarms. Alternatively, one can compare the typical traffic of a resource and discriminate the traffic that generates alarms to better understand its nature.
- *Tracking and evolution* – While it is always possible to consider the data under inspection as the whole data accumulated so far (besides the obvious computational and scalability problems), we noticed that there is an interest per se in comparing the state of the network before and after some specific moments in time (e.g., the application of a new policy) or even to visualize the evolution through animated visualizations. To this end, it is also important to provide analysts with powerful annotation and tracking tools that permit to easily find their elements of interest and compare their status at different times.

The analysis vision proposed in this article might sound ambitious and we do not plan to solve it in one shot. In the rest of the article, we present three applications that take place in the middle layer of our pyramids and support various analysis tasks corresponding to the ones presented above. Those applications support the idea of reducing network data complexity through visualizations manipulating a reduced number of dimensions, that we call dimension-centric views, enabling to explore network data through specific facets. Further, we believe that there is an adequacy facet/task, i.e., some dimensions are more adequate for supporting some tasks. The following visual applications use a proprietary engine developed by NEXThink S.A.¹, which, differently from most existing engines, is able to convey, other than traditional data (such as IP addresses, ports, etc.), information about applications (in terms of binaries in opposition to protocols or services in most systems) and users (e.g., Windows SID).

4 User and Application Centric Views of the Corporate Network

Classifying users and applications within a company is a big challenge. The goal of this analysis is to visually represent the pertinent information to help the administrator answer this kind of questions:

- What is the typical user in my company? What is her/his typical behavior?
- How many different groups of users do I have in my network?
- Which is the typical profile of a “marketing” user? Has one user a behavior similar to this typical profile?
- Can we cluster applications and identify the family of an unknown one?
- Is there any differentiating pattern between users involved in alarms and users never involved?

We do not intend to completely solve those questions in the following subsections but rather to propose initial designs of dimension-centric visualizations to explore potential capabilities of tools addressing such problems. It is important to make clear from the beginning that some of these prototypes represent only initial designs and therefore their completeness or final effectiveness is not in question here. The following visualizations address similar problems from opposite point of views: while the RadViz aims at plotting similarities, the OriginalityView aims at plotting the uncommon. These two visualizations are examples illustrating the underlying adequacy between task, data facet, and visualization techniques.

4.1 The RadViz: Visually Grouping Similar Objects

In order to visually group similar users, and thus to find profiles, we use a customized version of RadViz (also known as StarCoordinates) (Hoffman et al., 1999;

¹ <http://www.nexthink.com>

Kandogan, 2001), a common technique to visualize n -dimensional datasets to find clusters and outliers. A number of anchors equal to the number of data dimensions are laid out in a circular manner. Each data item is connected to each anchor through a spring whose force is proportional to the value of the given dimension for the given data item. The dots occupy the position where the sum of the forces is equal to zero. As a result, the data points that share common combinations of values across all the dimensions occupy similar positions on the screen, thus segmenting the datasets in groups.

We apply this design to our particular case to find users who have similar application usage behaviors. In our design each user is a data item and each dimension an application with values corresponding to the user-application pair network usage (measured in terms of number of connections). An anchor can represent a class of applications, e.g., browser, email, multimedia, etc., or directly an application, and each dot is a single user in the network. The design is particularly suitable for the task because the visual technique is known to scale well as the number of data dimensions (i.e., applications) increases.

In our custom design we use a bivariate color scheme to distinguish between users who generated alarms, with red hues and increasing brightness as the number of total alarms increase, and users who never generated alarms, with a green hue. The size of dots is proportional to total network activity, i.e., big dots high activity, small dots low activity. The anchors also convey useful information. Their size is proportional to the number of users who use the application and its color to the total activity. In a single view we can isolate groups of users, and compare which applications they use, their activity level, and their alarm level. We can also spot the applications (anchors) that are most used and therefore that influence most the placement of dots on the screen.

The user can interact with the visualization in many ways. The applications/anchors can be filtered out/in to select interesting subsets. A small 2D scatter plot supports the user in this task displaying the anchors on a space where usage and number of users are mapped to the axes. The user can select subsets of anchors and use them to see how the population maps to the selected resources. The anchors can also be moved around the circle and their force can be increased/decreased in order to discriminate between clusters and isolate possible outliers (similar techniques are described in (Kandogan, 2001)).

By selecting one or multiple dots, it is possible to activate a bar chart that compares the usage profile of the selected users. This feature enables to understand what is the usage profile represented by the cluster and how the items inside relate each to another. As an example, in Fig. 3 we have selected the applications with the highest ratio between total activity and number of users. Four applications are markedly influential: *iexplore.exe* (Internet Explorer), *nlnotes.exe* (Lotus Notes), *ldiscn32.exe* (LANDesk Management Suite), *amclient.exe* (LANDesk Application Management Client). *nlnotes.exe* is the one that attracts most of the users (bottom right), as clearly shown on the figure. We can see at least three major groupings: users who mostly use *nlnotes.exe* (bottom right), users using mostly *iexplore.exe* and *ldiscn32.exe*, and those using mostly *ldiscn32.exe* and *amclient.exe*. We can also see that there is a big

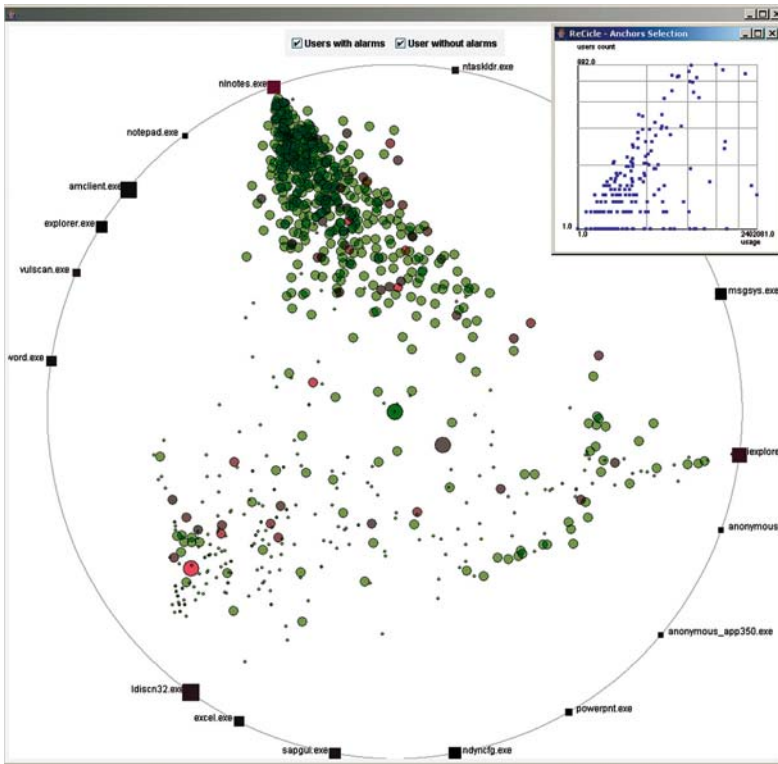


Fig. 3 The RadViz application: plotting similarities

red user on the bottom, quite distant from any other user. This represents the system user that comprehensibly is very distinct from the others: very high number of connections and alarms and an original composition of applications.

There are many others patterns that can be found using the interactive capabilities offered by the tool. Here we just want to give a glimpse of how this visualization can help in spotting groups of users and therefore in building profiles. However, the same technique can reasonably be applied to other combinations of network entities, e.g., to see how users use target hosts, or how applications use network ports. This kind of activity can increase the knowledge of a security analyst and help her/him in formulating novel strategies to apply.

4.2 The OriginalityView: Plotting the Uncommon

The OriginalityView (Fig. 4) exploits the, so called, originality metric as a way to discover original users, detect outliers, and segment user population according to their usage of applications. The originality metric, that we adapted from the

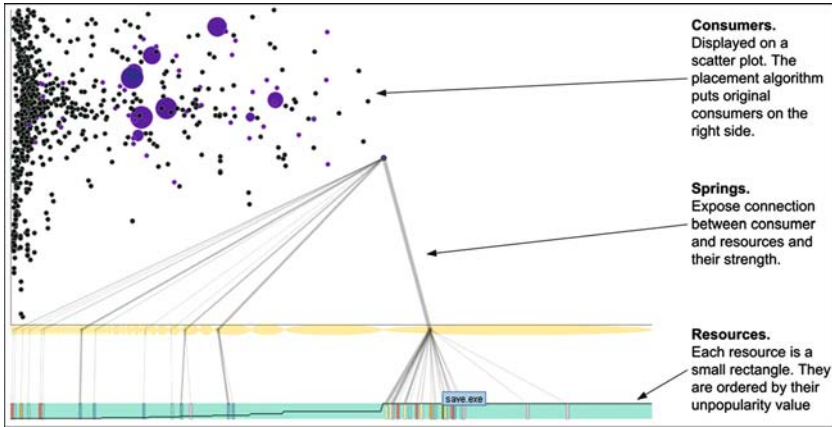


Fig. 4 The OriginalityView: eliciting outliers

standard TF.IDF used in information retrieval (Salton and Buckley, 1988), measures the weight of an application to segment a population of users. In other words, it measures how important an application is to a user in respect to the overall population. This originality metric takes into account the global use of an application, over the number of users that employ it. The corresponding OriginalityView consists of two main parts: the top is a scatter plot of consumers (users in this case) and the bottom an axis of resources (applications in this case). The small colored rectangles in the bottom area are the applications, colored according to the application category they map to (e.g., browser, file transfer, spyware), and ordered by their unpopularity value. The users are represented by the colored dots and their placement on the horizontal axis depends on their barycenter: each user is virtually connected through springs to all the applications he uses and attracted by them with a force proportional to originality value. When a user is selected, the visualization exposes his connections with applications through lines whose width is proportional to the originality value, which represents the force of the spring. This way it explains what makes a user original and draw his profile by highlighting the resources he uses. The given design leaves at least three relevant visual properties available for use. Key visual features like y-axis position, size and color can be mapped to other relevant parameters to investigate correlation with the originality. In our figure, as an example, we mapped the y-axis to average number of network connections (that is how active a user is), size to the level of access privileges (e.g., system administrator up to standard user), and color to estimated risk. Interestingly, purple users, the one with more associated risk, tend to be also original. Investigating more in details their usage pattern is very beneficial in terms of security findings.

Besides the complexity of the underlying placement algorithm and the absence of a real metric space on the horizontal axis, the final visual result is easy to understand: (1) original users and unpopular applications are on the rightmost side, (2) the lines connecting a selected user with the applications explain her/his originality and usage

profile. Even if the tool has not been evaluated through any formal user study, we gained some evidence of its understandability by showing it to stakeholders in the network security domain: the design principle was very easily understood and the interface can be operated with ease.

5 Alarm/Event Centric Views

The SpiralView is our first attempt to build a tool to analyze network data in large periods of time, in order to address various analysis tasks such as tracking and evolution, segmentation of users, and overview of the network considering a limited proportion of the population, as an entry point to the overall population. In its current implementation, the SpiralView is first of all a tool designed to observe alarms over time, and in an extended way, all types of events, as long as they are aggregated. Alarms are a particular case of events. Events are objects in time or instantiations of properties in objects. In principle, any temporal record on the network can be considered as an event. We also consider events being any known actions that could have an impact on a corporate network such as an awareness program given to a group of employees, or new security policies, etc. While the SpiralView can be used to monitor the network, its primary purpose is to support the analyst in reasoning about how the network evolves and in taking informed decisions on how to administer it. The focus is shifted from day-to-day monitoring, as a way to spot dangerous events and react, to the analysis of extended periods of time to devise policies that improve the network's behavior. Examples include: better targeted awareness programs, restriction or relaxation of network constraints, redefinition of access rules. To this end, the system also allows to attach notes to alarms or specific moments in time to remember when some strategies have been implemented.

All alarms generated in the system in the last k months are displayed, starting from the oldest in the center up to the most recent in the outer ring. The spiral shape has the following advantages over other time-based visualizations: (1) it can present data sequentially; (2) it exposes periodic behavior through radial alignments of objects; (3) it assigns more space to recent alarms. The perception of time periods in the spiral is extremely important. We decided to use a daily period as a default (that is, one ring represents one day) because this is the most natural way to see alarms in time from the point of view of an administrator, other layouts are available however (e.g., week layout) and might be used to expose periodic behaviors at different time scales. Certain types of network alarms, in fact, tend to be clustered around specific times in the day. The spiral thus follows a 24 h period, starting at midnight in the top, following with 6 am in the right, noon in the bottom, 6 pm in the left. The color of alarms represents their type because it is the most important information administrators use to discriminate between alarms, and corresponds to the same colors displayed in the bar charts for a ready correlation. Their size is mapped to the severity that is the second most important information.

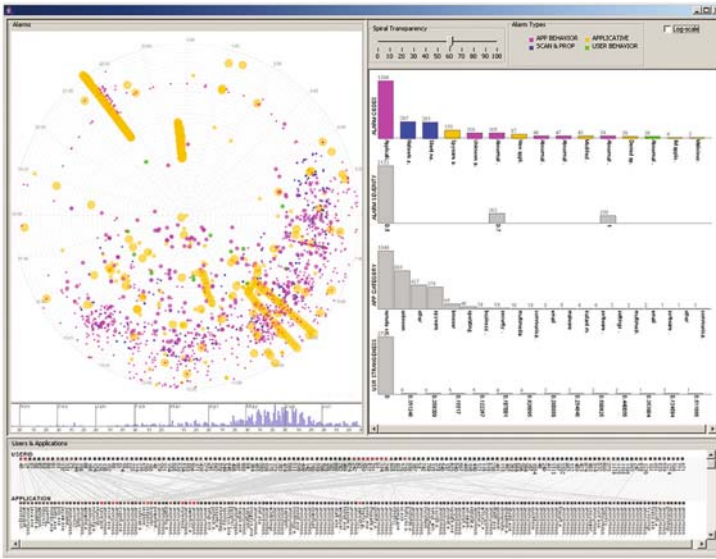


Fig. 5 The SpiralView: Analyzing alarms in time as an entry point to deeper analysis of relationships between network resources

The tool is provided with additional views, coupled with the spiral, to visualize related network resources and attributes. Their design is based on simple interactive bar charts and a custom user/application view which we chose because of their familiarity and ease of use. The bar charts measure the number of alarms falling in each category. As an example, the top bar chart in Fig. 5 presents the number of alarms pertaining to each alarm type category (e.g., network scanning, malicious activities, etc.). The user can select a single or a combination of bars, similarly to brushing histograms (Spence and Tweedie, 1998), to make queries and filter out alarms that are not within specified categories. The tool implements a two-way interaction mechanism. Selection on bar charts filters out and thus segments the set of alarms according to categories of interest. At the same time, interaction with alarms in the spiral enables to select groups and see how they map onto network resources. The spiral is further coupled with a time histogram at its bottom, which is used to convey aggregate data about how the total number of alarms evolves over time. The histogram is also used to select a time period in the spiral and zoom on it. We have also implemented an animated zoom that supports the user in understanding the change of view. When zooming in, each alarm is moved along a radial path and the substrate changes (e.g., the distance between rings grows) to reflect the change in time resolution.

Finally, thanks to annotation capabilities the spiral also serves as a communication tool between administrators and as a tool to keep track of the manual interventions made on the network. Indeed, the analyst can annotate it in order to label alarms or specific times to measure the effectiveness of deliberate interventions.

For instance, an administrator can enter an annotation on the fly, explaining the origin of the highlighted group of alarms and also marking the action undertaken on the engine or on the network to relax this type of alarms. This capability is extremely important in that it permits to remember when certain actions took place and thus to compare the status of the system before and after an intervention. Since the primary purpose of the system is to permit long term analysis and policies' assessment, with annotations not only it is possible to devise new strategies but also to check if and how new rules have changed the network's behavior and to share this knowledge between stakeholders.

The SpiralView's design is the result of various interactions with network security analysts from private companies who already use NEXThink's engine for more than a year. Taking into account real world tasks, they provided us valuable feedbacks on the usefulness of our visualizations and on their usability. The major concern we had to face with this design is the visualization performance and its complexity. Finer grain modifications have been made recently to improve the interaction and readability of the actual SpiralView (zooming mechanisms, brush/link with histograms, etc.) but its performance remains an issue to be solved. More information about the SpiralView can be found in Bertini et al. (2007).

6 Limitations and Challenges

The approach we propose in the article opens numerous challenges and limits to overcome. Visually abstracting relevant information to support trending by managers is not a trivial task. Finding the right balance between usability and completeness, implies finding the right level of abstraction/aggregation (visual vs. data aggregation), but also producing computationally reactive interactive visualizations. For instance, when developing our SpiralView we had to face usability problems, due to the computational time to interact with the view in production with thousands of users. We believe these problems can be often solved with user evaluations of the prototypes produced, and dynamic adaptation of the views to the network topography. However these approaches are time consuming efforts. We believe this problem can be bypassed by limiting the spectrum of an application, with fewer dimensions, and supporting only specific tasks. Similarly, finding the ways to produce fruitful exploration (actionable insights) is a very challenging problem since exploration by definition leads to unknown discoveries, outcomes and thus actions to be taken. For this reason, we believe supporting network administration is an incremental process, that should be supported with simple customizable tools that can be assembled in a toolkit, among which the final user can pick the tools of interest to build her/his own environment. Other challenges include finding communication bridges among the layers of the pyramid so that the different types of users within a company can exchange their findings and decisions. We currently believe that annotations are the best way to capture, store and exchange information, but

more formal representations might be found to be able to search and retrieve this type of information.

7 Conclusion

This article presents a pyramidal approach to network management, showing various levels of time and data granularity, in order to support various types of users: system engineers, analysts and managers. The paper first introduces our vision and discusses the need to reduce the complexity of network data to abstract analysis and trends in time and further to convert decisions into action. The article further introduces envisioned analysis tasks and presents two different ways to support them and to overview network data both concentrating on specific dimensions: users and applications (resources centric) firstly, and alarms (events centric) secondly. The paper presents in particular three visual tools built to support different tasks. We believe these tools are still too close from the tasks performed usually by system administrators and much more efforts are necessary in order to develop tools that support higher level roles and activities such as trending. Even though our approach opens numerous challenges, our major objective with this article is to support the idea that visualization tools can be useful for a broader range of applications related to the administration of a corporate network, that at the end will benefit not only to define adapted security policies but also to improve the understanding of the network usage within the company. Finally, our technical goal is to build a visual customizable toolbox, to bridge the gap between daily network monitoring and strategic trending and decision making.

Acknowledgements We would like to particularly thank Florian Evéquo for his brilliant implementation of the OriginalityView and the Swiss Innovation Promotion Agency CTI/KTI for financing this project.

References

- Abdullah, K., Lee, C., Conti, G., Copeland, J.A., Stasko, J. (2005) IDS RainStorm: visualizing IDS alarms. Proceedings of the IEEE Workshops on Visualization for Computer Security
- Allen, J., Christie, A., Fithen, W., Mchugh, J., Pickel, J., Stoner, E. (2000) State of the practice of intrusion detection technologies. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. CMU/SEI-99-TR-028,CMU/SEI
- Bertini, E., Hertzog, P., Lalanne, D. (2007) SpiralView: towards security policies assessment through visual correlation of network resources with evolution of alarms. IEEE Symposium on Visual Analytics Science and Technology (VAST)
- Card, S.K., Mackinlay, J.D., Shneiderman, B. (1999) Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, Los Altos, CA

- Conti, G., Grizzard, J., Ahamad, M., Owen, H. (2005) Visual exploration of malicious network objects using semantic zoom, interactive encoding and dynamic queries. Proceedings of the IEEE Workshops on Visualization for Computer Security
- D'Amico, A., Kocka, M. (2005) Information assurance visualizations for specific stages of situational awareness and intended uses: lessons learned. Proceedings of the IEEE Workshops on Visualization for Computer Security
- Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A. (2005) Preserving the big picture: visual network traffic analysis with TN. Proceedings of the IEEE Workshops on Visualization for Computer Security
- Han, J., Kamber, M. (2000) *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Los Altos, CA
- Hertzog, P. (2006) Visualizations to improve reactivity towards security incidents inside corporate networks. Proceedings of the Third International Workshop on Visualization for Computer Security
- Hideshima, Y., Koike, H. (2006) STARMINE: a visualization system for cyber attacks. Proceedings of the Asia Pacific Symposium on Information Visualisation – Vol. 60
- Hoffman, P., Grinstein, G., Pinkney, D. (1999) Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. Proceedings of the 1999 Workshop on New Paradigms in Information Visualization and Manipulation
- Kandogan, E. (2001) Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
- Keim, D.A. (2000) Designing pixel-oriented visualization techniques: theory and applications. *IEEE Transactions on Visualization and Computer Graphics* **6**: 59–78
- Koike, H., Ohno, K. (2004) SnortView: visualization system of snort logs. Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security
- Lakkaraju, K., Yurcik, W., Lee, A.J. (2004) NVisionIP: netflow visualizations of system state for security situational awareness. Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security
- Mansmann, F., Vinnik, S. (2006) Interactive exploration of data traffic with hierarchical network maps. *IEEE Transactions on Visualization and Computer Graphics* **12**: 1440–1449
- Ramakrishnan, R., Gehrke, J. (2000) *Database Management Systems*. McGraw-Hill, New York, USA
- Salton, G., Buckley, C. (1988) Term-weighting approaches in automatic text retrieval. Vol. 24. Pergamon Press, New York, pp. 513–523
- Spence, R., Tweedie, L. (1998) The attribute explorer: information synthesis via exploration. *Interacting with Computers* **11**: 137–146
- Stolte, C., Tang, D., Hanrahan, P. (2002) Query, analysis, and visualization of hierarchically structured data using Polaris. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
- Takada, T., Koike, H. (2002a) MieLog: a highly interactive visual log browser using information visualization and statistical analysis. Proceedings of the 16th USENIX Conference on System Administration
- Takada, T., Koike, H. (2002b) Tudumi: information visualization system for monitoring and auditing computer logs. Sixth International Conference on Information Visualisation (IV'02)
- Yarden, L., Jim, A., Shaun, M., Stefano, F. (2005) Visual correlation for situational awareness. Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization
- Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K. (2004) VisFlowConnect: netflow visualizations of link relationships for security situational awareness. Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security

Visualizing Network Security Events Using Compound Glyphs from a Service-Oriented Perspective

J. Pearlman and P. Rheingans

Abstract Network security is the complicated field of controlling access within a computer network. One of the difficulties in network security is detecting the presence, severity, and type of a network attack. Knowledge of such an attack is used to mitigate its damage and prevent such attacks from occurring in the future. We present a new visualization of a computer network for security purposes by approaching the problem from a service-oriented perspective. This approach involves a node graph visualization where each node is represented as a compound glyph, which gives details about the network activity for the specific node based upon its service usage. Furthermore, we visualize temporal activity using time slicing techniques in the compound glyph to give more details about the network and allow interactive controls for an administrator to actively monitor a network in order to react to security events quickly. Our resulting visualizations of networks successfully identified and described denial of service (DoS) and compromised network attacks.

1 Introduction

Network security is crucial to maintaining stable networks in order for institutions to continue normal operations. Network attacks are designed to cripple or disable normal functionality of a network, interrupting normal operations. A network administrator's primary task is to enable secure and legitimate communications between machines on a network. A large portion of this task involves both reactive and proactive prevention of attacks. The administrator is concerned with any type of anomaly that could represent an attack or an intrusion. Furthermore,

J. Pearlman and P. Rheingans
University of Maryland Baltimore County, Baltimore, MD 21250, USA, e-mail: jpearl1@cs.umbc.edu, rheingan@cs.umbc.edu

the administrator must recognize signature-based network threats, as most attacks follow some pattern. The network administrator performs three tasks: monitoring, analysis, and response (Komlodi et al., 2004). In the first phase, *monitoring*, the administrator attempts to find something problematic about the network, such as an attack or unauthorized access. Once a problem is found, the administrator will *analyze* the specifics of the problem in order to *respond* by taking steps to correct the issue and prevent the problem from occurring again. Network security visualization aids the network administrator in the first two generalized tasks, monitoring and analyzing. The more specific tasks that a network security administrator must perform in the monitoring and analysis stages include detecting insecurities, detecting intrusion attempts, defending against network attacks, and detecting resource misuse.

The primary goal of network security visualization is to provide a network administrator with visual information that allows the administrator to perform their job-related tasks, including identifying and preventing unauthorized access to resources, attacks on their network, and misuse of resources from within the network. One of the difficulties of this task is handling large amounts of data and filtering the data in such a way that security events stand out. Another difficulty is enabling the visualization to show data for individual nodes while showing data for the entire network to better detect and understand security events. Most of the current network security visualization techniques focus on one of these areas, either displaying data for only one node on a network, or displaying overall network data without going into detail on the particular nodes. Without an understanding of the node's traffic in its significance within the overall network traffic, certain types of attacks are difficult to detect.

The specific problem we address is visually aiding a network system administrator in identifying security events on their network from a service oriented perspective. The major goal of this research is to provide a visual means for a network administrator to take steps to prevent attacks, mitigate damage from attacks, and monitor service traffic. The features that a visualization tool must provide to perform this task is to visually identify anomalous behavior in a network and allow the administrator to gain information about the anomalous behavior. The visualization tool must provide service information about each node on the network to detect anomalous behavior at the service level. Finally, the visualization tool should be able to use temporal data to distinguish between heavy usage and attacks. This visualization will aid the administrator in identifying the presence, severity, and type of network security even present in a network by representing network data at the service or application layer.

This research will visually identify service activity on a per node basis with an emphasis on anomalous service activity. Any inbound network activity occurring on a service that the administrator is unaware of is a cause for concern. Particular services, such as Internet relay chat (IRC), are commonly used for outbound traffic when Trojans or worms are present on a network. Also, common service activity from a set of machines to a machine on another network can lead to the discovery of an attacker's control points. This research will provide a real time system for

network administrators to monitor service activity, allowing for early detection of attacks, such as a denial of service (DoS) attack.

We present a new approach to network security visualization by extending existing approaches in order to add service and temporal information into the node itself. We begin with a node scatter plot, which is similar to other approaches based upon network traffic data sets. Within each node in the scatter plot, we embed more information than previous approaches by using time slicing and service differentiation visualization techniques. By visualizing different service activity over time on a per-node basis, we are able to differentiate between attacks, discover more details about the attack, and identify different types of attacks not available in previous scatter plot node graph visualizations of network data.

2 Related Work

A cluster of previous research represents the structure of the network as a graph of links and nodes. Becker performed research on visualizing node links; this research is frequently cited because most visualizations contain some type of node link information (Becker et al., 1995). The vertices in the plot represent machines, and the edges in the plot represent network connections. Ball introduced a method that reduced the clutter by focusing in on nodes for a specific network and called this a home-centric approach (Ball et al., 2004). Teoh's work took node links to a higher level of detail, by using focus + context techniques to display time data in addition to node link data (Teoh et al., 2004). In another application of this technique, Goldring experimented with using scatter plots for various visualizations, including network link traffic (Goldring, 2004). Building upon the common node link scatter plot, Erbacher used glyphs instead of dots as nodes, and glyphs instead of lines for links, in order to add more information to the visualization (Erbacher, 2002). Ball uses size to represent amount of traffic and opacity to represent inactivity time (Ball et al., 2004). These explicit approaches have a tendency to be overwhelmed by very large network sizes.

Other previous research uses other information visualization techniques to show the multivariate nature of network data. Conti used parallel coordinates to plot external port, internal port, source address, and destination address (Conti and Abdullah, 2004). Yin et al. (2004) used NetFlow data, a format that logs network data transferred from end to end by ignoring intermediate communications, to visualize a network. McPherson et al. (2004) used a color mapping technique to identify interesting ports on a network. Papadopoulos et al. (2004)'s Cyberseer is a unique combination technique using visual and auditory techniques for network monitoring. Girardin (1999) used a self-organizing map to visualize port activity over time. NVisionIP (Lakkaraju et al., 2005) uses a galaxy view as its main view but then allows other more focused views on selected areas of interest.

3 Technical Approach

The method used to solve the problem of identifying anomalous network nodes to determine attacks is a glyph-based visualization technique similar to scatter plots of network traffic developed in previous work. Our method adds more information to the glyphs in order to better identify anomalous behavior by the service activity. In order to identify vulnerable nodes and network attacks, more information than general network activity must be used, such as service information. The service activity information can differentiate between normal usage and potential attacks.

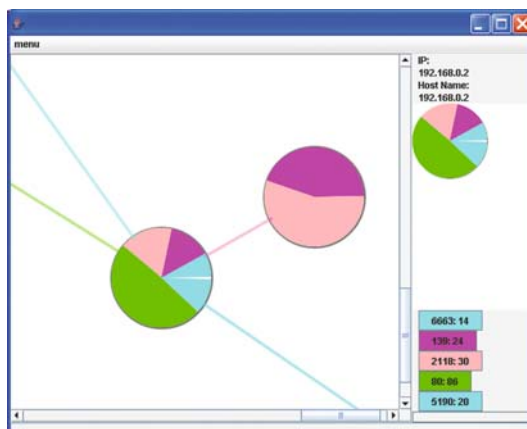
In order to detect DoS attacks and compromised networks, it is important to distinguish traffic based on service type and time. Our method combines several techniques to accomplish the goal of determining the presence, severity, and type of a network attack. Like in Ball's approach, glyph representations and differentiation between managed and unmanaged nodes are techniques used in this method and like in Girardin's technique, we break down difference service activity visually. However, service activity is displayed within each node in order to give more detail about network attacks. Furthermore, temporal data is displayed in a static manner to allow for more analysis on an attack. Finally, adjusting opacity on a per glyph basis is used to compare the network to normal network conditions for detection of anomalies.

3.1 Network Node Glyph

This method maps port information to a glyph representing a node on the managed network. Each open port, or service running on a machine, exposes a potential point of entry, authorized or unauthorized. Each glyph, representing a node on the network, represents the presence and amount of activity for a particular service. Each glyph is a compound representation of services and their activities, with each region of the glyph representing the amount of activity on an open service of a machine. The size of the glyph represents the total amount of activity on the node, while the regions identify what percentage of that total activity belongs to a particular service. The size is scaled with amount of activity, measured by number of packets, with a relative maximum size. The glyph contains a representation color for each different service but reuses colors for services because there are more services than visually distinguishable colors (approximately 65,000 different services). Following Ball's home-centric approach, managed nodes on the network will be rendered differently from unmanaged nodes outside the network (Ball et al., 2004). Finally, node activity links exist between managed nodes to other managed nodes, and unmanaged nodes to managed nodes.

Service mapping. The compound glyph representation is a pie chart. Each service is a region of the pie chart, with its size representing the amount of activity on that service and its color differentiating it from other services. The simple pie chart representation adapts well to adding temporal data. When raw traffic is seen on the wire, several attributes are determined from the network packet including source

Fig. 1 Service mapping of a node. Mouse over highlighting provides a key which details the ports of each service region. Host 192.168.0.2 is performing several different activities during this visualization, including web traffic, chat traffic and file sharing traffic to another managed node

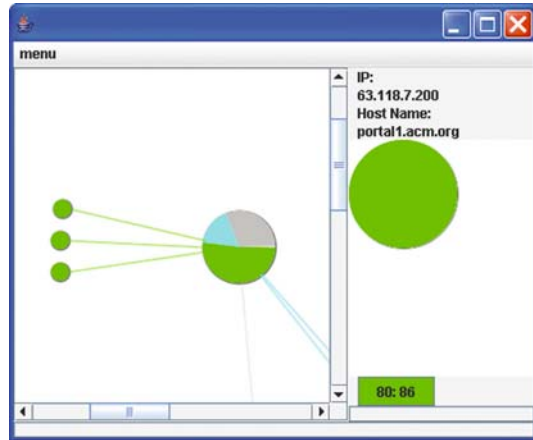


address, destination address, and the type of protocol. The protocol can often be determined by the destination port of the initial packet that establishes a connection (the SYN packet in TCP). However, ports can be customized so looking at the actual data in the packet can more accurately determine the protocol. The protocol then determines the service. In each node representation glyph, an arc is created for each service and filled with a color mapping to that service. Figure 1 shows two managed nodes which are divided up into several services. The managed node 192.168.0.2 is highlighted to show the key which displays ports for each service represented by a colored region. Notice that both nodes in this visualization are participating in various different service activity, and from the key of the highlighted node, we can see that node 192.168.0.2 is conducting approximately two-thirds of its traffic as web traffic. It is also using the instant messenger service (port 5190) and windows rpc service (port 139) in lesser amounts than the web service.

Managed and unmanaged nodes. Managed nodes will be visually larger than the majority of unmanaged nodes barring relatively large amounts of activity occurring on an unmanaged node. Furthermore, unmanaged node traffic that does not have one endpoint at a managed node will not be visualized. Unmanaged node traffic (both endpoints are unmanaged nodes) is rarely useful in determining a network attack and is extremely vast, since this effectively means all Internet traffic! Also, having access to capture all traffic on the Internet poses a different problem. By eliminating unmanaged node traffic, the administrator can focus on managed nodes and their communications. Figure 2 is an example of one managed node connecting to three different unmanaged nodes. Each unmanaged node happens to be a web server, the one highlighted by the key being the ACM portal. In this visualization, the larger glyph, representing a managed node, has slightly more than half of its activity occupied by web traffic to three different web sites.

Temporal activity. Another goal of the visualization is to show the change of the amount of service activity over time. Reoccurring network activity on a predictable schedule is an indication of a worm or trojan on a machine and a stronger indication when the reoccurring network activity is of similar amount and using the

Fig. 2 Managed nodes vs. unmanaged nodes. Managed nodes are represented by larger glyphs and placed in a centralized location of the visualization. This figure shows a managed node connection to three different web sites, one of them being the ACM portal, which is highlighted by a mouse event



same service. Shanbhag et al. (2005) used visual time slices to display changing attributes over time. They present several different methods including rings, slices, and wedges to display temporal data of a region. In order to display the change in activity for a particular service, this technique is applied to each region of the glyph that represents the activity of a particular service. The size of the region represents the amount of activity for the service, the color of the region distinguishes the different services, and temporal slicing is used within the region to show when network activity occurred on the service represented by the region. Ring based temporal slices are used, with the most outer ring representing the most recent time slice. Visually, the slicing is trivial to display by drawing the outer pie chart first and drawing inner pie charts on top of the previous one. The size of each time slice is determined by subtracting a variable step size from the size of the previous radius of the time slice, which eventually hits zero when too many time slices are created. The amount of time each slice represents is configured by the user. However, since the outer time slice represents the most recent data, the oldest time slices will disappear first. Figure 3 shows an example of a display of a node cut into four time slices. In the oldest time slice which is the center of the circle, only two types of traffic are present: IRC represented by yellow, and instant messenger chat represented by cyan. In the second oldest time slice, different service activity occurs including windows file sharing activity represented by pink, email traffic represented by gray, and web traffic represented by green. In the second most recent time slice, only web and instant messenger traffic is present. In the most recent time slice or the most outer ring or the glyph, more than 75% instant messenger traffic occurs with a little web browsing traffic and some IRC chat traffic.

3.2 Layout

The layout of the nodes follows a trivial formula and is not a primary focus of this visualization. The (x, y) coordinates of the node on the two-dimensional plane are

Fig. 3 A glyph representation of a managed node on a network. This node is sliced into four time slices, each representing a different amount and type of service activity

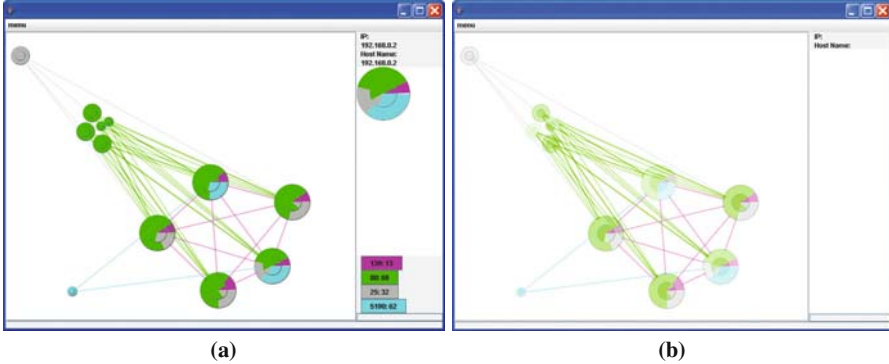
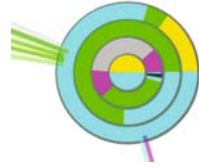


Fig. 4 A visualization of a small normal network. Notice that each of the web traffic nodes (*green*), representing web sites the managed nodes are visiting are grouped in the same area. One time slice has been performed on this visualization among all nodes. **(a)** Without comparing to a simple model. **(b)** Comparing to a simple model that detects abnormal behavior. Nearly all nodes are faded because our simple model doesn't detect very anomalous behavior

calculated based upon the type of activity occurring within that node. For managed nodes, their location starts near the center of the viewing area and randomly moved slightly from the center to create some separation from other managed nodes. For unmanaged nodes, the service with the largest amount of traffic is found and used to position the node along the *x*-axis and slight randomization is added to prevent excessive overlap. By performing this positioning, unmanaged nodes with similar characteristics are grouped together, allowing the user to treat such groups as larger entities if needed. In Fig. 4, notice that all of the unmanaged nodes producing web traffic are grouped together. Finally, we allow the user to manually reposition nodes as well.

3.3 Comparing to a Model

Because all networks are different, it is difficult to generically detect normalcy in a network. Because of this, an interface to a model for a network was created to have a visual method for comparing a network to a “normal” network. A “normal” network can be defined by a custom model which follows a limited interface, which includes functionality for rating the abnormality of a node in the actively monitored network. A custom network model is used for this feature because every network has

a different definition of normal activity. Using a direct relationship to the anomaly factor given by the network model, the visualization modifies the alpha value of the node. This technique will allow the administrator to fade out nodes that are not considered abnormal by the network model.

The model interface requires the model to return an anomaly factor for a specific node. The node's graphical representation will be altered by opacity based upon the model's anomaly factor for the node. As an example, we use an overly simplified model to demonstrate this visualization technique. The sample model works by starting with an anomaly factor of 0.3 on a zero to one scale. The model is based on college dorm traffic just like the simulation. Activity that we consider normal, such as web and instant messaging traffic, lowers the anomaly factor. Unknown service traffic raises the anomaly factor and ICMP traffic raises the anomaly factor by an even larger step. ICMP traffic is generally used for obtaining detailed information about a network but our sample network only has one administrator and does not expect to see ICMP traffic initiated by other sources. Furthermore, if there is traffic to a select group of common web sites, the anomaly factor is lowered slightly and otherwise raised slightly. A simple host based check raises the anomaly factor when having traffic to or from sites without a common suffix like .com, .net, or .edu. In Fig. 4 we apply the comparison of a normal network to our simple model. Because this visualization is of a normal network, our simple model detects the nodes in the network as fairly normal and therefore fades them out in the right image which is compared to a model.

3.4 Results

These methods were applied to a simulated network consisting of a small set of client users representing college student's dorm computers with some added network servers. Most of the common types of attacks were identifiable by applying these methods and creating a visualization of the network under attack. More specifically, details of the attack can be determined from the visualization down to which computer and which service is under attack. Session hijacking and man in the middle attacks will not be visible using these methods as that different style of attack would require a different visualization approach. This visualization deals with endpoint to endpoint traffic which will not identify changes along the route which occurs in man in the middle attacks. Furthermore, in order to detect session hijacking, unmanaged node to unmanaged traffic must be visualized, which creates large scalability issues.

Network packets are captured at each machine on the network and stored in some pcap (packet capture) format. Each packet consists of TCP/IP header information, containing meta information such as the source and destination of the packet. In addition to the source and destination address, the header contains the source and destination port. The source and destination address allows for mapping of activity on a node to node basis. The destination port allows for mapping the availability of services on a node in addition to the amount of activity for a particular service on

a node. This application is capable of real time network monitoring and therefore needs the capability to sniff network traffic in real time. JPCap (Java API to the pcap library) is used for real time packet sniffing and creating simulated network packets (Charles, 2001).

3.4.1 Compromised Network

Figure 5 is an example of an IRC trojan which has infected several machines on the managed network. IRC traffic is mapped to a yellow color and covers ports 6667–7000, which are the typical set of ports used for IRC. The small yellow sections indicate a small amount of IRC traffic coming from the managed nodes, in each of the time slices. The small yellow node is the control center the attacker uses to control the IRC trojans. Trojans are difficult to detect from a network administrator view who is not familiar with the network. Certain trojans use fairly uncommon services, for example, Backdoor.IRC.Snyd.A uses IRC as its command protocol. When comparing the infected network to a simple network model, normal traffic can be faded, highlighting potential risks. A very basic model, which does not consider IRC common traffic, can highlight infected nodes in Fig. 5. Even if IRC was commonly used by several users on the managed network, if the administrator is familiar with seeing such traffic, the added IRC traffic when other nodes are infected will appear anomalous. Finally, if the model is created or adapted to consider IRC traffic from particular IRC users to be normal, and even consider the set of IRC servers normally used to be normal, then comparing to the model will still effectively highlight the infected nodes.

3.4.2 Denial of Service Attacks

Figure 6a is an example of a network receiving an application level distributed DoS attack. This example is a web(port 80) distributed DoS attack. The attackers are

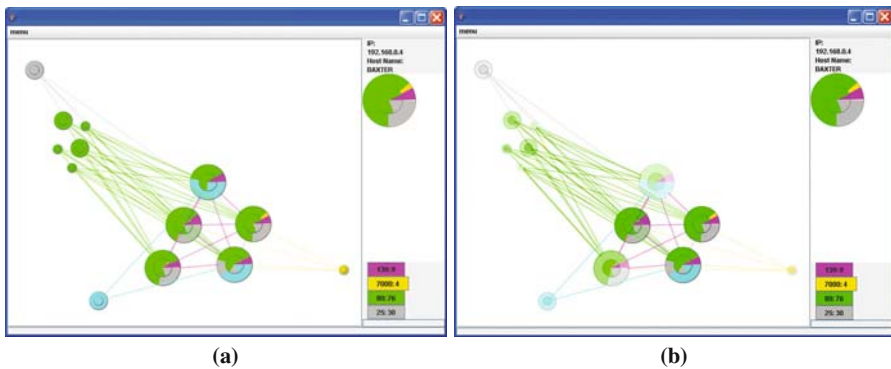


Fig. 5 A visualization of network infected by an IRC trojan. (a) Without comparing to a simple model. (b) Comparing to a simple model that detects abnormal behavior. Notice that the nodes emitting abnormal network behavior are darker

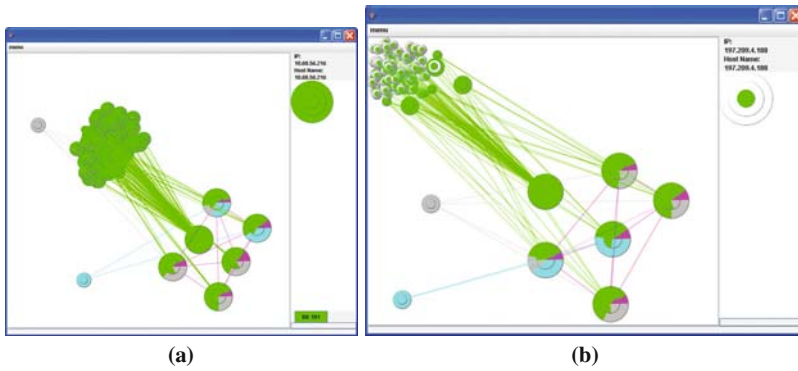


Fig. 6 Network undergoing heavy web traffic on its web server (a) This represents a Distributed DoS attack. Notice that in each timeslice, web traffic is still occurring from each of the nodes contacting the webserver represented by the large green glyph near the center. (b) This represents large amounts of web traffic from approximately 100 different clients in the time span of the visualization. Time slicing is used to show that the majority of these clients have activity in a single time slice and are not repeatedly creating connections like in a DoS attack

represented by the large cluster of green colored nodes all connecting to the same managed node. The larger size of the attacking nodes indicates a larger amount of network traffic. A comparison is difficult to find in this network because nearly all represented nodes experience high traffic volume. However, in the northwest region of the web traffic nodes cluster, there is a smaller traffic node in which only about half of the glyph is visible peeking out on the left side. This likely represents normal usage or a weak attacker in the DoS attack.

Time slicing is performed among the web client (or attacking) nodes to show that traffic occurs in each slice. By using time slicing, it makes it more apparent that this is a DoS attack and not a case of large legitimate web traffic. Small time slice intervals will improve the effectiveness of distinguishing between heavy usage and a DoS attack. Notice that web traffic occurs in every time slice of the attacking node, indicating constant web traffic over an extended period of time. Normal usage would have a colored inner ring, indicating the initial connection, but would likely see the traffic trail off toward the outer rings. The number of time slices colored would be proportional to the amount of time the client continues to browse the same web server. Of course, it is extremely unlikely that all clients continue to browse the same web server for even a small amount of time.

Figure 6b is an example of a network receiving large amounts of application level network traffic, but not necessarily a distributed DoS attack. In this network visualization, many web client connections are made to the web server (represented by the cluster of green nodes), which happens in a DoS attack. However, using time slicing, this visualization shows that most of the nodes do not continuously communicate with the web server. Most nodes are represented by a green inner circle, representing an initial connection and traffic to the web server, and clear outer rings. This indicates that after the initial connection and traffic occurs, the user no

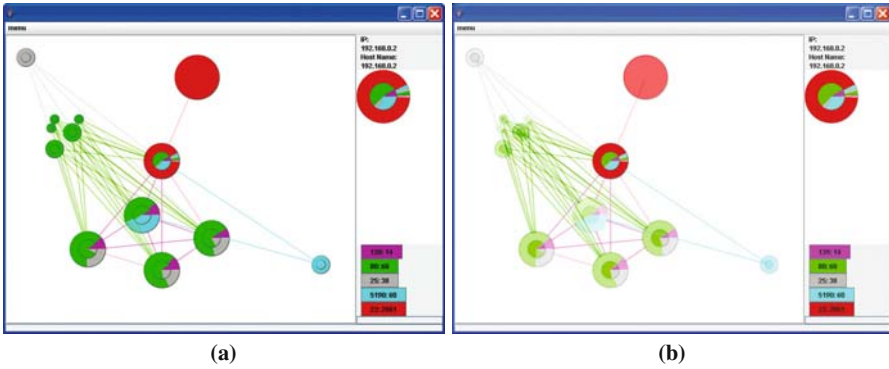


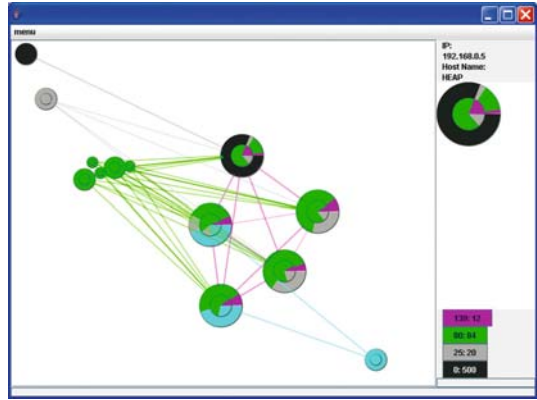
Fig. 7 A visualization of network under attack by an application level DoS attack against the SSH service. (a) Without comparing to a simple model. (b) Comparing to a simple model that detects abnormal behavior. Notice that only the nodes involved in the SSH DoS attack still have a strong opacity

longer browses the managed web server in this visualization. This is more typical of normal web usage than Fig. 6a as the user would likely come to a web server, obtain what they need, and move onto to another web server or stop browsing all together.

Figure 7 is an example of a network receiving an application level DoS attack against the ssh service. The large red node indicates the attacker and is large because of the amount of traffic present. Under normal network conditions, no node will become larger than a managed node. Baseline network traffic is also present in this visualization of normal college student traffic. Analyzing the managed node under attack, normal traffic patterns are present in the initial time slice (center) with a mix of web, chat, and file sharing traffic. In the most recent time slice (outer), normal traffic is present but heavily overcome by SSH traffic as indicated by the strong red color presence. In this particular network, the managed node is a normal network user with the SSH service open to allow remote logins to that machine. However, that service is under a DoS attack in this visualization, which is likely inhibiting that user’s ability to conduct normal network traffic and certainly inhibiting the use of the SSH remote login service. The SSH DoS attack network visualization is also compared to a normal usage network model and is easily able to identify the managed node under attack.

Figure 8 shows one of the managed nodes in the college student network getting ping flooded. Note that this visualization does not look much different from the SSH application level DoS attack. Underneath, the attack is similar, lots of network packets directed at the same network service. For this visualization, ICMP packets are regarded as port 0, since ICMP as a protocol does not have ports. Also notice that the attack occurred in the second time slice of the managed network. There is no black coloring, indicating ICMP traffic, in the first time slice represented by the center of ring in the managed node representation of the machine with the host name “HEAP”. The difference between this network attack and an application level network attack such as the SSH DoS attack above is that the ICMP layer is harder to

Fig. 8 Visualization of a student's machine under attack from an ping flood



defend. The main reason for this is that services such as SSH must be enabled by a user and are not active by default while ICMP services (such as ping) are generally enabled by default and often require changes to the operating system's kernel to disable or change behavior. However, ICMP services are often more lightweight than application level services, requiring less communication to perform their function, making it more difficult to use these services for a DoS attack. It is important for an administrator to see these attacks because although the user may not be able to disable such services, the network administrator can prevent malicious traffic from getting to the user at all.

3.4.3 Evaluation

This visualization was evaluated by surveying network administrators about effectiveness. The survey consisted of questions asking the network administrator to draw conclusions from the visualization. Network administrators of varying experience and ability were used in the survey to better evaluate who the visualization is useful for. The survey displays different visualizations of the same network with varying network conditions, including different types of network attacks. The administrator was asked to identify the anomaly by comparing the different visualizations of the same network. One of the choices we give is a network attack that this visualization technique does not support and is not visually present in our resulting figures. This evaluation was designed to address the feasibility of this visualization technique assisting network administrators in detecting security events but is not a formal validation of the effectiveness of the technique.

Five network administrators were able to identify which type of network attacks were taking place in the visualization with high accuracy. Even when they were incorrect, they were able to identify the features shown in the visualization and describe them. The survey questions were asked for the figure representing the normal network (Fig. 4) and each of the five analysts produce similar answers. Each

participant was easily able to identify managed nodes vs. unmanaged nodes. Each participant was also able to identify that two machines were using instant messaging traffic, four machines have sent email, and machines were accessing web servers. One participant thought it was odd that all of the students access the same set of web servers which is fairly unlikely and more a problem with my simulation. Although some analysts were skeptical of this being a normal network, when asked to choose between the choices listed above, each participant chose normal network. This image was shown to the participants first in order to give them some base knowledge of the network. All of the attacks occur on this same network, so this at least gives the participants of the survey some knowledge of the base network because a network administrator would certainly have knowledge of their network.

The SSH DoS attack (Fig. 7) produced consistent results among the participants of the survey. When asked to choose an identifier for this particular figure, all participants chose DoS attack and were able to identify it was against the SSH service. One of the participants noted that it could simply be a large file transfer over SSH, using a tool like sftp. Two other participants noted differences in instant messaging traffic over time, which was present, but was just a result of the randomness of the simulation. Two of the participants noted that the SSH traffic only occurred in the most recent time slice, although the other three participants were not explicitly asked about when the attack occurred. One participant was considering this to a possible session hijacking, but thought it out and chose DoS. This figure, especially compared with the normal network figure, provides an easily identifiable increase in traffic on the SSH service between two nodes and each participant was immediately able to identify it.

The compromised network (Fig. 5) produced good results among the participants of the survey. Each participant was able to identify light traffic occurring across port 7000, or IRC traffic. When comparing this model to the network, each participant tried to figure out the anomalous behavior in each of the darker nodes. Three of the five participants were able to quickly identify this network as network compromised by trojans that communicate via IRC. However, two of these administrators were aware that IRC is a commonly used protocol for trojans. The other two participants eliminated most of the choices listed (DoS, distributed DoS, etc.) but were unable to decide between normal network, compromised network with trojans, or session hijacking. However, these two participants, when asked, did not know that IRC is a commonly used protocol for trojans. One of the two participants that did not choose compromised network was wondering how much IRC traffic is a normal amount of IRC traffic for a particular host but assumed the small number of packets using the IRC service were normal and therefore chose normal network. The compromised network is one of the types of attacks that other visualizations do not focus upon and our method is designed to identify. With network security knowledge that IRC is a commonly used protocol for trojans, our participants were able to use this visualization to identify the attack.

The distributed DoS attack (Fig. 6a) was quickly and easily identified by each participant as a distributed DoS attack. The addition of multiple unmanaged nodes, each using the same service and grouped together, made this an easily identifiable

attack. Four of the five participants looked at the time slicing on the unmanaged web traffic nodes to conclude that this was a distributed DoS attack. Those participants were able to assume that continuous traffic over time slices from all unmanaged web traffic nodes would only occur in an attack. Two of the five participants were considering this to be normal traffic to a web server assuming that maybe the web server just got turned on or got linked from a popular news site but still chose distributed DoS when presented with the options.

The normal heavy web usage (Fig. 6b) took the most thought for each of the five participants. Each of their initial thoughts was a potential distributed DoS attack. Three of the five participants considered it normal network usage and the other two participants considered it a possible distributed DoS attack. One of the participants that considered this figure a DDoS attack said it was the end of a DDoS which occurred in the first time slice. The other participant that considered this a DDoS, said it was possible a quick DDoS which stopped, or just heavy usage. Each of the five participants viewed time slicing on the unmanaged web traffic nodes to identify that the amount of web traffic went to zero over time. One of the participants noted that some of the unmanaged web nodes produced constant web traffic over multiple time slices which was a result of the randomization of the simulation. Each participant was able to view time slicing to differentiate this figure from the web DDoS figure which is the goal of time slicing in this context.

The ping flood (Fig. 8) proved to be another figure where the attack was quickly and easily identifiable by each of the five participants. Each participant identified some type of DoS attack occurring over ICMP. Three of the five participants identified this as a “ping of death” or ping flood attack, which was the attack we attempted to simulate in this figure. All participants noted that this attack occurred in the most recent time slice against one of our managed nodes that was behaving normally in the previous time slice.

Each participant was asked after each of the different figures if using Ethereal, which is a common practice of a network security administrator, would help or be better than using our visualization to identify an attack. Four of the five participants felt that this visualization method makes it significantly easier to identify attacks. All five participants felt that using this visualization tool to identify a starting point for analyzing anomalous behavior and then further investigating using ethereal would be useful. One of the five participants preferred to use Ethereal setup with various filters for each type of attack to actively monitor a network, but noted that it would only be effective if the administrator was able to actively see everything in Ethereal meaning that the amount of traffic was low enough to be managed on a line by line basis by the administrator.

4 Future Work

In our comparison to network model section, we used an overly simplistic model to demonstrate the opacity effect. Our future work should include creating several models designed for defined networks to give a more realistic feel impressive of the

comparison to model section. Furthermore, using better models in the evaluation section will improve the conclusiveness of the results of the visualization technique based upon the model. Along with using more realistic data models, using real network data instead of simulated network data would improve the practicality of the evaluation section. Creating a good network model may also prove to be a difficult task for a network administrator. We will consider using machine learning techniques using user actions as attributes to attempt to build a network model based upon what the administrator does interactively. While our layout technique is effective in clustering nodes experiencing similar network behavior, we lose details due to occlusion as the node set gets larger. More work can be done using more interactive controls, grouping, and self-organizing maps to address this issue.

5 Conclusions

This approach aids a network administrator in identifying network attacks and intrusions. By visualizing the network from a service perspective, more specific types of attacks can be detected. The contribution of this research is to provide an application using a combination of existing visualization techniques applied a network traffic data set in order to better detect the type, severity, and presence of a network attack. Furthermore, this research visualizes more information than previous network traffic maps by approaching the data from a service oriented perspective and embedding multivariate information into the glyph representing a node. Based upon the results of our evaluation survey, network administrators feel that they gather more information in order to successfully detect attacks using this visualization technique. Finally, the temporal data present in the compound glyph via time slicing provides information for an administrator to distinguish between high volume cases and distributed DoS attacks.

There are some limitations to this approach including visual scalability and custom models. The comparison feature of this visualization is only available and effective when used with a good model of an existing network. In our results we use a very basic and simple model, which covers a variety of anomalous behavior, but does not get into enough detail about the normal activity of the network in order to be more useful. Basically, our model assumes all networks are the same, and things such as web and instant messaging chat should be the large majority of traffic. Such a model would not compare well with a cluster of SSH servers. Also, scalability to large networks is a limitation.

References

- Ball, R., Fink, G.A., North, C.: Home-centric visualization of network traffic for security administration. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 55–64. ACM Press, New York (2004)

- Becker, R.A., Eick, S.G., Wilks, A.R.: Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics* **1**(1), 16–28 (1995)
- Charles, P.: Jpcap: Network Packet Capture Facility for Java. <http://sourceforge.net/projects/jpcap> (2001)
- Conti, G., Abdullah, K.: Passive visual fingerprinting of network attack tools. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 45–54. ACM Press, New York (2004)
- Erbacher, R.F.: Glyph-based generic network visualization. In: *Proceedings of the SPIE '2002 Conference on Visualization and Data Analysis*, pp. 228–237 (2002)
- Girardin, L.: An eye on network intruder-administrator shootouts. In: *Proceedings of the Workshop on Intrusion Detection and Network Monitoring (ID'99)*, pp. 19–28. USENIX Association, Berkeley, CA, USA (1999)
- Goldring, T.: Scatter (and other) plots for visualizing user profiling data and network traffic. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 119–123. ACM Press, New York (2004)
- Komlodi, A., Goodall, J.R., Lutters, W.G.: An information visualization framework for intrusion detection. In: *CHI '04: CHI '04 extended abstracts on Human Factors in Computing Systems*, pp. 1743–1746. ACM Press, New York (2004)
- Lakkaraju, K., Bearavolu, R., Slagell, A., Yurcik, W., North, S.: Closing-the-loop in nvisionip: Integrating discovery and search in security visualizations. In: *VIZSEC '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, p. 9. IEEE Computer Society, Washington, DC, USA (2005)
- McPherson, J., Ma, K.L., Krystosk, P., Bartoletti, T., Christensen, M.: Portvis: a tool for port-based detection of security events. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and Data Mining for Computer Security*, pp. 73–81. ACM Press, New York, NY, USA (2004)
- Papadopoulos, C., Kyriakakis, C., Sawchuk, A., He, X.: Cyberseer: 3d audio-visual immersion for network security and management. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 90–98. ACM Press, New York (2004)
- Shanbhag, P., Rheingans, P., desJardins, M.: Temporal visualization of planning polygons for efficient partitioning of geo-spatial data. In: *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*, pp. 28–36. IEEE Computer Society, Washington, DC, USA (2005)
- Teoh, S.T., Ma, K.L., Wu, S.F., Jankun-Kelly, T.J.: Detecting flaws and intruders with visual data analysis. *IEEE Computer Graphics and Applications* **24**(5), 27–35 (2004)
- Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: Visflowconnect: netflow visualizations of link relationships for security situational awareness. In: *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pp. 26–34. ACM Press, New York (2004)

High Level Internet Scale Traffic Visualization Using Hilbert Curve Mapping

B. Irwin and N. Pilkington

Abstract A high level analysis tool was developed for aiding in the analysis of large volumes of network telescope traffic, and in particular the comparisons of data collected from multiple telescope sources. Providing a visual means for the evaluation of worm propagation algorithms has also been achieved. By using a Hilbert curve as a means of ordering points within the visualization space, the concept of nearness between numerically sequential network blocks was preserved. The design premise and initial results obtained using the tool developed are discussed, and a number of future extensions proposed.

1 Introduction

This paper describes the tool developed for providing high level Internet scale visualization of network traffic. The development of the tool was prompted by the need to have a tool which was able to display large volumes of IP traffic collected at network telescopes, while being able to communicate some kind of semantic and sequential relationship between the nodes representing networks displayed on the resultant plot. This work was to a large extent inspired by the work done by Randall Munroe, author of the xkcd.com web comic who published a “Map of the Internet” (Munroe, 2006a, b) based on aggregation of IP space by class A (/8 bit) in December 2006. This work was based on an interpretation of the IP address to Hilbert curve mapping algorithm that was used in his work, which was then extended for producing higher order and hence more fine grained curves. The real value of the application of the Hilbert mapping is that it preserves the locality of adjacent network blocks when the one-dimensional numerical ordering of octets is rendered to a two dimensional grid.

B. Irwin and N. Pilkington

Department of Computer Science, Rhodes University, Grahamstown, South Africa, e-mail: b.irwin@ru.ac.za, nicholas.pilkington@gmail.com

The pure information security aspects of this tool are not focused on in this paper, but rather the value of the technique for plotting IP Network data. Using this layout mechanism as a basis any number of colouring or other additional attributes can be added as an augmentation.

The remainder of the paper is organized as follows. A brief introduction to the Hilbert curve is presented as a prelude to the details of the implementation of the tool. The bulk of the paper is focused on sample output obtained and the interpretation thereof. Conclusions and reflection on planned extensions conclude the paper.

2 Related Work

The Hilbert curve is a continuous fractal curve, the limit of which fills a square. This was conceived by German mathematician David Hilbert in 1891 (Riemersma, 2006; Weisstein, 2007) and falls into a larger family of space filling curves including the Peano curve family. The Hilbert curve can be used to extrapolate data from one dimension into two dimensions while still maintaining properties of the original one dimensional data – particularly the notion of ordering and closeness to sequential nodes within the sequence. A Hilbert curve maintains locality of data on the curve. This means that data ordered a certain way in one dimension will still be ordered the same way along the curve in two dimensions. Another interesting property of the curve is that it visits every lattice point in a square with side length a power of two. This is especially useful in extrapolating data which occurs in powers of two, onto a plane.

The process of generating a Hilbert curve fractal can be formally expressed with a Lindenmayer production system as shown in Fig. 1.

A somewhat less formal, and possibly more intuitive, description of the operation of the curve can be presented graphically. The Hilbert curve is comprised of what are known as “cups” and “joins”. Cups are squares with one side open, and joins are straight lines that connect two cups. Each can be orientated in any way as long as they remain parallel to the cardinal axes. The simplest Hilbert curve is the first order one which fills a 2×2 square grid and it just a cup with the top side open (Fig. 2a). In order to generate the next (second) order curve – one replaces the existing cup

Alphabet : L, R
Constants : F, +, -
Axiom : L
Production rules:
 $L \rightarrow +RF - LFL - FR +$
 $R \rightarrow -LF + RFR + FL -$

Fig. 1 Lindenmayer representation of the Hilbert curve

Fig. 2 (a) Order 1 Hilbert curve. (b) Second order Hilbert curve showing cups

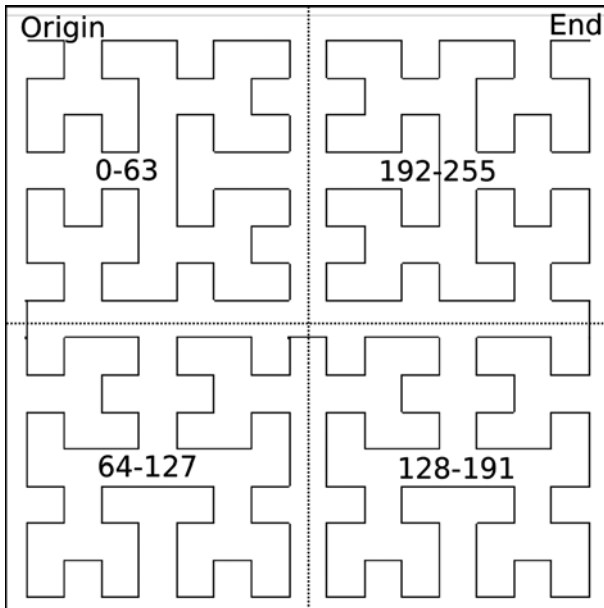
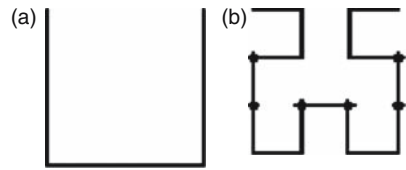


Fig. 3 Fourth order Hilbert curve showing plotting path for class A (/8) network blocks. 0/8 is mapped to the top left, and 255/8 to the top right corners, respectively

with four other cups connected together with joins as shown in Fig. 2b below. This process can then be repeated in the same way to generate all higher order curves as needed.

The Hilbert curves of order 4, 8, 12, and 16 are especially interesting as they have 256, 65536, 16,777,216 and 4,294,967,296 points, respectively. These values correspond to the natural grouping of Internet networks blocks by Class A (/8), Class B (/16), and Class C (/24) with the range of 32-bit Internet address space. A 16th order curve provides the same number of points as 2^{32} which is the same as the total potential number of addressable nodes on the IP protocol version 4 (IPv4) Internet.

Figure 3 shows the layout used for plotting class A network blocks using a fourth order curve. The curve starts with 0.0.0.0/8 mapped to the top left, and 255.0.0.0/8 to the top right corners, respectively. The grid can be divided into quadrants showing the placement of the 256 network blocks, which can be used as a guide when interpreting output such as that shown in the next sections. Raising the order of the

curve for /16 and /24 networks used the same layout, but increases the tightness of the curve within the respective quadrants.

It is worth noting that a similar pixel based plotting approach was proposed by Teoh et al. (2002), which uses a quadrant based mapping scheme based on the most significant bits of an IP address. The Hilbert plotting scheme has a more natural binning effect (to those habituated to dealing with traditional “classful” subnets) than the aforementioned method.

Two other projects are making use of the same Hilbert Layout for performing similar work (in both cases also inspired by Munroe’s work). The first is the ANT Censuses of the Internet Address Space (Heidemann and Pradkin, 2007b), using the curve for plotting Address space usage as part of the Internet Census project (Heidemann et al., 2007a). The second related project is that being run by Measurement Factory using this to visualize BGP route data (Wessels and Claffy, 2007). In many ways similar to the original goals and purpose of the system evaluated in Teoh et al.

3 Technical Approach

A proof of concept implementation of the Hilbert curve based layout system was developed in C++ and OpenGL. While development and testing was on the Microsoft Windows® platform, it should be portable to any system POSIX and OpenGL compliant system. Other implementations have been found in php (Bosci, 2007), but follow a similar system. The base layout code was developed to map a given substring of a dotted-quad IP address representation to a particular point on the grid being produced. In effect this associated a bin or bucket holding IP networks of a given ordinarily (representing natural 8-bit divisions of the dotted-quad, and hence clustered by most significant octets) to nodes on the curve.

As previously discussed, curves of orders 4, 8 and 12 map easily to the natural netmasks of pre-CIDR class A, B and C network blocks. Hilbert curves of 16th order were not implemented due to the complexities of mapping these to a reasonable screen size, and memory constraints. While it is recognized that much Internet address space allocation today does make use of CIDR based variable length masks, for most purposes, the traditional masks are still useful. The net result is a graphical output showing nodes which are coloured as containing elements aggregated dependant on the order of the curve. Initially a true/false flagging system was used where a node was drawn if it contained at least one member in the aggregation bucket. This was extended to use colouring based on criteria such as unique hosts with the bucket, and total number of packets within the bucket.

A number of discrete implementations were produced, each focusing on separate aspects of the visualization, but all using the same underlying Hilbert curve generation for layout. Specifically versions were produced that provided colour-indexed quantifications of the number of unique hosts within a specific network block bucket, and the number of packets received within a bucket.

In the program the vertices were generated recursively using the Lindenmayer System representation of the Hilbert – this was the simplest representation and allowed the curve to be generated quickly and accurately. Additionally, the use of this representation also generates the vertices of the curve in numerical order which allows points to be plotted at the same time the curve is generated thus providing much greater efficiency.

The IP addresses to be plotted were stored in a hash table that allowed the required IP address count associated with the current vertex on the Hilbert curve to be quickly recovered from the table and plotted.

Input to the system was by means of simple text files containing a single dotted-quad IP address per line. This format was chosen due to the simplicity to produce form a number of different data sources, such as *libpcap* format packet captures, databases (containing Intrusion detection data), and simulation software. The code could also be trivially extended to work directly with formats such as *libpcap*.

4 Results

Initial results presented by the tool have been promising with tests of up to 63 million individual addresses being rendered with relatively modest hardware requirements by modern measures. The test system used was a dual Xeon 2.2 Ghz with 4 Gigs of DDR2 RAM running Microsoft Windows Server 2003, and an Nvidia Gforce 5200.

Performance was found to remarkably good on the test system with the exception of the colour mapping based on total packet count which was found to be particularly resource intensive, requiring over 1,800 MB of RAM in which to store the state tables, and needing 60 min to render the 63 million addresses on an order 12 curve (equivalent to /24 network bins each holding 256 individual IP addresses). It is anticipated that this can be improved, as zooming and other navigation became near impossible at this level of performance.

The simple flagging and host count based colour mapping were found to perform, responsively, and required in the order of 300 MB of memory to hold the large 63 million address dataset. Smaller datasets required on average 100–150 MB of RAM. Sample images of the tool output should be interpreted bearing the layout show in Fig. 3 in mind as a guide to the relative addresses of nodes within the overall address space.

Generating an overview at class A level provides a quick overview of input data, and particularly allows for sanity checking that data is not being plotted in regions where it is not. Figures 4–6 show the plot of 2.4 million unique data points sampled from CAIDA network Telescope data between 12h00 and 17h00 EST on February 28th 2007 (Shannon et al., 2007). Figure 4 is the fourth order curve with the actual Hilbert path plotted as a guideline. The same data is shown in Figs. 5 and 6 but plotted onto an eighth order curve showing data bins with netmasks of /16 (thus holding 2^{16} IP addresses) and a 12th (/24 bins) order curves, respectively.

Fig. 4 Plot of the fourth order showing background curve. Colouring is based on number of distinct nodes within the bucket

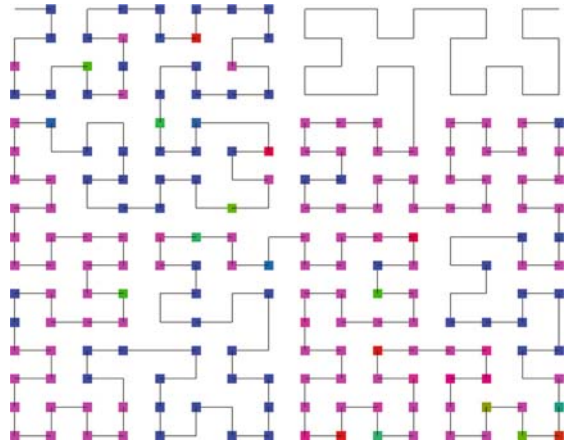


Fig. 5 Plot of eighth order representation of the data in Fig. 4, showing buckets corresponding to /16 networks (Class B)

The increase in granularity can be seen, particularly in the sequences show in Fig. 7 and discussed in the following section.

The above images illustrate the value of increased resolution, which can then be further analysed.



Fig. 6 Plot of data on a 12th order curve. Points correspond to buckets of size/24 (Class C). The boxed area relates to the zone of discussion in Fig.7, and represents the network ranges from 23.232.0.0/13 to 24.170.0.0/16

Fig. 7 Close up of the view of the 23.232.0.0/13 and 24.0.0.0/8 networks using /16 buckets (A – left), and then with a finer resolution of /24 buckets in B (right) showing a distinctly different pattern. This area is shaded in Fig. 6



4.1 Output Analysis

Once traffic plots have been created, further analysis of the resultant images can be performed. Figure 7 shows close up of the view of the 23.232.0.0/13 and 24.0.0.0/8 networks using /16 buckets, and then with a finer resolution of /24 buckets in Fig. 7b showing a distinctly different pattern, with a number of networks having no traffic originating from them. These ranges are particularly interesting to analyse as the 24/8 network has been used widely for the provision of broadband cable access in the Americas, with large portions belonging to providers such as Comcast, Charter Communications and RoadRunner – major players in the provision of home broadband connectivity.

Portions of the netblock are also allocated to LACNIC where they appear to be sub-allocated to similar sorts of companies, but these do not show up as having been sources within the traffic set used. Data sources networks range from 23.323.0.0/16 to 24.170.0.0/16. This range is also indicated by a grey box on Fig. 6. Interestingly IANA lists the 23.0.0.0/8 netblock as currently reserved, so one can speculate as to the origins of this traffic – possibly misconfigured hosts as the range appears in a number of vendors documentation as example addresses.

One of the strengths of the plotting scheme developed is the aggregation of nearby networks as shown in Fig. 7. This allows for immediate visual recognition of the logical and special relationship between these nodes far more easily than some traditional grid based and wrapped linear plotting methods.

Analysis of the data can be performed either by means of working with the resultant image, or within the application which provides a rudimentary heads up display type information where feedback is provided as to the actual network address of the node over which the mouse cursor is placed.

Using the colour-index based mappings, as a means of shading graph nodes, further information relating to the plotted nodes can be conveyed. Figure 8 gives an example of 63 million events being blotted on a eighth order curve, with colouring indicating the number of unique hosts in the particular block that have originated traffic (or at least purported to based on IP datagram source addresses) to the network telescope. The graduation runs from green through blue orange and finally red. The insert within Fig. 8 shows a zoomed view of the 210.0.0.0–221.0.0.0 network ranges. The area of the image occupied by this insert is traditional Class D (224.0.0.0/4) and E (240.0.0.0/4) address space, which are, respectively, used for Multicast and reserved.

4.2 Other Applications

Following the initial work done with the curve the authors applied it to the visualisation of two other related problem areas. The first being to allow for a visual comparison and evaluation of the effectiveness of network telescopes of differing sizes. The results for this can be seen in Fig. 9, where the general shape of the plot produced when only using data from a single address still maintains the general shape of that produced when using data from the entire Class A telescope. This shows that smaller telescope still have some value, when used over long temporal baselines.

The second area in which it was applied was in evaluating the impact and effectiveness of differing network worm propagation algorithms. In the case study shown below the Hilbert curve is used to map snapshots of activity from a simulated Blaster worm at iterations 5,000, 7,500 and 10,000 within its propagation cycle. From these images it can be easily seen that while the propagation has covered 96% of /8 network bins (Fig. 10c), the coverage at /16 (Fig. 11) and particularly /24 sizing is very sparse.

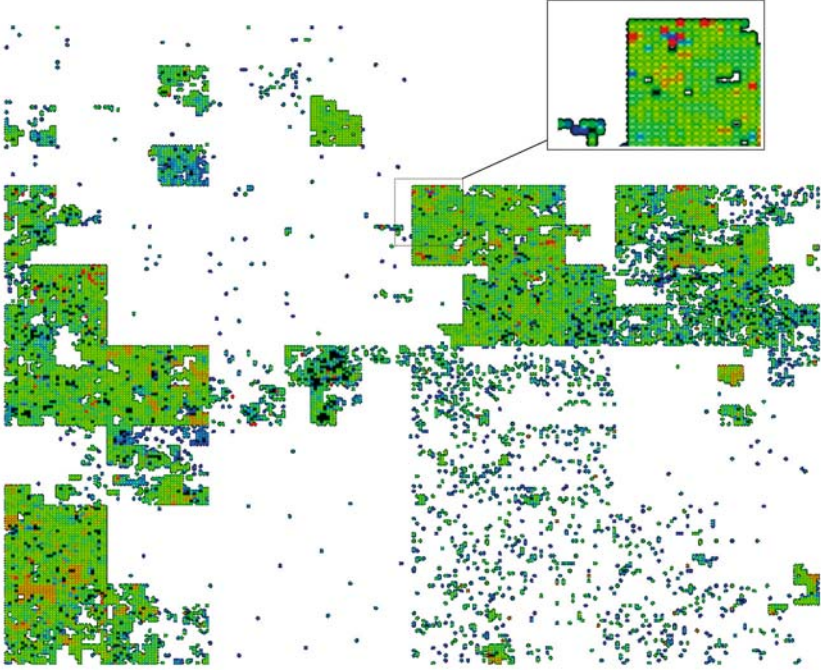


Fig. 8 63 million events plotted using host based colouring. Red dots show networks with the greatest number of unique hosts

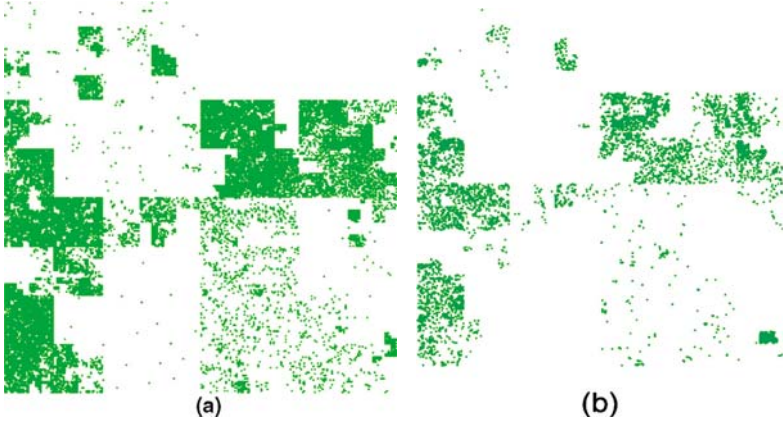


Fig. 9 Comparative mappings of (a) Class A telescope (b) data collected by a single IP address within the telescope

The ability to use images such as those presented above allows for researches to perform rapid visual evaluations and comparison, on the basis of which further more detailed statistical analysis may be performed.

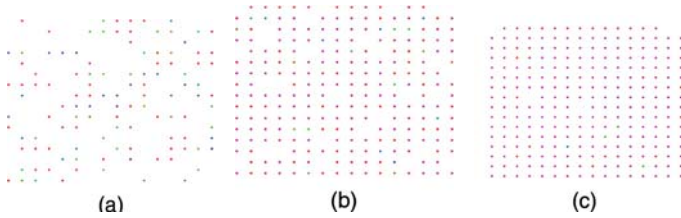
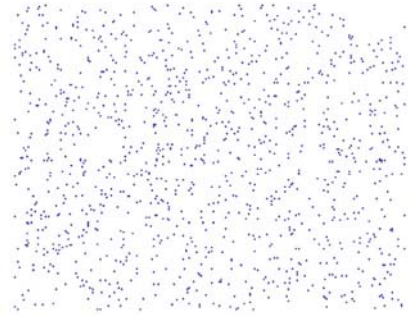


Fig. 10 Progressive coverage of IPv4 space by Blaster. /8 bit bins used for aggregation, and plotted on a fourth order curve. From left showing coverage at 5,000 (a), 7,500 (b) and 10,000 (c) iterations of the scanning and propagation algorithm

Fig. 11 Coverage of IPv4 space by Blaster after 10,000 iterations of propagation algorithm. Blotted using eighth order curve and bins of size /16. Despite near full coverage of the space with larger bins, the coverage is fairly low at this level of detail



5 Future Work

The authors have anticipated a number of extensions to the tool now that the value of the plotting scheme has been established. A more in-depth usability study needs to be completed, and aspects highlighted there will need to be addressed. From initial informal evaluation and operational use by the authors and researchers within the department, the following items are envisaged as being useful extensions and additions to the toolset:

- IPv6 support – with the move to IP version 6 the curve should scale, provided sufficient memory is available on the host system. Curves of order 32 and above would need to be used.
- Heads-up display capability – in order to provide for improved navigation of the curve, a HUD mode needs to be added, which could also provide further information relating to the specific nodes within the network.
- Overlay mode – allow network block of interest to be defined and input to the application which can then display a toggle-able highlight or shadow mask to show these when navigating. Network blocks defined as Bogons or reserved blocks are likely to be a useful set which to highlight traffic originating from.
- Time sequence views – to add accelerated and retarded time playback of *libp-cap* files with the resultant output being captured as frame sequences or video

for analysis. This would illustrate changes in source traffic over time. Currently this has to be implemented through external processing, and only allows for snapshots.

- A Geographical mapping where colour mapping can be performed based on continent, country or regional registry. This may work best for highlighting specific countries rather than as a general mapping mechanism where nearly 200 colours would prove difficult to discern and differentiate, particularly at the /24 level.

The implementation of these features and a more detailed usability study is planned for the forthcoming academic year.

6 Conclusions

Since its initial inception the tool implementing the plotting scheme discussed has been used for detailed analysis of network telescope traffic, in particular to perform quick visual comparisons between data collected on the authors' own network telescope (comprising a single /24 network under the auspices of AFRINIC) and data provided by CAIDA.org (Shannon et al., 2007) from their large telescope located in the United States comprising a class A network – 65,535 times larger than that of the authors! The outputs generated using the Hilbert plotting scheme have shown that the same address ranges are being seen as the sources of both backscatter and malicious traffic, with minor variations between the two networks.

Another use to which the tool has been put is to visualise the output of simulated worm scanning activity in order to assess the effectiveness of various scanning algorithms and defensive measures such as strike back, and counter worms. The resultant images have allowed for quick visual evaluation of datasets ranging in the tens of millions discrete dataset members.

The tool developed has proved useful in providing high level overviews of large volumes of traffic while still maintaining the sequence order of the input data.

From a pure Information Security perspective this tool allows for easily interpreted visual summative reports to be generated from very large volumes of network traffic. While the proof of concept tool developed has noted shortcomings, the actual principle of using the Hilbert curve, or possibly other space filling fractal curves for plotting IP traffic data is a solution portable to other visualisation applications and problem spaces. The use of such visual reporting allows for easier interpretation of data at executive level, and in high workload environments.

Acknowledgements This work was performed in and funded by the Centre of Excellence in Distributed Multimedia at Rhodes University with financial support from Telkom SA, Business Connexion, Converse, Verso Technologies, Tellabs, StorTech, Mars Technologies, Amatola Telecom, Bright Idea Projects 39 and THRIP. Randall Munroe of XKCD.com provided the initial inspiration for this work.

References

- Bosci, H. Map-o-net.com image drawing code. May 2007. <http://map-o-net.com/internet.html>
- Heidemann, J., Pradkin, Y., Govindan, R., Papadopoulos, C., and Bannister, J. Exploring Visible Internet Hosts through Census and Survey. Technical Report ISI-TR-2007-640, USC/Information Sciences Institute, May, 2007a
- Heidemann, J. and Pradkin, Y. Mapping the Internet Address Space (poster). USC/Information Sciences Institute. <http://www.isi.edu/ant/address/>, August 2007b
- Munroe, R. Map of the Internet. XKCD.COM <http://xkcd.com/195/>. http://imgs.xkcd.com/comics/map_of_the_internet.jpg, August 2006a
- Munroe, R. Map of the Internet. 11 December 2006b. <http://blog.xkcd.com/2006/12/11/the-map-of-the-internet/>
- Riemersma, T. The Hilbert Curve, August 2006. <http://www.compuphase.com/hilbert.htm>
- Shannon, C., Moore, D., and Abden, E., The CAIDA Backscatter-2007 Dataset February 2007–November 2007. http://www.caida.org/data/passive/backscatter_2007_dataset.xml
- Teoh, S.T., Ma, K.-L., Wu, S.F., and Zhao, X. Case study: interactive visualization for internet security. In Proceedings of VIS '02: Proceedings of the Conference on Visualization '02. IEEE Computer Society, 2002
- Weisstein, E.W. Hilbert Curve. From MathWorld – A Wolfram Web Resource. <http://mathworld.wolfram.com/HilbertCurve.html>, 2007
- Wessels, D. and Claffy, K., IPv4 Heatmaps: BGP Route Advertisements. <http://maps.measurement-factory.com/gallery/Routeviews/2007>

VisAlert: From Idea to Product

S. Foresti and J. Agutter

Abstract Visalert is a visualization system designed to increase the monitoring and correlation capabilities of computer network analysts engaged in intrusion detection and prevention. VisAlert facilitates and promotes situational awareness in complex network environments by providing the user with a holistic view of network security to aid in the detection of sophisticated and malicious activities, and ability to zoom in-out information of interest. The system provides a mechanism to access data from multiple databases, and to correlate *who*, *what*, *when* and *where*. This chapter describes the design process that enabled the team to go from the conception of rough visual sketches to the implementation and deployment of a finished software. In addition, the chapter describes the issues that the interdisciplinary team had to address to carry the project from idea to product.

1 Introduction

This chapter describes the interactive development process to design a visualization system for computer network security. We want to focus on the design phase of the visualization and show how initial concepts were developed with lessons learned from these concepts. In addition, we wish to illuminate the design choices along the development path that ultimately led to a successful visualization paradigm. First, we will discuss the different people and roles of the interdisciplinary team. Second, we will describe the different design sketches that were developed. Third, we will describe the transformation of these pen–paper based sketches into a refined computer visualization scheme. Finally, we will briefly discuss the move from final static prototype to implementation.

S. Foresti and J. Agutter
University of Utah and Intellivis, Inc. e-mail: stefano@intellivis.com, jim@intellivis.com

1.1 The Project and Team

The project was started by the CROMDI team at the University of Utah. We were awarded a grant by the Intelligence Community (ARDA-DTO-IARPA) to research novel visualizations that would aid network analysts in detecting cyber-anomalies, with particular interest for stealthy attacks that are diluted in time, and very hard to detect.

The Center for the Representation of Multi-Dimensional Information is a Utah State Center of Excellence that performs science, R&D and commercialization of user centered interactive displays. Our team uses an interdisciplinary methodology that integrates cognitive psychology, visual design, computational and visualization methods, and knowledge from domain experts.

CROMDI had previously developed novel displays from idea to commercialization, in medicine. Prior to this project the team had no specific experience in network security: this enabled the team to approach the project by “thinking outside of the box” and come up with a novel visualization method.

1.2 The VisAlert Metaphor

VisAlert enables correlation of heterogeneous network data, leveraging the fact that all events possess what we term the “ W^3 ” premise: *When*, *Where*, and *What* attributes:

- *When* refers to the point in time when the event happened.
- *Where* refers to the network node, e.g., an IP address, to which the event pertains.
- *What* refers to some indication of the type of the event, e.g., \$log = snort, gid = 1, sid = 103\$.

The visual layout, as shown in Fig. 1 maps the *Where* of an alert into the center of the circle. This is represented via a topology map of the network under scrutiny. The *What* of an alert instance is mapped to the different sections of the outside circular element. The *When* of an alert instance is mapped to the radial sections of the circle moving from most recent (closest to the topology map) to the past as it radiates outward. Alert instances are visualized as lines from the alert type on the outer ring, to the node location in the inner circle. The *When* space is divided into history periods, that show the number of alert instances that occurred in them, while the alert instance lines are only shown for a selected history period only.

Additional visual indicators encode information to increase the situational awareness of the user:

- The icon size increases when nodes experience several alerts. The assumption is that a node that is experiencing multiple unique alerts has a higher probability of malicious activity than one experiencing only one alert. The size will make it stand out and focus the user’s attention so he or she can take action.

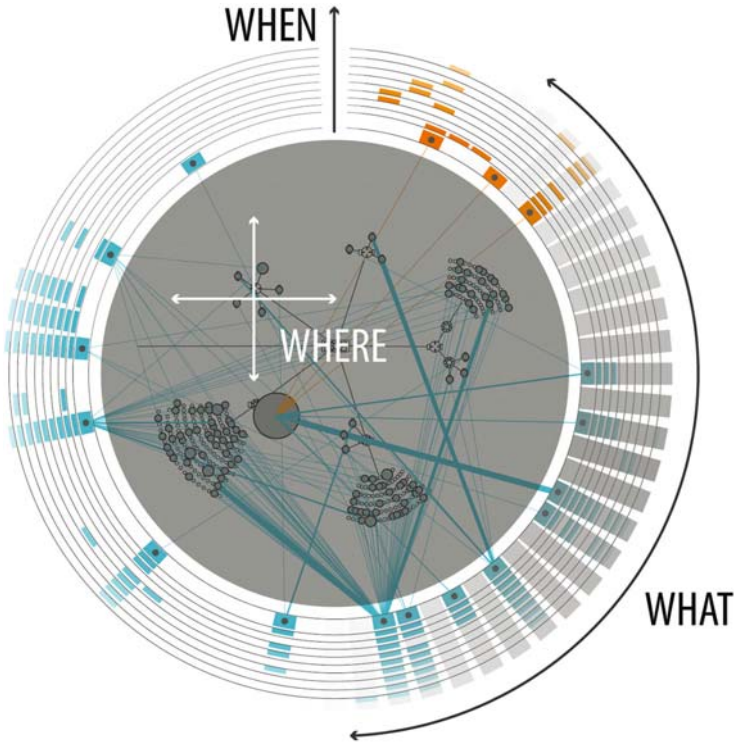


Fig. 1 An event is represented by a line connecting an alert type (*What*) at time (*When*) to a resource (*Where*). VisAlert here exhibits multiple alerts and relevant visual indicators to analysts: alert type via color coding, larger node size with larger number of different alert types, and larger beam size for persistence of the same problem

- The alert beams encode persistence of a particular problem. If a large number of the same alerts are triggered on a particular node over an interval of time the line changes in thickness to show how many alerts. In this manner, continual or recurring problems become evident very quickly, enabling the user to take swift action.
- Color is used to determine user selected ranges and severity levels.

For more detail about the visualization scheme refer to Foresti et al. (2006).

2 Related Work

2.1 Visualization of Network Security

The growing community of VizSec indicates the relevance and number of problems that need to be tackled in network security, where visualization is seen as a potential

solution or aid. Many of these techniques have been demonstrated to be effective at allowing users to see malicious activities such as worm or DoS attacks (Teoh et al., 2002).

One general theme that can be found in the previous work of visualization of security is that it was driven by the identification of specific problems (Teoh et al., 2003) or the monitoring of specific data types (Cox et al., 1996; Estrin et al., 2000). Much research has been done on visual techniques to improve visual pattern matching (D'Amico and Larkin, 2001), and to improve recognition of known events or levels of threat (Polla et al., 1998).

Traditional representations and network alert reporting techniques tend to use a single sensor – single indicator (SSSI) display paradigm. Each sensor has a unique way of representing its information (indicator) and does not depend on information gathered by other sensors. The benefit of such an approach lies in the separation of the various sensors. Each sensor's indicator can thus be optimized for the particular data produced by its sensor, and the user can pick and choose which sensors to use in an analysis. Furthermore, the failure of one sensor does not impact the capability of the rest of the system.

Consequently, the separation between the various sensors is also the weakness of this representation technique. Because each indicator is isolated, the user must observe, condense, and integrate information generated by the independent sensors across the entire enterprise. This process of sequential, piecemeal data gathering makes it difficult to develop a coherent, real-time understanding of the interrelationship between the information being displayed.

2.2 Design

The discipline of design has collected a comprehensive knowledge base of the nature, methods, and value of basic 2D and 3D design and their relationship to human collective and individual psychology and behavior. This knowledge base consists of basic principles (e.g., scale, shape, rhythm, color, structure), elements (e.g., line, figures, objects, space) and organizational rules (e.g., hierarchy, layering, symmetry) (Arnheim, 1977; Bogdan, 2002; Wong, 1972, 1977).

Our team determined that information visualization tools are more effective for decision making if developed with an iterative design process that permits simultaneous attention to multiple perspectives, skills and knowledge-bases. We also found that the *design process* allowed for a spontaneous and natural way of socially engaging a wide range of disciplines and individuals working in a very difficult problem. This is in line with existing knowledge that the *design studio model* in general and the design process in particular are a successful working laboratory and methodology for addressing open-ended, fuzzy, and multivariable problems (Cross, 1982; Rowe, 1987).

2.3 *Inter-Disciplinary Collaboration*

Collaborative success is ultimately grounded in the careful structuring of a team's group dynamics, which are based on clear roles, respect, trust, values, shared goals, and a common language (Friedman, 1997). Accommodating different methods, techniques, positions, interests, standards, languages, perspectives, knowledge, expectations of people from different disciplines takes considerable time and effort, as one has to overcome prejudices each field has of the others (Kraut et al., 1988).

Our team determined that the production of technology that meets users' needs requires the involvement of several roles and perspectives, and that trust among different disciplines is both essential to success, and becomes stronger with the demonstrated value of the work.

3 Technical Approach

3.1 *The Team Dynamics*

The CROMDI team utilizes an iterative interdisciplinary process to design, a built-in evaluation process to verify its design output, and a business approach to meet customer needs. The lifecycle of the project included several dozens of people at different levels of involvement and periods in time.

We can identify the following teams or roles, each one addressing the problem from their specialty but in direct collaboration with others according to needs. The Design Team establishes the overall rhythm of the process, and interacts with all other teams at different times in a modality similar to that of the traditional design studio. Following is a description of the roles and their tasks involved in the design process:

1. The *Client* is the actual organization or user asking/supporting the development of a new data representation solution to a particular information problem.
2. The *Application Team* takes the role of the specialist and works as middle-person between the Client and the Design and Psychology Teams (3 and 4). This team "translates" the client needs and requirements into programmatic needs. This team also works as critic and adviser to our research group at large during the design process. Application teams have been in Medicine, Finance, and Network Security.
3. The *Design Team* is in charge of developing the data representation scheme following a collaborative design process and using special principles and techniques discussed elsewhere (Bermudez et al., 2000). During the initial phases, the design team works very closely with (2) and (4). As the schemes become more final, the design team begins to have direct contacts with (1) and with (5).
4. The *Psychology Team* extracts the mental model experts in the field of application use to make sense and act upon the data. During the initial phase, (4) works

in close relationship with (2) to study (1). In later phases, this team is heavily involved in the evaluation of the representation schemes developed by (3) and implemented by (5).

5. The *Computing Team* addresses algorithms, software development, and distributed computing implementation. This team is particularly involved during the final phases of a project. At that time, it works closely with (3) and receives advice from (2) and (4). Its initial input consists of providing prescriptive and “budgetary” advice regarding actual computer implementations of the data display. In areas of application relating to computing (e.g., networking monitoring), this team may also become (2).
6. The *Administration Team* is focused in supporting the day-to-day operation of our research group as well as seeking new areas of work, recruiting consultants and collaborators, managing intellectual property, etc.
7. *Consultants*: external reviewers enter the process at critical times to evaluate the ongoing results and provide an unbiased review of our team’s design effort.

3.2 The Design Process

The interactive work of all these people and disciplines occurs within an over reaching ideology of design as a function of human needs and behavior and the interaction between operator and display. As a result, the design process follows the concept of a “hermeneutic circle” (Snodgrass and Coyne, 1990). This concept is an iterative process of implementing a design, learning and understanding from discussion and feedback from the targeted users, and subsequent design refinement. First, the problem and the metaphors for the information that will be displayed are defined. Next, an iterative process via “dialogical exchanges” is used to gain additional insight into the design. New interpretations are discovered and the design is refined. The design development process contains a second feedback loop of iterative evaluation for design usability and intuitiveness.

Each design refinement is evaluated using a testing protocol. The results of these evaluations are methodically analyzed to elucidate design changes while minimizing designer bias. This process also minimizes alterations to the requirements and the design, late in the display designs lifecycle, when changes are more costly (e.g., a change during the design phase is less costly than a change after the display has been deployed). This methodology is successful because the design is evaluated and redesigned during each phase of development, with the intent that the majority of design changes occur in the early stages of development.

The development methodology of the VisAlert display system followed an iterative design process that allowed for many possible design solutions to be explored. The team first developed numerous sketches. These sketches were then refined and mad into conceptual computer based display concepts. These were then iteratively refined and developed until a final solution was achieved. Following the completion of the design phase, an iterative refinement period was conducted with end users. This ensured that the final solution would fit with the uses needs.

3.3 Sketches

Figure 2 shows an early sketch representing the idea to collapse many variables into one as a primary indicator. The position in the y dimension shows the amount of the deviation from normal both for the primary indicator, and the variables combined to create the primary indicator. This concept helped the team think about nesting of variables and the hierarchical representation of network data to create primary indicators from a series of disparate variables.

The sketch in Fig. 3 explored a metaphoric representation idea, based upon a large group of environmental formations such as a field of flowers. When flowers bloom they can be seen standing out of the rest of the group because they are visually unique and distinct. This follows general gestalt principles found in design. The design concept further explores different types of unique flower types to encode additional information.

The sketch in Fig. 4 explored time evolution of variables, which are represented by the y dimension's height. In addition, items could be placed on a series of quadrant grids that are subdivided to represent problem severity and problem relevance. Glyphs could represent different graphic primitives to indicate type of data or variable presented.

The sketch in Fig. 5 examined the idea of the “inside” protected by a firewall surrounding the enclave from the Internet “outside”. The multiple paths through the firewall were representative of unauthorized breaches in the firewall and the

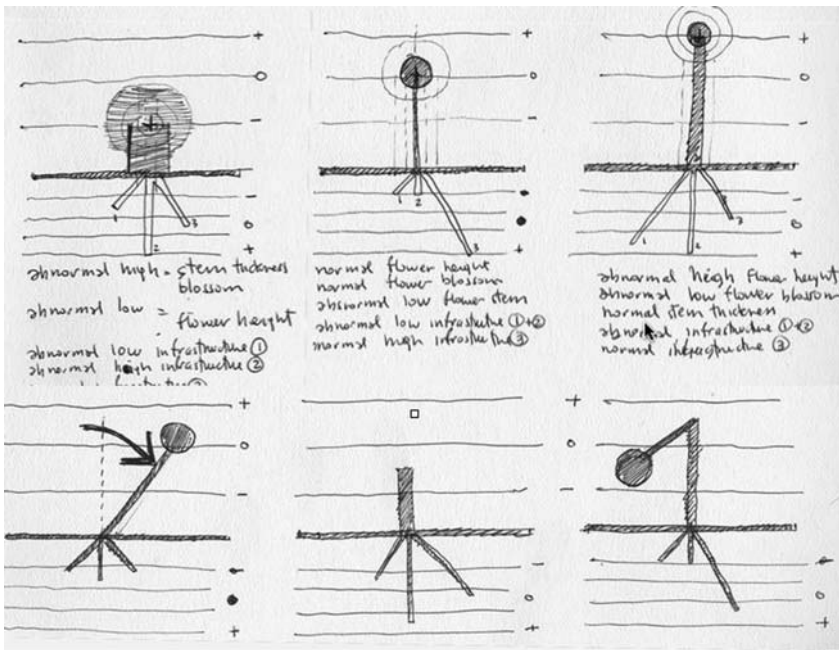


Fig. 2 Stem sketch

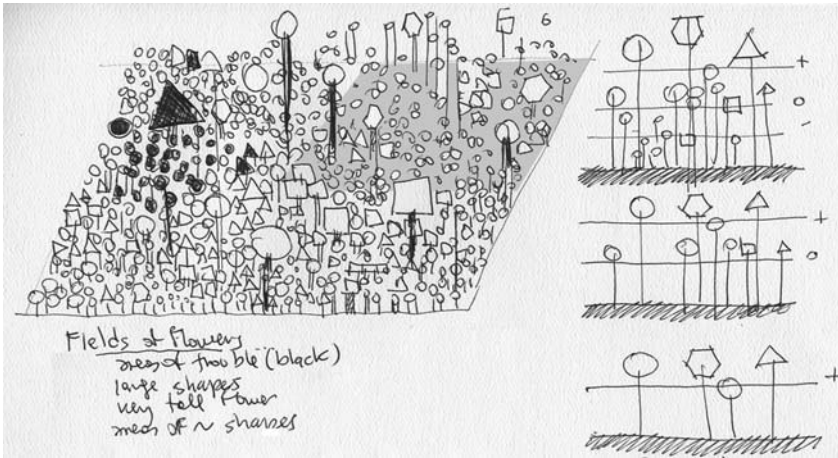


Fig. 3 “Field of Flowers” sketch

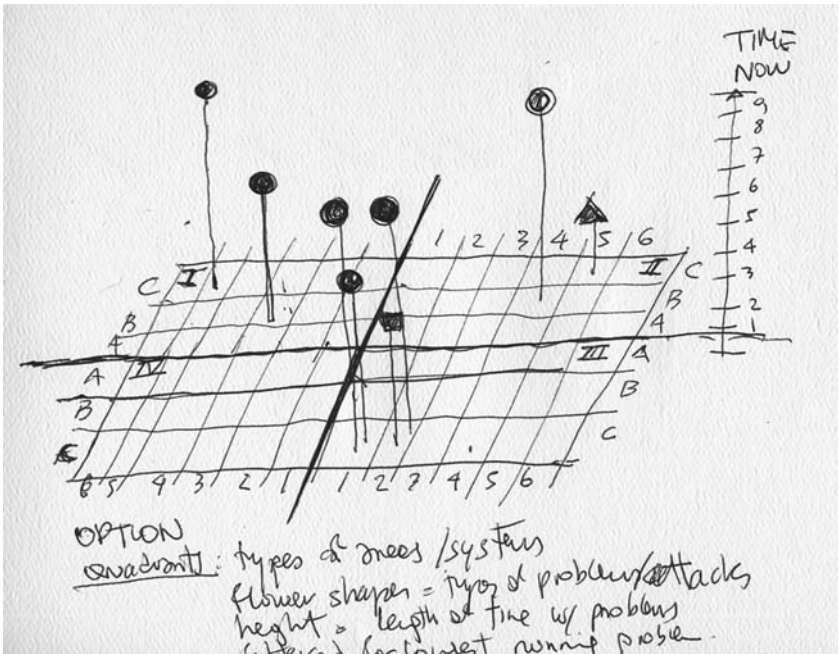


Fig. 4 Quadrants sketch

compromised machines. This sketch was the beginning of a network typology or the representation of “where” to provide users with a context of network events.

The concept in Fig. 6 explored the idea of a firewall through a literal representation of walls and increased levels of security as you move from outside to inside. It was thought that the most valuable assets would be placed in the center section, and

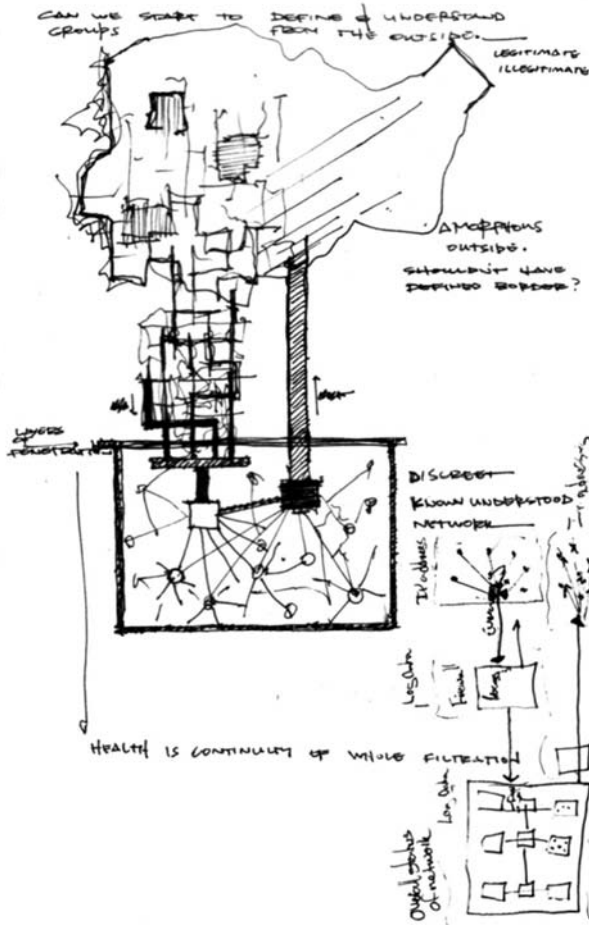


Fig. 5 In-and-out sketch

granted only selected access. Unauthorized breaches in the firewall could be shown as breaks in the wall.

3.4 Refined Conceptual Ideas

After the sketches had been discussed and revised, we then proceeded to move some of the more promising ideas into a more refined and computer-based representation. During this step we examined issues of scaling, usage of color, interfaces, and data handling.

Building upon the idea of the firewall, we developed a multi-dimensional icon that could map 3 variables (Fig. 7). These information bricks or blocks could be then

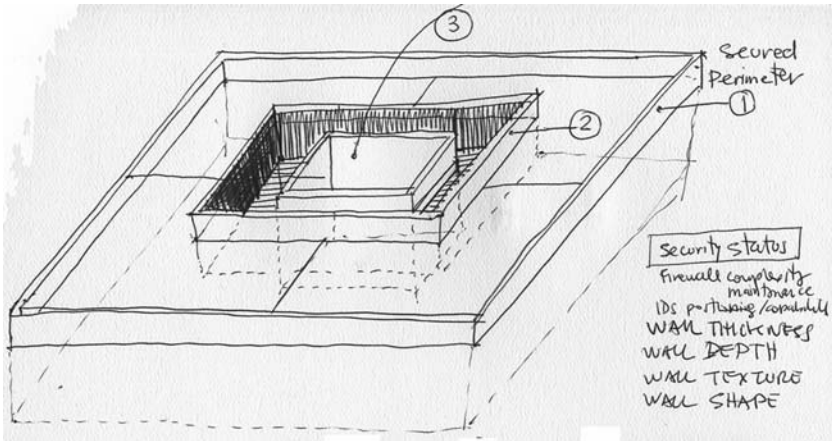


Fig. 6 Firewall sketch

arranged together to create an information structure. The blocks could be grouped together to show information that would be functionally related, such as information about all servers grouped together. The objective was to aid network administrators to quickly identify problems in specific portions of their network.

These icons could then be grouped together scaled to show more information. The images in Figs. 8 and 9 show how patterns of particular problems would emerge, and then allow the network operator to drill down by zooming in. This concept also explored what the interface might look like and what additional tools would be needed, such as filtering and zooming capabilities.

The next evolution in Fig. 10 was to develop the idea of the firewall icon as a larger representation and make it a placeholder for network topology and organization, thus combining two of the previous ideas. This combined the *where* and *what* of the display, even if limited to representing only firewall data through a metaphor.

The next step (Fig. 11) was to develop the sketch into a refined image to see what a complete display might look like. In this step we included different information around the four sides of the display such as firewall logs and alerts. In addition, the network topology was further developed. The concept of correlation between network alerts made its first appearance, as evidenced by the red connector lines to a particular machine, which indicates that a particular machine had experienced three different types of alerts.

A further refinement, as shown in Fig. 12, introduced the idea of time represented as radiating rings from the center. This enabled to include the *when* of an event, and sparked the idea of the representation of *what*, *where*, and *when* of network events. The last refinement was to create a radial representation so we could include many different types of alerts that could be functionally grouped.

The final iteration of the design was to include a network topology in the center and refine the look and feel of the concept, resulting in the VisAlert concept as described in the introduction.

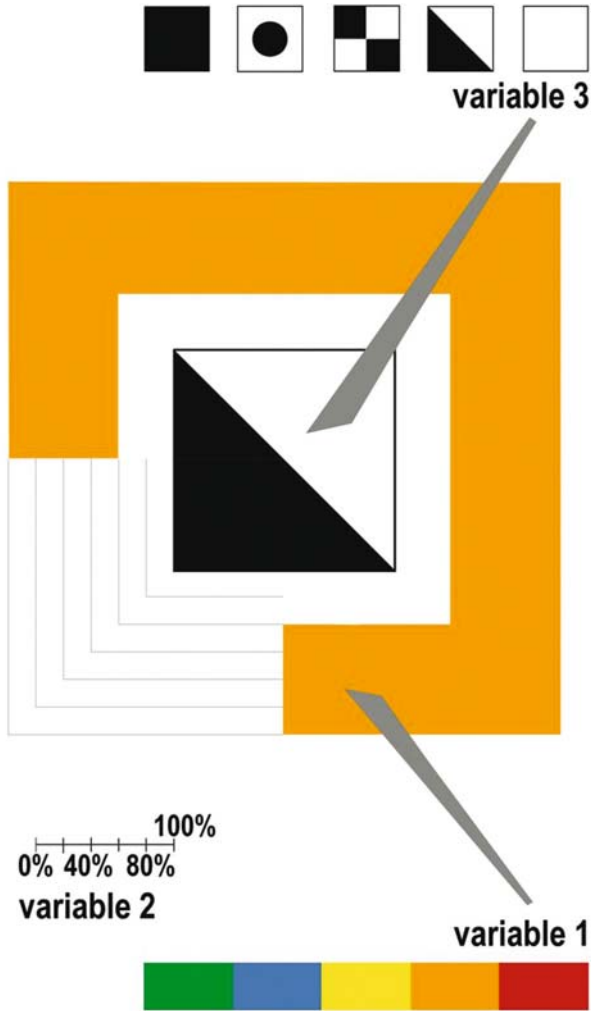


Fig. 7 Firewall icon

3.5 Implementation

The VisAlert prototype technology was implemented in C++ and first deployed at the Air Force Research Lab (AFRL) in Rome, New York. During the testing phase we collected comments and suggested features. VisAlert generated very positive response: users specifically noted its effectiveness, simplicity, and flexibility. They stated that it provided increased situational awareness to detect, diagnose and respond to network events and anomalies.



Fig. 8 Composite firewall icons – $O(2^{12})$

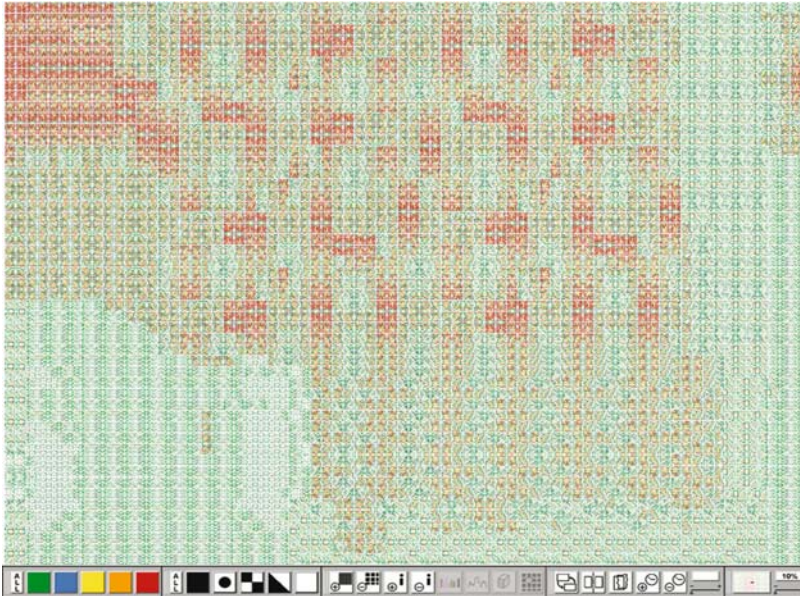


Fig. 9 Composite firewall icons – $O(2^{24})$

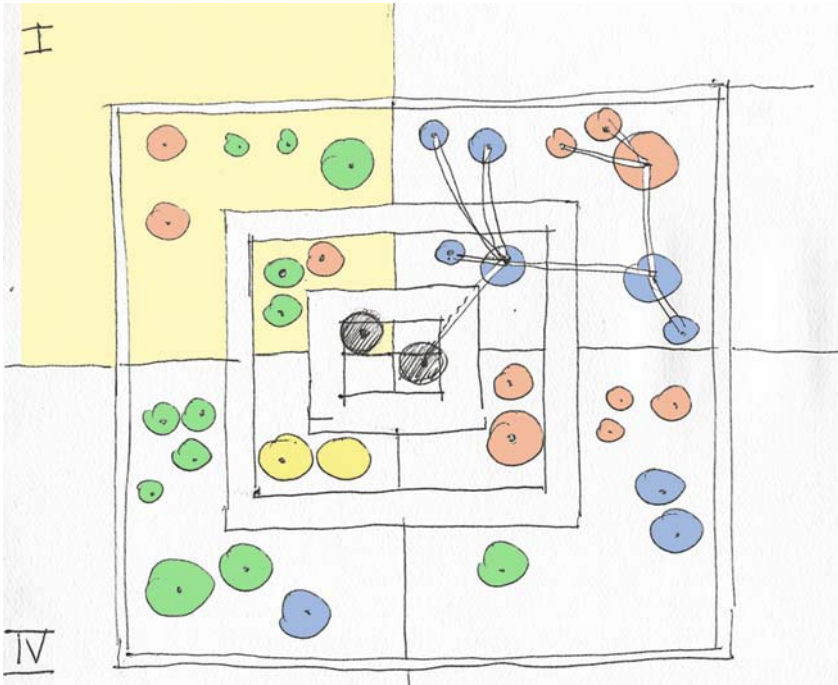


Fig. 10 Firewall topology sketch

However, the analysts wanted more deployment flexibility and a technology that would be operating system and database agnostic. We then completely rewrote the software in Java and integrated Hibernate database management software to accomplish the task (Fig. 13). Currently the system is being evaluated by several commercial network security software vendors.

4 Future Work

Ongoing and future work is in the following areas that are conducive to the complete user experience with her information space to make decisions.

- *Interaction.* The word “visualization” indicates how the data is presented to the user. The next step is to increase the ability of the user to fully interact with the information space, thus providing more effective ways to input data and to control which data is displayed.
- *Extensibility.* This refers to the ability of the visualization software to be extended and include new modules, and moreover to interact with other tools that users are familiar with. This includes the ability or users to encode and correlate their own alert algorithms.

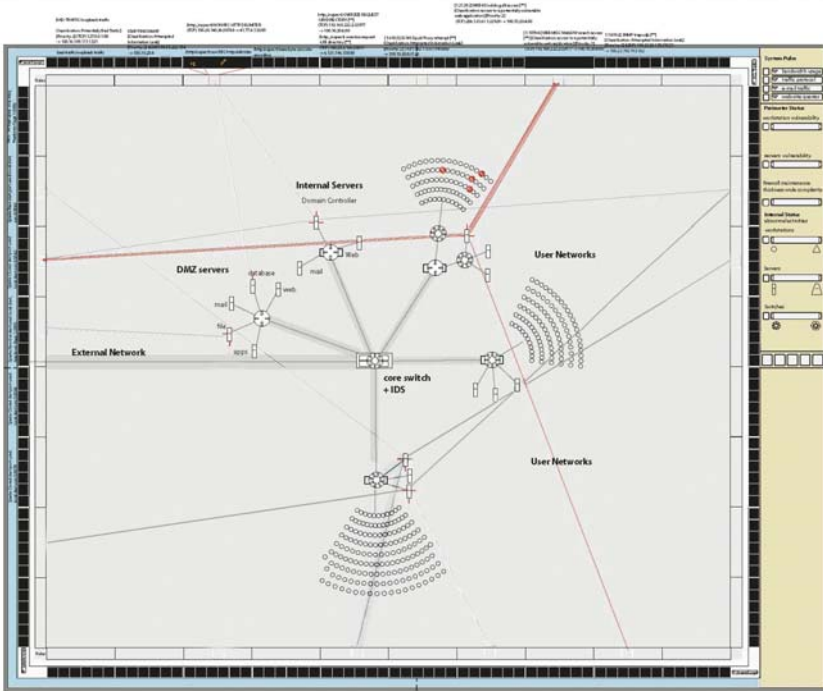


Fig. 11 Square visualization screen

- *Continuity*. This refers to ability to present multiple perspectives and level of details of the information space, and to smoothly transfer from view to view while maintaining the context and references.

More specifically to network security, future work includes the design of additional visualization structures that enable analysts to perform hypothesis testing of events and details, and reporting summaries to decision makers. The complete VisAlert system could then evolve in a visual continuum that would allow seamless transition from a holistic view of the system all the way to detail drill down.

5 Conclusions

The design process, starting from “thinking outside of the box”, creating several concepts, refining ideas, and combining them is an organic and effective method to produce visual concepts that encode the relevant information and enable users to enhance the way they work, use their information space, and make decisions.

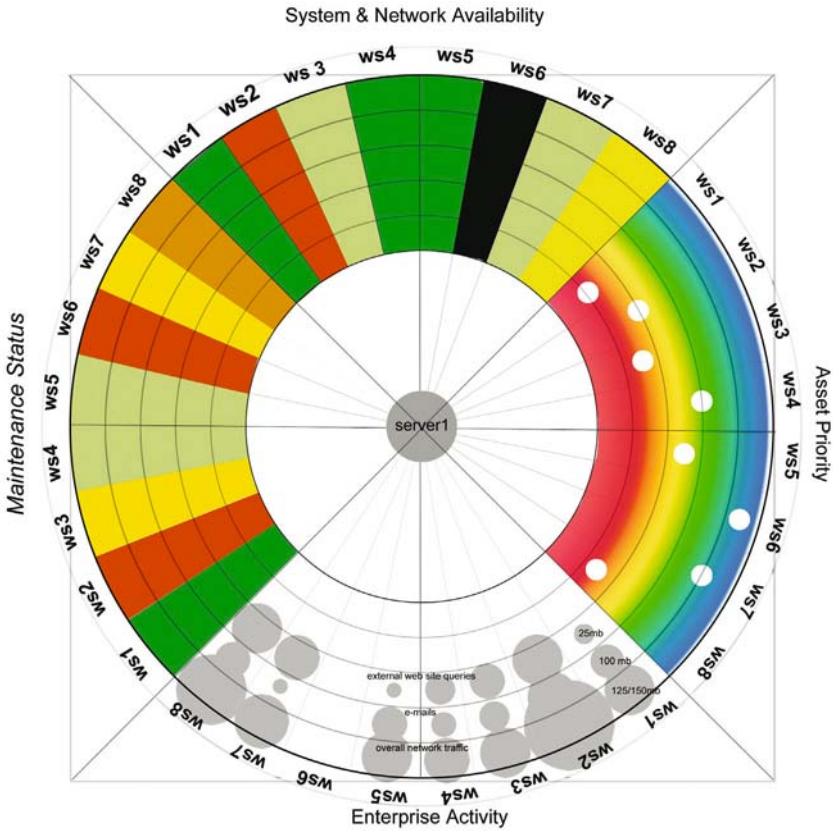


Fig. 12 Radial visualization

In order to optimize the chances that technology is actually used, the development requires interacting with users from the very beginning to address their specific problems, tasks, and mental models.

A very important technical characteristic of VisAlert that enabled user acceptance was the ability to fuse in one view *any* and *all* the data of choice of the user, and to filter out the unwanted one.

User centered design can be addressed very effectively by involving an interdisciplinary team that builds trust and value in different roles and perspectives.

Acknowledgements As indicated in the description of the team and roles, there are numerous people and organization that have had a significant contribution in VisAlert’s success. This includes the whole CROMDI team and other personnel at the University of Utah (with particular gratitude to Julio Bermudez and Shaun Moon), Utah State University (with particular gratitude to Robert Erbacher), the AFRL, Battelle, and Skaion Corporation.

This work was supported in part by a grant from the IC-ARDA, the Utah State Center of Excellence Program, and DARPA SBIR.

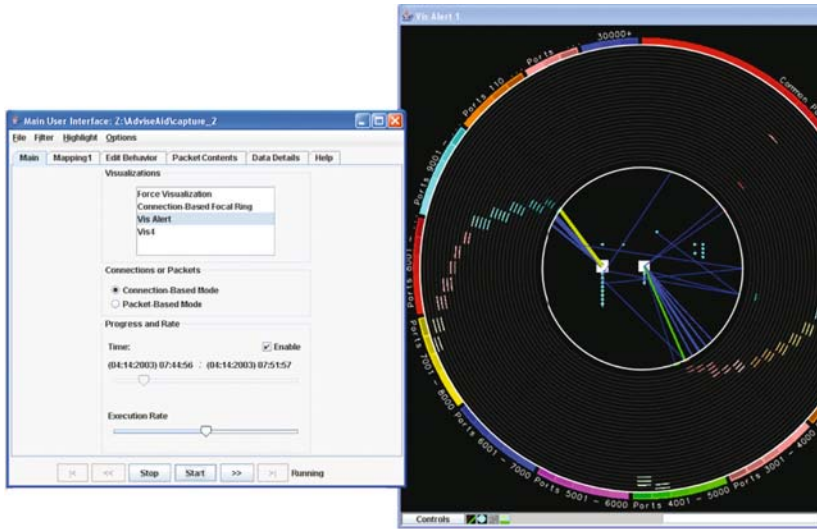


Fig. 13 Screen shot of the current VisAlert product

References

- Arnheim R (1977) *The Dynamics of Architectural Form*. Berkeley: University of California Press.
- Bermudez J, Agutter J, Westenskow D, Foresti S, et al. (2000) Data Representation Architecture, in M. Clayton and G. Vasquez de Velasco (eds): *ACADIA 2000*. Washington DC, pp. 91–102.
- Bogdan C (2002) *The Semiotic of Visual Languages*. New York: Columbia University Press.
- Cox K, Eick S, He T (1996) 3D geographic network displays, *ACM Sigmod Record*, vol. 25, no. 50.
- Cross N (1982) Designerly Ways of Knowing, *Design Studies* vol. 3, no. 4, pp. 221–227.
- D'Amico A, Larkin M (2001) Methods of visualizing temporal patterns in and mission impact of computer security breaches, *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, vol. 1, pp. 343–354.
- Estrin D, Handley M, Heidermann J, et al. (2000) Network visualization with NAM, the VINT network animator, *IEEE Computer*, vol. 33, pp. 63–68.
- Foresti S, Agutter J, Livnat Y, Moon S, et al. (2006) VisAlert: visual correlation of network alerts, *IEEE Computer Graphics and Applications*, vol. 26, no. 2, pp. 48–59.
- Friedman B (1997) *Human Values and the Design of Computer Technology*, Center for the Study of Language and Information, Stanford, CA.
- Kraut R, Galegher J, Egido C (1988) Tasks and relationships in scientific research collaborations, *Human-Computer Interaction*, vol. 3, pp. 31–58.
- Polla D, McConnell J, et al. (1998) A framework for cooperative intrusion detection, *Proceedings of the 21st National Information Systems Security Conference*, pp. 361–373.
- Rowe P (1987) *Design Thinking*. Cambridge: The MIT Press.
- Snodgrass A, Coyne R (1990) *Is designing hermeneutical?* Technical Report, Sydney Australia.
- Teoh S, Ma K, Wu S, et al. (2002) Case study: interactive visualization for internet security, *Proceedings of the IEEE Conference on Visualization*, pp. 505–508.
- Teoh S, Ma K, Wu S, et al. (2003) Visual exploration process for the analysis of internet routing data, *Proceedings of the IEEE Conference on Visualization*, pp. 523–530.
- Wong W (1972) *Principles of 2-D Design*. New York: Van Nostrand Reynolds.
- Wong W (1977) *Principles of 3-D Design*. New York: Van Nostrand Reynolds.

Visually Understanding Jam Resistant Communication

D. Schweitzer, L. Baird, and W. Bahn

Abstract The primary goal of information security is to ensure the confidentiality, integrity, authenticity, and availability of information. Availability is often relegated to a discussion of denial of service attacks on network resources. Another form of denying availability is to prevent communication through the use of traditional jamming techniques. At the United States Air Force Academy Center for Information Security, we have been working on a new algorithm, BBC, which is based on a new type of coding theory known as concurrent codes that is resistant to traditional jamming techniques. While the formal definition and proofs of concurrent codes can be daunting, the algorithm's effectiveness can be easily conveyed and appreciated through visual demonstration. This paper briefly introduces concurrent codes and describes an interactive applet that visually demonstrates the algorithm's effectiveness in a noisy environment.

1 Introduction

Traditional omni-directional techniques for jam resistance all assume that the sender and receiver share a secret key that is unknown to the attacker (Milstein, 1998). They use spread spectrum communication methods, where there is a sequence of frequency hops, or a sequence of chips, or a sequence of pulses, which are all unpredictable to the attacker (Kanterakis, 1994; Bergstrom and Chuprun, 1998; Bergel et al., 2003). It is this unpredictability that provides resistance to jamming. Shared secret approaches do not scale well in an environment with vast numbers of participants, such as a net-centric warfare environment. It is not practical to manage the distribution and secure access while maintaining the confidentiality of a shared key with thousands of users.

D. Schweitzer, L. Baird, and W. Bahn
United States Air Force Academy

To address the problem of providing jam-resistant communication without a shared secret, researchers at the United States Air Force Academy have developed a new message coding scheme described briefly below. The approach allows multiple messages to be simultaneously broadcast and correctly decoded in the presence of additional traffic and/or random noise whether intentional or not. The challenge is to provide an understanding of its effectiveness in a noisy/hostile environment without resorting to complex equations or theorem proofs. To provide this understanding, a visualization was developed that demonstrates the algorithm and its effectiveness even under heavy intentional “jamming” by the user. This visualization is described along with our experience in how well it demonstrates the concepts.

2 Related Work

Intentional jamming is an attempt to disrupt communications by lowering the signal to noise ratio for the legitimate communicators. The attacker sends (possibly random) noise at the same frequency as the legitimate signal. With spread spectrum technology, it is much more difficult to cover all frequencies being used to communicate. To jam a legitimate signal, the attacker must use far more energy than the legitimate sender.

This approach, which uses a shared secret, works well on a small scale. If a small group of people need to communicate by radio, then they can share a secret key beforehand. However, it breaks down completely on a large scale. For example, civilian GPS signals have no jam resistance. This is unfortunate, because the FAA has stated that the commercial airline industry will become increasingly reliant on civilian GPS for all of its navigational aids. If a terrorist jams the signal at a major airport, it could cause serious problems.

Another example is the civilian cell phone system. It is currently possible to buy a small cell phone jammer on the internet (<http://www.globalcadgetuk.com> 2007). The cell phone system has no resistance to jamming whatsoever, because it would be unthinkable to have every subscriber share a single shared secret (which wouldn't remain secret for long), or to give every subscriber a different secret key, and then have all the millions of keys loaded into every cell phone tower, and have it listen on millions of channels simultaneously.

Another example is the fact that the Air Force has stated that the future of warfare is net-centric warfare (Raduege, 2007). This means that every vehicle and device in theater will be a node in a wireless, ad hoc internet. If the Air Force plans to rely heavily on this network, then it is critical that it be resistant to jamming. However, a jam resistance system based on a shared key will not scale up. It is not practical to create a single shared key, and to distribute it to every person, vehicle, radio, and device that will be deployed to the theater, and to assume it will remain secret. This scaling problem is why the common access card (CAC) is based on public key cryptography rather than on symmetric cryptography that uses a shared secret. Just as the 1970s saw the invention of public key crypto, which scaled better than

symmetric crypto, so we currently have a critical need for the equivalent for jam resistance (Diffie and Hellman, 1976).

2.1 BBC and Concurrent Codes

The first system ever proposed for this problem is the BBC algorithm. It allows jam resistance without a shared secret. Many radios can broadcast messages simultaneously, and the receiver will receive all of the messages without error. It can be built on top of any of the three most common forms of spread spectrum: frequency hopping, direct sequence, or pulse-based Ultra Wide Band.

The BBC algorithm is based on a new field of coding theory, concurrent codes. These are a subset of superimposed codes, and are very different from traditional error detecting or error correcting codes. A number of theorems have been proven about these codes, and about the BBC algorithm and other algorithms based on these codes. This math can be fairly complex, which is why some form of visualization is useful for describing it to a less technical audience of decision-makers.

The BBC algorithm works as follows. First, there must be a way to send an *indelible mark* at chosen *locations*. If BBC is run on a pulse-based UWB system, then the *indelible mark* is a short pulse of very high power RF noise that spans a large spectrum of frequencies. The *location* of the mark is the exact time of the pulse. There is no information encoded in the random noise of the pulse itself. All of the useful information is encoded in the exact timing of the pulse. Because the pulse is very short, very random, and very powerful, it is not possible, in any practical sense, for an attacker to erase a pulse, in the way a sine wave is erased by destructive interference when an identical wave is sent exactly out of phase with it. Therefore, an attacker can only create new pulses, not erase existing pulses.

In pulse-based BBC, the message is sent by appending several zeros which act as checksum bits to a binary message, hashing each possible prefix of the message, and sending one pulse for each prefix at a location determined by the hash. For example, to send the message 1011, it would first append zeros to get 1011000, then it would take each prefix of the message {1, 10, 101, 1011, 10110, 101100, 1011000}, then it would send seven pulses, where the exact time of each pulse is determined by a hash of each of those seven prefixes as shown graphically in Fig. 1.

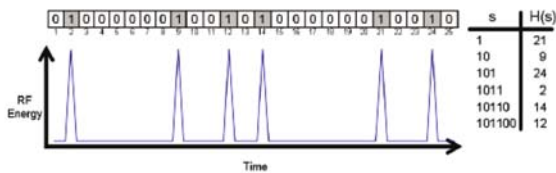


Fig. 1 Encoding messages as pulses in time

This allows the user to actually see the encoding. The decoding tree can also be shown visually, so that the user understands the amount of computation being performed. It can also allow interactive use by the user, where the user draws additional noise on the picture in an attempt to jam it. This gives insight into the robustness of the system. For these reasons, the visual approach appears to be the most useful for gaining an understanding of the system.

3 Technical Approach

While concurrent codes offer many advantages to traditional approaches for jam resistant communications, these benefits are not always readily understood by decision-makers and possible project funders. The basic algorithm for encoding and decoding can be quickly explained, but the resiliency to additional noise in the system is difficult to appreciate based solely on the mathematical analysis. This is especially true for a less technical audience, such as some higher level management and fiscal managers.

The obvious approach to demonstrate the effectiveness of the algorithm is to simply have two communication devices talking using the algorithm and show that they can continue to operate, even under noisy conditions. Unfortunately, having two software radios, or computers, communicate is not a very impressive demonstration. As discussed previously, the algorithm is “hidden” in the software, and the noise level is hard, if not impossible, to judge. The user simply sees messages being sent and received correctly. Even if additional information about noise levels, such as some visual presentation of frequencies, is presented, the user has a difficult time understanding the significance of what they are seeing.

To address this challenge in communicating the effectiveness of the BBC algorithm, the ACIS research group developed a demonstration that would be meaningful and understandable to a less technical audience. The goal was for viewers to easily understand how resistant to noise the algorithm was, and to be able to appreciate some of the more subtle characteristics of the algorithm, such as how the computational costs grow with noise.

3.1 *An Audio Solution*

The first attempt at developing an understandable demonstration of the BBC algorithm was to use sound pulses as the communication medium. Three laptop computers were set up close to each other with two sending messages while the other received them. To send a message, the desired message was encoded as a series of high-pitched “beeps” set in time based on the hash of the message prefix. At approximately the same time, each sending computer would start sending their encoded messages simultaneously as audio beeps transmitted through the laptop speaker.

The receiving laptop would be “listening” for beeps through the attached microphone. Each beep received would be recorded. After recording long enough to receive all of the beeps, the receiver would decode the received messages using the BBC algorithm interpreting each beep as a received bit that may or may not be in the final message and display the results.

This demonstration was successful in demonstrating how two senders could simultaneously send messages that could be correctly decoded by a receiver. It also made the concept of “noise” easy to understand, since in addition to the two senders simultaneously beeping, the receiver was also recording any background noise such as people talking, equipment noise, etc. Another advantage of this demonstration is that it shows the algorithm’s robustness to inexact timing. The two senders did not start at exactly the same moment in time, and the timing calculations for when to send beeps vs. when beeps were received are slightly inexact for the three laptops. However, the receiver is able to correctly decode messages in spite of these variations.

While a powerful and useful demonstration, a disadvantage of the audio version is that it still does not give the viewer a clear understanding of the amount of noise in the system and exactly how resilient the algorithm is. It is easy to understand that there is additional noise, but not the amount or the effect of the noise to computation cost. To achieve this understanding, the group set out to develop a visual demonstration. The goal was to have a demonstration that was easy to understand, visually meaningful, and interactive.

3.2 A Visual Representation

The first step in developing a visual demonstration was to come up with a meaningful representation for encoded messages that was visually compact and understandable. The description of the algorithm often uses the specific implementation of pulses on a timeline as an easy to understand way of encoding messages. Pulses represent bits that can be turned on, but not off. The user can envision this as a linear timeline with pulses at set points in time as was shown in Fig. 1. Multiple messages are encoded by adding pulses at the appropriate point in time. Noise is represented as pulses that are not part of the encoded messages. The number of pulses, and thus density of the timeline, represents how noisy the message space is.

A variation of this visual analogy was used for the visual demonstration. The linear timeline is “wrapped around” as successive rows of a bit map. Rather than pulses represented as visual “spikes”, a single dot on the bit map represents the presence of a pulse at that point in time. Bits can be turned on as additional messages and noise are added to the system, but bits can not be turned off. Using this analogy, the visual denseness of the bit map image represents how noisy the message space is (see Fig. 3).

In addition to the compactness, simplicity, and understandability of this representation, there is another advantage to using the bit map approach. Adding noise

Fig. 3 Bit map representation of bits in message space

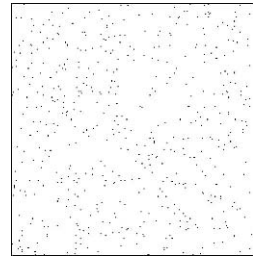
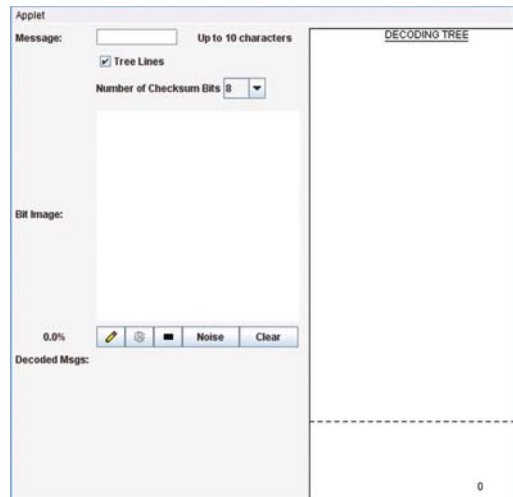


Fig. 4 Initial screen shot of the BBCVis applet

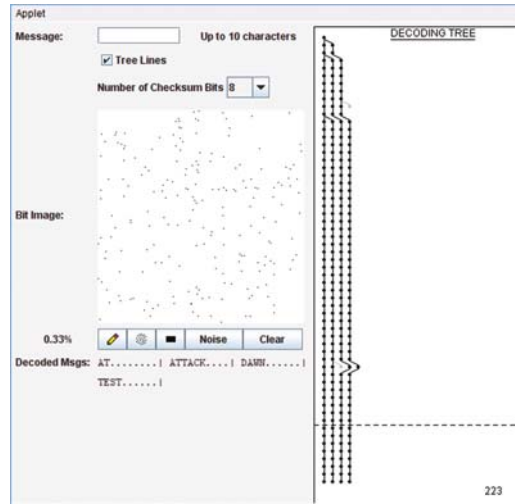


to the system is equivalent to turning on random bits in the bit map. Familiar paint tools (pencil, spray can, and solid rectangle) can provide this capability to the user without needing lengthy explanation.

Along with representing the message space, it was desirable to show how much computation was required to decode the message(s), and the impact of multiple messages and noise in the system. A binary decoding tree as described earlier was used for this purpose. The number of tree nodes represents how much computation is necessary to decode the message(s). As additional messages and noise are added to the message space, the size of the tree grows, both for actual messages as well as false paths. A sense of the relative amount of computation can be quickly gleaned from the visual density of the tree representation. For example, visually comparing the decoding trees in Figs. 5–8 shows the increased amount of computation necessary as the amount of noise is increased.

The bit map message space representation and binary tree computation metaphor are combined in BBCVis, an interactive applet for experimenting with and understanding concurrent codes. Figure 4 shows the initial layout for the applet. Messages are entered into the message box and encoded into the bit image. Decoded messages are displayed below along with the associated decoding binary tree on the right. Figure 5 shows the applet after four messages have been entered. The density of

Fig. 5 Four encoded messages



the bit map image is updated as images are encoded, and represent less than 1% in the example shown. The decoding tree shows that very few additional nodes were searched beyond the actual messages (nodes resulting in terminated branches are shown as hollow circles). The four paths surviving to the bottom of the tree represent the four messages that were decoded. The dotted line on the decoding tree panel represents the start of the checksum bits which were appended to each message before encoding.

Figure 6 shows the effect of “adding random noise” to the system. Random bit locations are set along with the original bits for the encoded messages. This is analogous to noise added to the received signal, either intentionally or not. As a result of this additional noise, the decoding tree has grown to account for the additional nodes that needed to be searched to find the final decoded messages. An indication of the robustness of the algorithm is that a significant increase in the bit density of the message space (over 50 times the number of bits) has resulted in only a modest increase in the number of binary tree nodes, about a 25% increase in the number of nodes to search. This is easy to quickly see by the change in the size of the displayed tree. In addition, the four original messages are correctly recovered.

In addition to random noise, the applet allows for intentional noise set at specific locations using the drawing tools. This is analog to an attacker using directed attacks against specific areas of the message space. Figure 7 shows the results of using these tools including large areas that were “blacked out”.

Figure 8 shows how the algorithm behaves under extreme noise conditions. With almost 50% density in the message space (half of the bit image covered), the number of nodes in the decoding tree, and thus computation time, has grown significantly. The original four messages are still recovered, but additional “hallucinations” or false messages are also decoded. One way of removing the hallucinations is to use additional checksum bits when encoding the messages. Figure 9 shows what

Fig. 6 Encoded messages with random noise added to message space

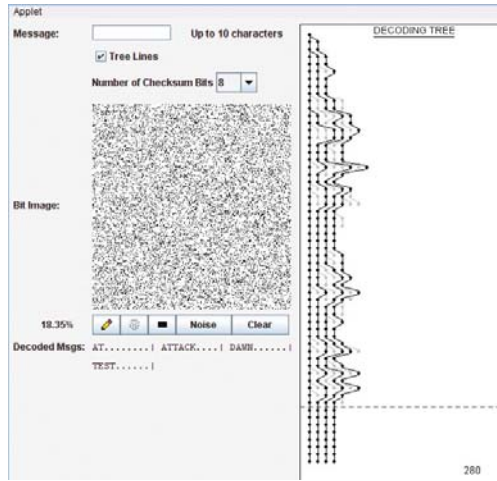
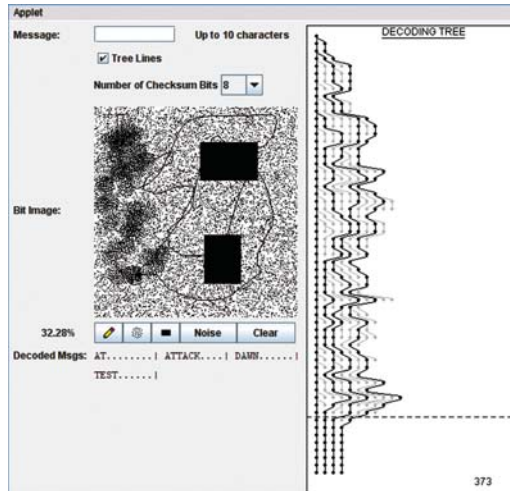


Fig. 7 Encoded messages with both random and directed noise



happens at 100% noise level (completely black image). The tree shows the expected exponential growth (for practical purposes, the algorithm ceases building the tree after a predefined number of maximum nodes at any level occurs).

In an effort to make the visualization as dynamic and informative as possible, we have made the decoding portion of the visualization occur in real-time as messages are entered, or noise is added to the bit image. This allows the user to immediately see the impact on computation as the message space is changed. Perhaps the most dramatic effect occurs when drawing a solid rectangle. The tree grows and shrinks as the rectangle is resized dynamically showing the sensitivity of the computation to the amount of noise in the system. Figure 10 shows two screen shots while dynamically increasing the size of a black square.

Fig. 8 Hallucinations occurring from excessive noise

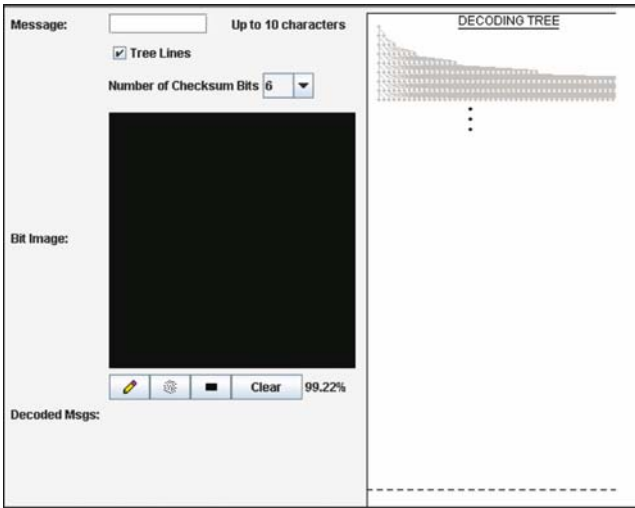
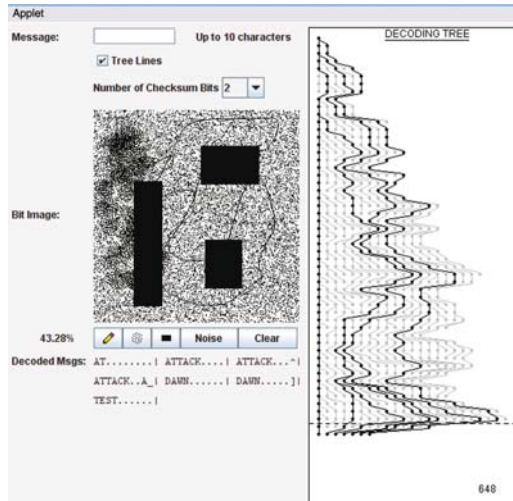


Fig. 9 Maximum noise density

4 Future Work

Concurrent codes and the BBC algorithm are gaining recognition as a viable alternative to traditional communication techniques for certain applications. One approach that is being investigated is a hybrid approach of using BBC for initial handshaking and key exchange followed by traditional spread spectrum techniques once a connection has been established. We are also conducting research and performing analysis on different attacks an adversary might attempt against the technique.

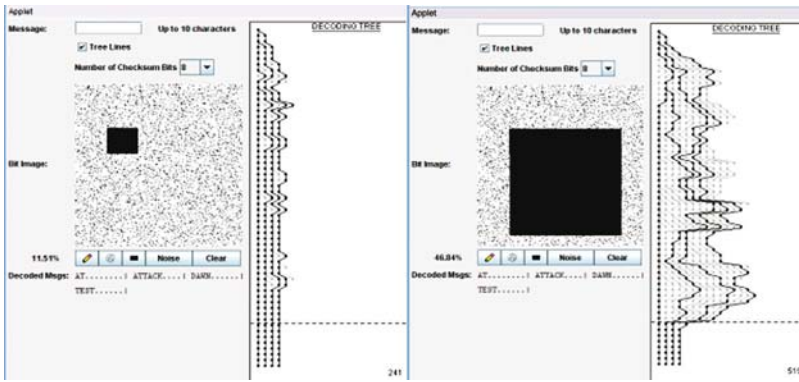


Fig. 10 Decoding tree dynamically changing while adding noise

In addition to the communication application of concurrent codes, other applications are being investigated such as rapid text retrieval for search engines. In theory, a concurrent code approach could speed up Google searches. Another potentially huge area would be the use of concurrent codes for RFID tags. Because large numbers of messages can be simultaneously transmitted and decoded, a shelf of tagged items in a warehouse can theoretically all send their information at the same time and be correctly decoded for inventory purposes.

The BBCVis applet has proven effective for communicating the algorithm. As new applications for concurrent codes are developed, ways of modifying or expanding the visualization will be investigated to incorporate the new approaches. BBCVis is part of a larger suite of visualization applications for security education at the Air Force Academy.

5 Conclusions

The BBCVis applet has been demonstrated to several individuals at different levels of familiarity with the algorithm. The concept of a message encoded as bits in the image and decoded seems to be quickly grasped. The use of the drawing tools to add noise is also quickly understood. The significance of the tree is less intuitive without explanation. However, the concept that the number of tree nodes and visual density relates to the amount of computation time is rapidly appreciated.

In addition to the basic understanding, several viewers expressed similar reactions to interacting with the tool. For example, many are surprised at how much noise can be added without significantly affecting the amount of computation or corrupting the decoded messages. Similarly, many noted how the sensitivity increases dramatically at key noise densities, such as 50%. Figure 11 shows a chart of the number of nodes in the decoding tree as a function of percent of coverage in

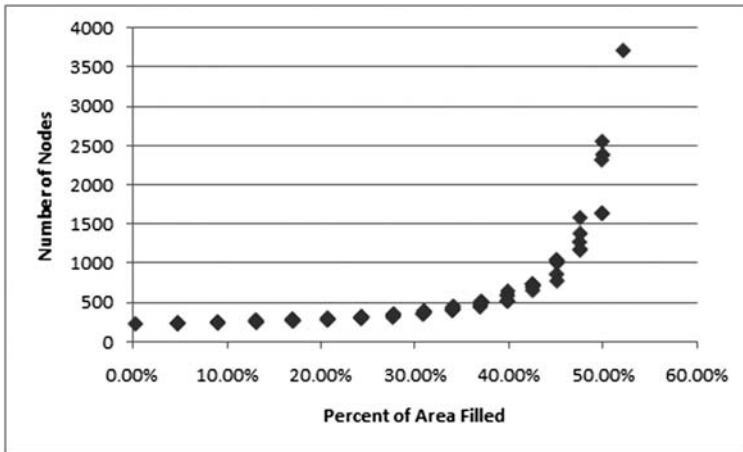


Fig. 11 Chart of decoding nodes to message/noise density

the signal space. These results, which were obtained experimentally, match the theoretical model and illustrate the sharp increase that occurs at the 50% point.

In summary, the BBCVis applet is effective in describing the concurrent code algorithm and characteristics. Users are able to quickly understand the visual metaphors and effectively interact with the applet to experiment with and understand the underlying algorithm.

This work was sponsored in part by the Air Force Information Operations Center (AFIOC), Lackland AFB, TX, and was performed at the Academy Center for Information Security (ACIS) at the United States Air Force Academy.

References

- Baird L, Bahn W, Collins M (2007) Jam-resistant communication without shared secrets through the use of concurrent codes. US Air Force Academy Technical Report, USAFA-TR-2007-01
- Bergel I, Fishler E, Messer H (2003) Low complexity narrow-band interference suppression in impulse radio. Proceedings of the 2003 International Workshop on Ultra Wideband Systems (IWUWBS)
- Bergstrom C, Chuprun J (1998) Optimal hybrid frequency hop communication system using nonlinear adaptive jammer countermeasures and active fading mitigation. IEEE Global Telecommunications Conference, GLOBECOM 98, volume 6
- Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans. Inform. Theory, IT-22(6):644–654
- Kanterakis E (1994) A novel technique for narrowband/broadband interference excision in ds-ss communications. MILCOM '94, volume 2, 628–632
- Milstein L (1998) Interference rejection techniques in spread spectrum communications. Proc. IEEE, 76(6):657–671
- Raduege H (2007) Net-centric warfare is changing the battlefield environment. CrossTalk. <http://www.stsc.hill.af.mil/crossTalk/2004/01/0401Raduege.html>, accessed March 2007

Visualization of Host Behavior for Network Security

F. Mansman, L. Meier, and D.A. Keim

Abstract Monitoring host behavior in a network is one of the most essential tasks in the fields of network monitoring and security since more and more malicious code in the wild internet constantly threatens the network infrastructure. In this paper, we present a visual analytics tool that visualizes network host behavior through positional changes in a two-dimensional space using a force-directed graph layout algorithm.

The tool's interaction capabilities allow for visual exploration of network traffic over time and are demonstrated using netflow data as well as IDS alerts. Automatic accentuation of hosts with highly variable traffic results in fast hypothesis generation and confirmation of suspicious host behavior. By triggering the behavior graph from the HNMap tool, we were able to monitor more abstract network entities.

1 Introduction

Today, a lot of research deals with an increasing amount of data being digitally collected in the hope of revealing valuable information that can eventually bring about a competitive advantage. Visual data exploration, which can be seen as a hypothesis generation process, is especially valuable, because (a) it can deal with highly non-homogeneous and noisy data, and (b) is intuitive and requires no understanding of complex mathematical methods (Keim and Ward, 2002). Visualization can thus provide a qualitative overview of the data, allowing data phenomena to be isolated for further quantitative analysis.

The emergence of *visual analytics* research suggests that more and more visualization research is closely linked with automatic analysis methods. Its goal is to turn information overload into the opportunity of the decade (Thomas, 2005;

F. Mansmann, L. Meier, and D.A. Keim

University of Konstanz, Germany, e-mail: mansmann@inf.uni-konstanz.de, meier@inf.uni-konstanz.de, keim@inf.uni-konstanz.de

Thomas and Cook, 2005). Decision-makers should be enabled to examine this massive, multi-dimensional, multi-source, time-varying information stream to make effective decisions in time-critical situations. For informed decisions, it is indispensable to include humans in the data analysis process to combine flexibility, creativity, and background knowledge with the enormous storage capacity and computational power of today's computers. The specific advantage of visual analytics is that decision makers may focus their full cognitive and perceptual capabilities on the analytical process, while allowing them to apply advanced computational capabilities to augment the discovery process.

Our objective is to show how visual analysis can foster better insight in the large data sets describing IP network activity. A non-trivial task of detecting different kinds of system vulnerabilities can be successfully solved by applying the visual analytics approach. Whenever machine learning algorithms become insufficient for recognizing malicious patterns, advanced visualization and interaction techniques encourage expert users to explore the relevant data and take advantage of human perception, intuition, and background knowledge. In the process of human involvement acquired knowledge can be further used for advancing automatic detection mechanisms.

This paper focuses on tracking of behavioral changes in traffic of hosts as one of the most essential tasks in the domains of network monitoring and network security. We propose a new visualization metaphor for monitoring time-referenced host behavior. Our method is based on a force-directed layout approach which allows for a multi-dimensional representation of several hosts in the same view. This new visualization metaphor emphasizes changes in the traffic data over time and is therefore well suited for detecting uncommon system behavior. We use the visual variable position to give an indication about traffic proportions of hosts at a particular moment in time: High traffic proportions of a particular protocol attract the observation nodes resulting in clusters of similar host states. So-called traces then connect the snapshots of hosts (one snapshot for every time interval) in chronological order resulting in one chain per host.

Various interaction capabilities allow for fine-tuning the layout, highlighting of hosts of interest, and retrieval of traffic details. As a contribution to visual analytics, we implemented an automatic highlighting of hosts with high variations in the used application protocols of network traffic in order to guide the interactive exploration process.

The rest of the paper is structured as follows: Sect. 2 discusses related work in the field of visualization for network monitoring and security with a focus on tools analyzing application ports, graph-based approaches, and visual analytics applications. The next section details our system and the graph-based layout, including a description of the available user interactions. Since the tool lends itself to be applied to more abstract information, we then show how it can be integrated in our previously proposed HNMap tool to monitor network behavior of prefixes, autonomous systems, countries, or continents. To demonstrate and evaluate the usefulness of the behavior graph, we conduct a small case study and present means for automatic highlighting of high variance hosts. After presenting some ideas about further developments of our tool (Sect. 4), the last section sums up our contributions.

2 Related Work

Ultimately, all previously proposed methods support the administrators in their task to gain insight into the causes of unusual traffic, malfunctions, or threat situations. Besides automatic analysis means, network operators often relied on simple statistical graphics like scatter plots, pair plots, parallel coordinates, and color histograms to analyze their data (Marchette, 2001). However, to generate meaningful graphics, the netflow data and the countless alerts generated by IDSes need to be intelligently pre-processed, filtered, and transformed since their sheer amount causes scalability issues in both manual and visual analysis. Since traditional statistical graphics are familiar to analysts, their design often forms the basic metaphor of newly proposed visualization systems. Therefore, additional interaction features enhance the user's capabilities to discover novel attacks and to quickly analyze threat situations under enormous time pressure.

One such visualization systems is *IDS Rainstorm*, which bridges the gap between large data sets and human perception (Abdullah et al., 2005). A scatterplot-like visualization of local IP addresses vs. time is provided to analyze the thousands of security events generated daily by the IDS. After zooming into regions of interest, lines appear and link the pictured incidents to other characteristics of the data set.

A demonstrative example of work in the field is the situational awareness is *VisAlert* (Livnat et al., 2005) which is built upon the w^3 premise, assuming that every incident has at least the three attributes what, when, and where. In the *VisAlert* display, the location attribute is placed on a map, the time attribute indicated on concentric circles around this map, and the classification of the incident is mapped to the angle around the circle. For each incident, the attributes are linked through lines.

This linking in detail views is also utilized in other applications like *TNV* (Goodall et al., 2006). The main matrix links local hosts, which are colored according to their activity level, to external hosts through straight and curved lines. In addition to that, the system includes a time histogram, a bifocal lens to enlarge the focus area, colored arrowheads to show traffic direction and protocols, parallel coordinates linking source and destination port, and details on demand interaction techniques. While this open source tool is excellent for monitoring a small local network, its limit to display approximately 100 hosts at a time might cause scalability issues when monitoring medium or large size networks.

As already mentioned, parallel coordinates have become a popular analysis technique when dealing with network data. *VisFlowConnect* uses the parallel axis view to display netflow records as in- and outgoing links between two machines or domains (Yin et al., 2004). This techniques allows the analyst to discover a variety of interesting network traffic patterns, such as virus outbreaks, denial of service attacks, or network traffic of grid computing applications.

It is worth mentioning that visualization techniques like parallel coordinates and graphs have meanwhile found their way into commercial products, such as the *RNA Visualization Module* of SourceFire (Sourcefire, 2005). However, major drawbacks of parallel coordinates' are that they introduce visual clutter due to overplotting of lines and that only correlations between neighboring axes can be identified.

2.1 Analysis of Application Ports

An important subarea is visualization of application port activity as an indication to the running network applications. (Lau, 2004), for example, presented the *Spinning Cube of Potential Doom*, a 3D scatterplot with the dimensions local IP address, port number, and global IP address. The cube is capable of showing network scans due to emerging patterns. However, 3D scatterplots may be difficult to interpret on a 2D screen due to overlay problems.

Another port analysis tool is *PortVis* described by McPherson et al. (2004). It implements scatterplots (e.g., port/time or source/port) with zooming capabilities, port activity charts, and various means of interaction to visualize and detect port scans as well as suspicious behavior on certain ports.

For a more detailed analysis, Fink et al. (2005) proposed a system called *Portall* to allow end-to-end visualization to view communications between distributed processes across the network. This system enables the administrator to correlate network traffic with the running processes on his monitored machines.

2.2 Graph-Based Approaches for Network Monitoring

In network monitoring and security, graph-based approaches have been intensively used. In most cases, however, their use is limited to expressing communication between hosts or higher-level elements of the network infrastructure among each other along with information about traffic intensity.

Early internet mapping projects put their focus on geographic visualization where each network node had a clearly defined geographic position on a map. The same principle was applied in a study to map the multicast backbone of the internet (Munzner et al., 1996). Since the global network topology was shown, the authors used a 3D representation of the world and drew curved edges on top. Other research focused on visual scalability issues in 2D representations ranging from matrix representation to embeddings of the network topology (Eick, 2005) in a plane.

Measuring the quality of network connections in the internet through metrics results in huge data sets. Visualizing this information in graphs becomes both challenging in terms of the layout calculation as well as in terms of visibility of nodes and links of such a graph. Cheswick et al. (2000), for example, mapped about 88,000 networks as nodes having more than 100,000 connecting edges. Another related study implements a hybrid approach by using longitudinal and hierarchical BGP information for their graph layout (Claffy, 2001).

For further reading, we recommend Chaomei Chen's book "Information Visualization – Beyond the Horizon" (Chen, 2004) since it contains a nice overview of the history of internet cartography.

2.3 Towards Visual Analytics for Network Security

One of the key challenges of visual analytics is to deal with the vast amount of data from heterogeneous data sources, such as the countless number of events and traffic collected in log files originating from traffic sensors, firewalls, and intrusion detection systems. Like demonstrated in (Lee et al., 2005), consolidation and analysis of these heterogeneous data can be vital to properly monitor systems in real-time threat situations. Because gaining insight into complex statistical models and analytical scenarios is a challenge for both statistical and networking experts, the need for visual analytics as a means to combine automatic and visual analysis methods steadily grows along with increasing network traffic and escalating alerts.

Muelder et al. (2005), for example, proposed a tool to automatically classify network scans according to their characteristics, ultimately leading to a better distinction between friendly scans (e.g., search engine web crawlers) and hostile scans. Wavelet scalograms are used to abstract the scan information on several levels to make scans comparable. These wavelets are then clustered and visualized as graphs to provide an intuition about the clustering result.

Xiao et al. (2006) start their analysis in the opposite direction. First, network traffic is visualized as scatterplots, Gantt charts, or parallel plots and then the user interactively specifies a pattern, which is abstracted and stored using a declarative knowledge representation. A related system is *NVisionIP* (Lakkaraju et al., 2005), which employs visually specified rules and comes with the capability to store them for reuse in a modified form of the `tcpdump` filter language. The visual analytics feedback loop implemented in both approaches allows the analyst to build upon previous discoveries in order to explore and analyze more complex and subtle patterns.

2.4 Summary

While graphs have previously been used to convey connectivity among network hosts, the novelty of our approach lies in its objective to convey the type of traffic through node position. We then connect all snapshots of one single host in chronological order through traces. In this paper we employ an adapted force-directed graph layout to better use the available screen space. At the same time, user interaction and automatic highlighting of suspicious hosts facilitate hypothesis generation and verification through exploration of their behavior in our visual analytics tool.

3 Technical Approach

The goal of our visualization is to effectively discover anomalies in the behavior of hosts or higher level network entities by comparing their states over time. Figure 1 shows the states of host A and host B at the time intervals 1–6 by calculating the

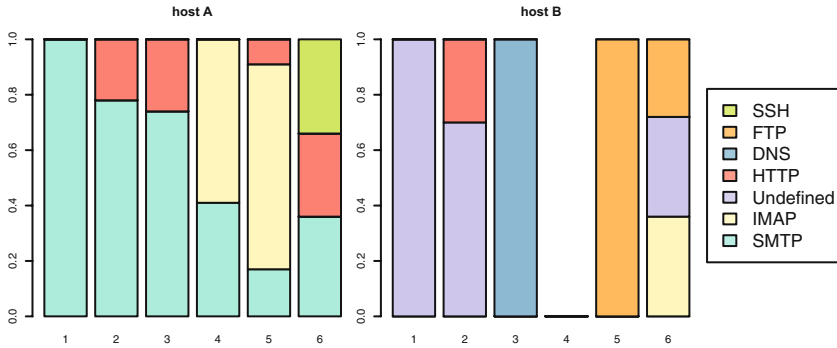


Fig. 1 The normalized traffic measurements define the states of each network entity (host A or host B) for the intervals 1–6. We interpret these states as points in a high-dimensional space (one dimension per traffic type)

normalized traffic proportions for each type of traffic within the interval. Although the figure shows all the relevant information, its scalability is limited since perceiving this detailed information for many hosts and time intervals makes it difficult to keep an overview.

We therefore represent every network entity in a two-dimensional map through several connected points, which all together compose the entity’s *trace*. Both color and shape are used to make the entities distinguishable among each other. Each node represents the state of one network entity for a specific interval and its position is calculated through the entity’s state at that interval. We basically map a high-dimensional space onto a distorted two-dimensional space. If the nodes for one entity are now not in the same place, the entity’s state has changed over time.

This leads to some nice effects which help to visually filter the image. Entities that do not change form small clusters or might even be only visible as a single point, whereas entities that have changed reveal visible trails, either locally or throughout the view. These long lines eventually catch the user’s attention.

To be able to visualize more than two dimensions in a two-dimensional plot, we use an force-directed layout approach to approximate distance relationships from high-dimensional space into 2D. Every data dimension is represented by a *dimension node*. In a first step, the layout of these nodes is calculated. Although arbitrary layouts are possible to place these dimension nodes, the current implementation uses a circular force-directed layout to distribute the nodes on the available space. This chosen layout now defines the distortion of the projected space. After fixing the positions of the dimension nodes, the *observation nodes* are placed in the plane and connected to their corresponding dimension nodes via virtual springs. All observation nodes of the same entity are also tied together with virtual springs. The forces are calculated in an iterative fashion until an equilibrium is approximated. Figure 2 sketches the layout calculation exemplarily for the two hosts from the previous figure. The analyst can now trace the state changes for all intervals of the host.

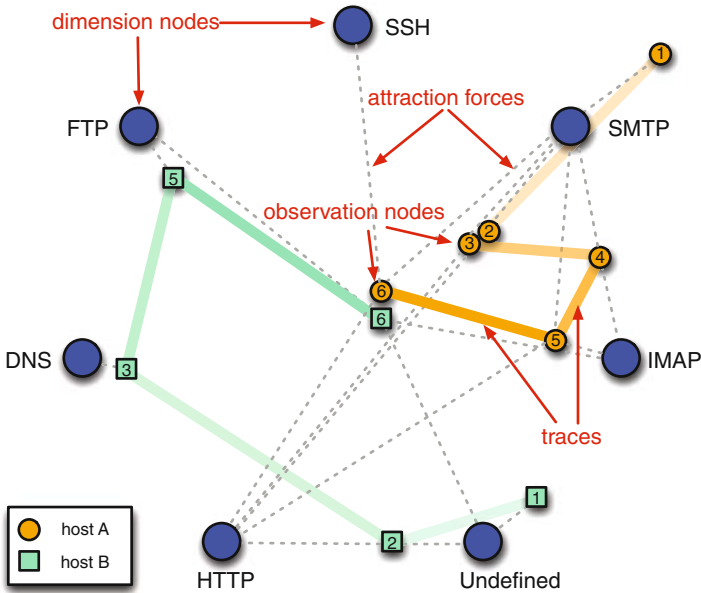


Fig. 2 Sketch showing the coordinate calculation of the host position at a particular point of time. The final graph layout is calculated using a force-based method considering all attraction and repulsion forces

Fine-tuning the graph layout with respect to trace visibility is done by attaching additional attraction forces to the trace edges, which are then taken into consideration during layout calculation. To visually highlight the time-dependency of the object nodes, we mapped the alpha value of the connecting traces to time. Older traces fade out while newer ones are clearly visible.

For many analysis scenarios, not only traffic proportions but also absolute traffic measures play an important role. In other words, the graph layout will assign almost the same position to two nodes with each having 50% IMAP and SMTP traffic, no matter that the first one has transferred several megabytes whereas the second one only a few bytes. We thus varied node size according to the absolute value of the traffic measure (normally the sum of the transferred bytes) using logarithmic scaling due to large variations in traffic measurements.

3.1 Layout Details

The weights of the attraction edges of each *observation node* represent the proportions of the employed application protocols within the network traffic of a particular time interval. The first node of host B in Fig. 2, for example, is only connected to the SMTP attraction node. Since node positions are calculated step-wise using a

spring-embedder graph layout and since all attraction nodes push each other away due to additional repulsion forces, a consistent graph layout is generated where each node has a unique position. We used the (Fruchterman and Reingold, 1991) spring embedder algorithm to calculate the forces between the nodes. The calculation of the attracting forces follows the idea of a physical model of atomic particles, exerting attractive and repulsive forces, depending on the distance. While every node repels other nodes, only nodes that are connected by an edge attract each other. It is important to note that the forces calculated by this algorithm result in speed, not acceleration as in physical systems. The reason is that the algorithm seeks for a static, not a dynamic equilibrium. There are several other algorithms that could solve our layout problem, like the force directed algorithm from (Eades, 1984), the variant of (Kamada and Kawai, 1989), and the simulated annealing approach of (Davidson and Harel, 1996). The reason for choosing the Fruchterman–Reingold algorithm is its efficiency, speed and robustness concerning the force and iteration parameters. As weighted edges were needed we extended the Fruchterman–Reingold implementation of the JUNG (O’Madadhain et al., 2007) graph drawing library to support additional factors on the forces.

3.2 Implementation

To build a flexible and fast analysis system, we relied on the database technology provided by a *PostgreSQL* database (PostgreSQL Global Development Group, 2007). Data loading scripts extract the involved IP addresses along with port numbers, the transferred bytes, and a timestamp from *tcpdump* files, and store them in the database. To speed up query time, traffic with identical IPs and ports can be aggregated in 10 min intervals in a new database table. The actual *behavior graph* application is implemented in Java.

3.3 User Interaction

Since node positions depend on the traffic occurring in the respective time interval and the pushing forces of nearby nodes, only an approximation of the actual load situation is given. Furthermore, due to the multi-dimensional nature of the data at hand, estimating traffic proportions from node positions becomes difficult or even impossible due to ambiguity (e.g., Fig. 2 shows that host A and host B have almost the same position in the sixth interval). This might happen because there exist several sets of traffic loads that are mapped to the same 2D location. We resolve this ambiguity through user interaction: by moving the mouse over a node a detail view is triggered (see Fig. 3). Alternatively, the so-called dimension nodes can be moved using drag & drop to estimate their influence on a particular node or a whole group of nodes. A simple click on a dimension node results in highlighting all observation

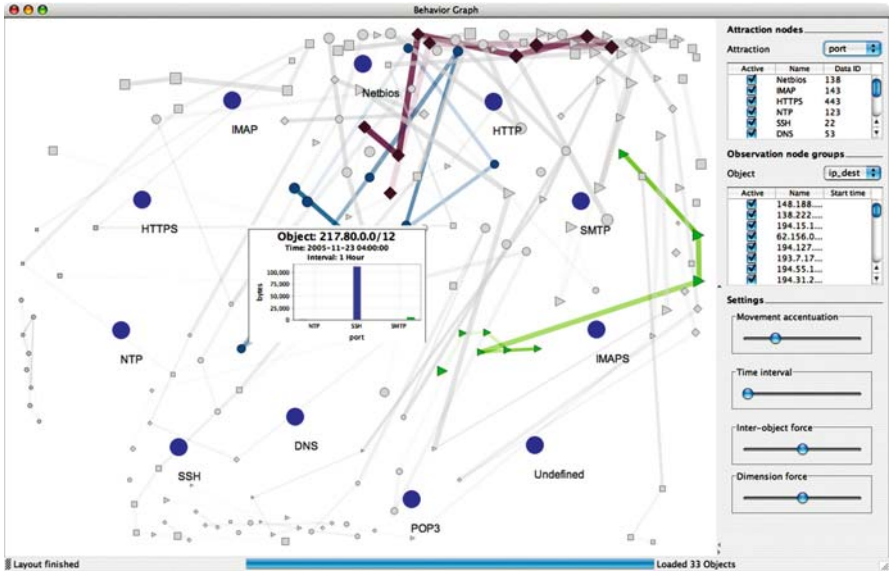


Fig. 3 Host behavior graph showing the behavior of 33 prefixes over a timespan of 1 h. Interaction is used as a means to retrieve traffic details for a particular node (bar chart in the middle). The user has selected three prefixes to trace their behavior. The configuration panel on the right allows for fine-tuning the graph

nodes containing the respective traffic. This highlighting is realized by coloring all normal nodes in grayscale while showing the highlighted nodes in color. Using the configuration panel, further *dimension nodes* and *observation node groups* can be added to or removed from the visualization.

Because we carefully designed our application for a multitude of analysis scenarios, the user can flexibly choose the attributes representing attraction nodes and observation node groups depending on the available data in the considered data set. To abstract from the technical details, he can simply select from the available data attributes in the two drop-down menus shown in Fig. 3.

In addition to this, the configuration panel has four sliders: (a) The movement accentuation slider highlights suspicious hosts with highly variant traffic. Further details about this are given in Sect. 3.5.2. (b) The second slider controls the number of observation nodes by increasing or decreasing the time-intervals for aggregating traffic. Changing the granularity of time intervals is a powerful means to remove clutter (less nodes due to larger time intervals) or to show more details (more nodes) to understand traffic situations. (c) Since each distinct node represents the state of a particular host during a time interval, we use edges to enable the user to trace a node’s behavior over time. However, following these edges can become a challenge since nodes can end up in widely varying places. In order to make these observation node groups more compact, additional attraction forces can be defined on neighboring nodes of a chain. The strength of these host cohesion forces can be fine-tuned

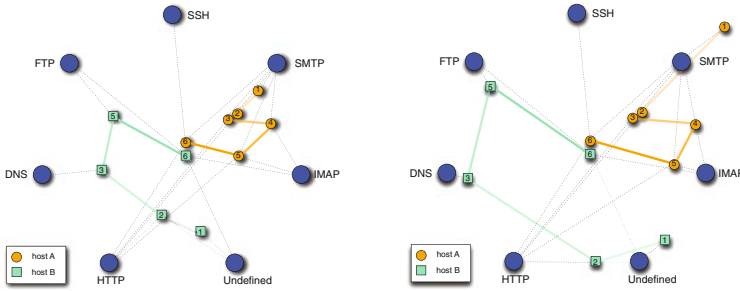


Fig. 4 Fine-tuning the graph layout through cohesion forces between the trace edges can improve the compactness of traces

with the third slider. Figure 4 demonstrates the effect of changing the forces. (d) Last, but not least, the attraction forces between observation and dimension nodes play an important role to ensure interpretability of the graph. Too strong attraction forces result in dense clusters around the dimension nodes, whereas too weak attraction forces result in ambiguity when interpreting traffic proportions since repulsion forces among observation nodes push some nodes closer to unrelated dimension nodes.

3.4 Abstraction and Integration of the Behavior Graph in HNMap

We previously presented the HNMap as a hierarchical view on the IP address space (Mansmann et al., 2007). Hosts are grouped by prefixes, autonomous systems (ASes), countries, and continents using a space-filling hierarchical visualization. This scalable approach enables the analyst to retrieve details about a quantitative measure of network traffic to and from hosts in the visualization using the above mentioned aggregation levels.

Figure 5 shows the HNMap on the AS level. Through the pop-up menu, a behavior graph for any one of the shown ASes can be displayed. Since detailed information to build up the behavior graph is available for all child levels, the user is free to choose the appropriate one. Note that only the lowest two levels of details are available since the selected node (red node at the upper left corner of the pop-up menu) is an AS node. The higher level behavior graphs can be triggered in less granular HNMap views.

While the behavior graph on prefixes, ASes, countries, or continents represents less detailed information about the particular substructures of the internet, it has proven to be beneficial since these aggregated behavior graphs significantly reduce the information overload that a network administrator needs to face when dealing with large-scale network traffic monitoring. Hence, finding the relevant subset using HNMap in combination with aggregation in detail views can be seen as possible solutions to get hold of scalability problems.

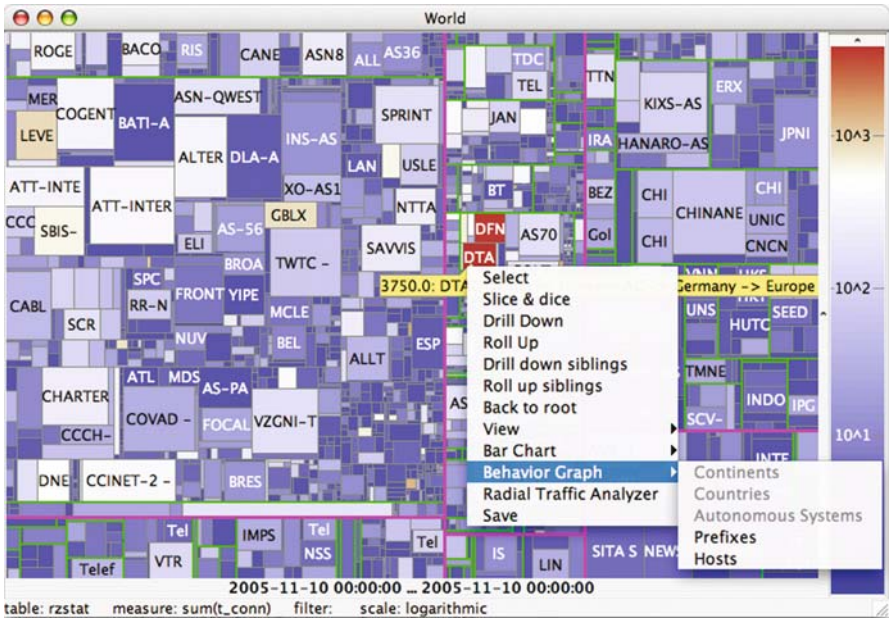


Fig. 5 We integrated the behavior graph into the HNMap visualization system. The behavior of the selected HNMap rectangle is presented by showing its child nodes (e.g., hosts, prefixes, ASes, countries, or continents) instead of being limited to the lowest-level host behavior

3.5 Application and Evaluation

In order to demonstrate the tool’s capabilities, we present a brief case study using SNORT intrusion detection alerts followed by a demonstration of the automatic accentuation visual analytics feature, which is demonstrated on the basis of traffic measurements from our university gateway.

3.5.1 Case Study

For this case study, we evaluated the 19,000 alerts generated by a SNORT intrusion detection sensor within our university network in slightly more than 2 days. The alerts referred to 17 hosts that scanned the network or generated suspicious network traffic. The attraction nodes were in this case not initialized with application port numbers, but rather with the 15 most prominent SNORT alerts of our data set and an “Undefined” traffic node for the remaining 71 rarely occurring alerts.

Figure 6 shows the outcome of our behavior graph. Larger nodes indicate a higher number of alerts and thus helps us to quickly identify the most actively attacking nodes. The graph layout generated a number of homogeneous and heterogeneous clusters. Color and shapes make nodes of different observation groups more distinguishable.

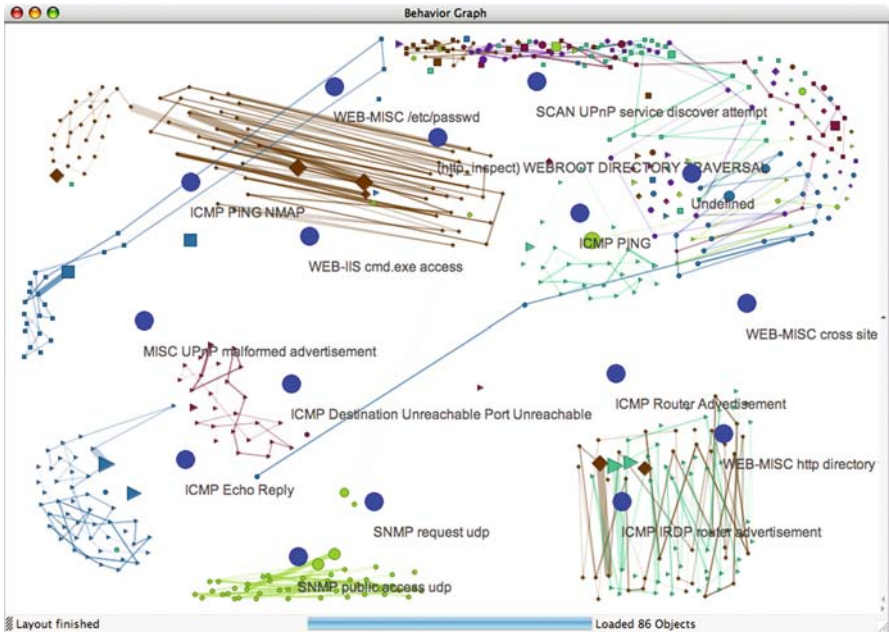


Fig. 6 Evaluating 2 days of SNORT intrusion detection alerts with the behavior graph

During interactive exploration, we discovered that the brown host on the upper left was continually scanning using ICMP PING NMAP and ICMP PING. Furthermore, the green host close to the middle and below “Undefined” generated about 1,000 alerts of various kinds, probably actively scanning the sensor machine for vulnerabilities. The two hosts with router advertisements in the lower left corner are actual routers of the network and the alerts were only generated because the SNORT sensor configuration did not exclude them.

3.5.2 Automatic Accentuation of Node Groups with Highly Variable Traffic

When regarding the behavior graph, clusters immediately stand out. However, in many scenarios the analyst is rather interested in nodes with highly variable traffic, or in other words, nodes that jump from one place to the other in the visualization. Since our visualization spans up a n -dimensional metric space, it is possible to calculate the normalized positional changes pc_{norm} of all t subsequent observations of a host in this Euclidean space:

$$\vec{o}^r = \frac{\vec{o}}{|\vec{o}|}, \tag{1}$$

$$pc_{norm} = \frac{\sum_{t=1}^{t_{max}-1} |\vec{o}_t^r - \vec{o}_{t+1}^r|}{t_{max}}, \quad 0 \leq pc_{norm} \leq 2. \tag{2}$$

Note that we first need to calculate the relative position \vec{o}^r of an observation node – our graph layout tries to place nodes with identical relative positions close to each other. Then, after calculating pc_{norm} for every node observation group, it becomes trivial to accentuate the groups with the highest values. The bounds of pc_{norm} can be explained through the fact that any component of vector \vec{o}_i^r is counted twice, once in the calculation of the difference between \vec{o}_i^r and \vec{o}_{i+1}^r and once between \vec{o}_i^r and \vec{o}_{i-1}^r .

To demonstrate the capabilities of our tool in a reproducible way, we used traffic measurements from our university network. In particular, we loaded all netflows passing the university gateway into the database, aggregated them the traffic /24 prefixes. An aggregating SQL query calculates the data for each node and loads it into the visualization and exploration tool. Figure 7 shows the behavior of the 96 of the /24 prefixes in the data set. Note that nodes with highly variable traffic are automatically accentuated in accordance with the outcome of our calculations.

3.5.3 Scalability

Our tool works well with approximately 1,000 observation nodes. The number of actual observation nodes depends on the number of monitored network entities, the time interval over which the data is aggregated and the monitored time span. Each one of these can be seen as a factor to estimate the number of observation

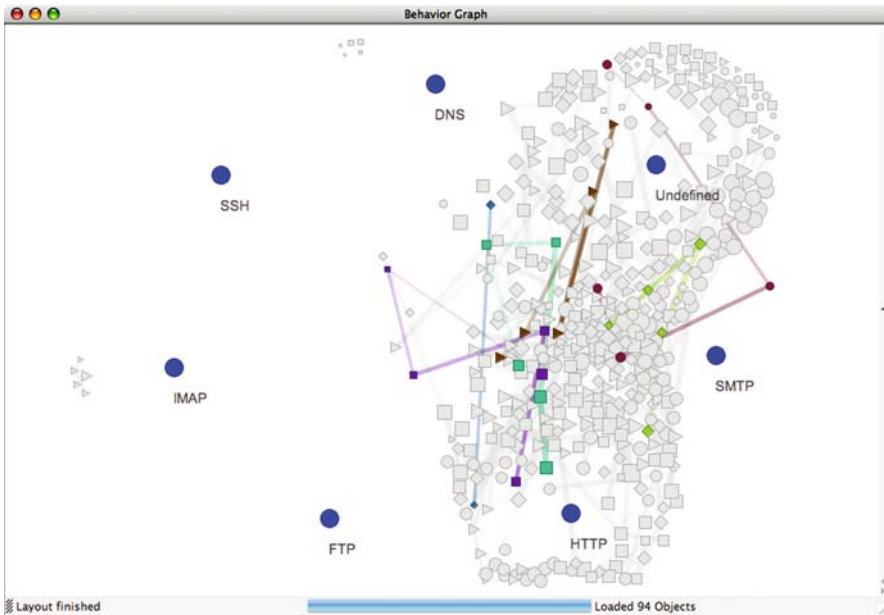


Fig. 7 Automatic accentuation of highly variable /24 prefixes using 1 h network traffic from the university network

nodes (e.g., monitoring 50 hosts over six 10-min intervals results in approximately 300 observation nodes). Above these 1,000 observation nodes, layout calculation becomes tedious and fine-tuning layout parameters turns into a challenge in itself.

4 Future Work

We noticed that it would add additional functionality if we enable users to select a certain point in time for visualization. One possibility would be to use a histogram of the amount of traffic over time. The user could then select an interval on this histogram to view the traffic. Another interesting possibility would be the option to visualize network flows in realtime with a sliding time window starting at the present and extending to some time in the past. As our layout is calculated iteratively, realtime visualization should be possible with a decent processor. Another direction for further work is the integration of an automatic dimension selection. For datasets with very high dimensionality the view gets cluttered. As our technique focuses already on a general view and not on details, it would make sense to use algorithms like PCA to eliminate dimensions which do have only a minor effect on the resulting visualization layout.

5 Conclusions

In the scope of this paper, we discussed a novel network traffic visualization metaphor to monitor host behavior. It uses an adaption of the force-driven Fruchterman–Reingold graph layout to place host observation points with similar traffic proportions close to each other. Various means of interaction with the graph make the tool suitable for exploratory data analysis.

Since our behavior graph can be used to evaluate both low-level host behavior as well as more abstract network entities, we integrated it in the HNMap tool. It can there be triggered through a pop-up menu on network entities of various granularity levels (e.g., hosts, prefixes, ASes).

The usefulness of the presented tool was demonstrated using traffic measurements from our university's gateway router and IDS alerts from a SNORT sensor. Within a brief case study, findings that can be extracted from the presented behavior graph are discussed. To equip our tool with a visual analytics feature, we introduced a normalized measure for positional changes in n -dimensional Euclidean space to automatically accentuate suspicious node groups with highly variable traffic.

Acknowledgements This work has been funded by the BW-FIT research cluster “Gigapixel displays” and by the German Research Society (DFG) under the grant GK-1042, “Explorative Analysis and Visualization of Large Information Spaces”, Konstanz. We thank the anonymous reviewers of the VizSec workshop 2007 for their valuable comments.

References

- Abdullah, K., Lee, C., Conti, G., Copeland, J.A., Stasko, J.: Ids rainstorm: Visualizing ids alerts. In: Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC). Minneapolis, USA (2005)
- Chen, C.: Information Visualization – Beyond the Horizon. 2nd edn. Springer, Berlin Heidelberg New York (2004)
- Cheswick, B., Burch, H., Branigan, S.: Mapping and visualizing the internet. In: Proceedings of 2000 USENIX Annual Technical Conference (2000)
- Claffy, K.: Caida: Visualizing the Internet. *IEEE Internet Comput.* **5**(1), 88 (2001)
- Davidson, R., Harel, D.: Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.* **15**(4), 301–331 (1996). DOI <http://doi.acm.org/10.1145/234535.234538>
- Eades, P.A.: A heuristic for graph drawing. In: *Congressus Numerantium*, vol. 42, pp. 149–160 (1984)
- Eick, S.G.: The Visualization Handbook, chap. Scalable Network Visualization, pp. 819–829. Elsevier, Amsterdam (2005)
- Fink, G.A., Muessig, P., North, C.: Visual correlation of host processes and network traffic. In: Proceedings of IEEE Workshop on Visualization for Computer Security (VizSEC), pp. 11–19 (2005)
- Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Software – Practice and Experience* **21**(11), 1129–1164 (1991). URL citeseer.ist.psu.edu/fruchterman91graph.html
- Goodall, J.R., Lutters, W.G., Rheingans, P., Komlodi, A.: Focusing on context in network traffic analysis. *IEEE Comput. Graphics Appl.* **26**(2), 72–80 (2006). DOI <http://doi.ieeecomputersociety.org/10.1109/MCG.2006.31>
- Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **31**(1), 7–15 (1989). DOI [http://dx.doi.org/10.1016/0020-0190\(89\)90102-6](http://dx.doi.org/10.1016/0020-0190(89)90102-6)
- Keim, D., Ward, M.: *Visual Data Mining Techniques*, pp. 403–427. 2nd edn. Springer, Berlin Heidelberg New York (2002)
- Lakkaraju, K., Bearavolu, R., Slagell, A., Yurcik, W., North, S.: Closing-the-loop in nvisionip: Integrating discovery and search in security visualizations. In: Proceedings of IEEE Workshop on Visualization for Computer Security (VizSEC) (2005)
- Lau, S.: The spinning cube of potential doom. *Commun. ACM* **47**(6), (2004)
- Lee, C.P., Trost, J., Gibbs, N., Beyah, R., Copeland, J.A.: Visual firewall: Real-time network security monito. In: Proceedings of IEEE Workshop on Visualization for Computer Security (VizSEC), pp. 129–136 (2005)
- Livnat, Y., Agutter, J., Moon, S., Erbacher, R., Foresti, S.: A visualization paradigm for network intrusion detection. In: *IEEE Information Assurance Workshop*, pp. 92–99 (2005)
- Mansmann, F., Keim, D.A., North, S.C., Rexroad, B., Shelehedal, 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), (2007)
- Marchette, D.J.: *Computer intrusion detection and network monitoring – a statistical viewpoint. Statistics for Engineering and Information Science*. Springer, Berlin Heidelberg New York (2001)
- McPherson, J., Ma, K.L., Krystosk, P., Bartoletti, T., Christensen, M.: Portvis: a tool for port-based detection of security events. In: Proceedings of ACM Workshop on Visualization and Data Mining for Computer Security, pp. 73–81. ACM Press, New York, NY (2004)
- Muelder, C., Ma, K.L., Bartoletti, T.: A visualization methodology for characterization of network scans. In: Proceedings of IEEE Workshop on Visualization for Computer Security (VizSEC). Minneapolis, USA (2005)
- Munzner, T., Hoffman, E., Claffy, K., Fenner, B.: Visualizing the global topology of the mbone. In: *IEEE InfoVis. IEEE Computer Society, Los Alamitos, CA, USA* (1996)

- O'Madadhain, J., Fisher, D., Smyth, P., White, S., Boey, Y.B.: Analysis and visualization of network data using jung. *J. Statist. Software* (2007). URL <http://www.jstatsoft.org/>
- PostgreSQL Global Development Group: PostgreSQL (2007). <http://www.postgresql.org/> cited 10/09/2007
- Sourcefire: Real-time network awareness (2005). URL <http://www.sourcefire.com/products/rna.html>. Cited on 11/11/2005
- Thomas, J.: Visual analytics: a grand challenge in science – turning information overload into the opportunity of the decade. In: *Proceedings IEEE Symposium on Information Visualization (InfoVis)*, p. xii. IEEE Computer Society (2005). Keynote address
- Thomas, J., Cook, K.: *Illuminating the Path: Research and Development Agenda for Visual Analytics*. IEEE Press, New York (2005)
- Xiao, L., Gerth, J., Hanrahan, P.: Enhancing visual analysis of network traffic using a knowledge representation. In: *Visual Analytics Science and Technology (VAST)*, pp. 107–114 (2006)
- Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: Visflowconnect: netflow visualizations of link relationships for security situational awareness. In: *VizSEC/DMSEC*, pp. 26–34 (2004)

Putting Security in Context: Visual Correlation of Network Activity with Real-World Information

W.A. Pike, C. Scherrer, and S. Zabriskie

Abstract To effectively identify and respond to cyber threats, computer security analysts must understand the scale, motivation, methods, source, and target of an attack. Central to developing this situational awareness is the analyst's world knowledge that puts these attributes in context. What known exploits or new vulnerabilities might an anomalous traffic pattern suggest? What organizational, social, or geopolitical events help forecast or explain attacks and anomalies? Few visualization tools support creating, maintaining, and applying this knowledge of the threat landscape. Through a series of formative workshops with practicing security analysts, we have developed a visualization approach inspired by the human process of contextualization; this system, called NUANCE, creates evolving behavioral models of network actors at organizational and regional levels, continuously monitors external textual information sources for themes that indicate security threats, and automatically determines if behavior indicative of those threats is present on a network.

1 Introduction

Visualization can have a central role in helping computer security analysts understand the changing state of their systems. But to take action on the basis of what they see in visual displays, analysts must be able to do more than just perceive anomalous changes; they must be able to *explain* them. Analysts also need the capacity to be proactive. Ideally, they should be able to identify potential security threats before an attack is incipient. Explaining and predicting security events often require more than just network- or host-centric information. External information such as exploit and vulnerability reports from other organizations, security advisories and even news stories are critical in helping analysts put anomalies in context.

W.A. Pike, C. Scherrer, and S. Zabriskie
Pacific Northwest National Laboratory, MSIN K7-28, P.O. Box 999, Richland, WA 99352, USA,
e-mail: william.pike@pnl.gov, chad.scherrer@pnl.gov, sean.zabriskie@pnl.gov

We introduce a security visualization and analysis approach called NUANCE, which results from extensive engagement with practicing analysts on computational methods that help them put their network observations into real-world context. NUANCE abstracts packet-level data to higher-level behavioral models for each IP address and group of addresses (such as an organizational subnet) observed on a network. Through a new heterogeneous data integration approach, NUANCE then visually fuses these behavioral models with contextual information on current threats and exploits from open sources. Our approach automatically collects this contextual information and determines to which actors or groups observed on a network (e.g., particular IP addresses, organizational units, or geographic areas) it is relevant. Creating, maintaining, and applying contextual knowledge of the threat landscape is the analyst's stock in trade, and NUANCE automates and scales up this practice. By seeking out context, NUANCE results in a lower information processing burden on the analyst and can direct the analyst's attention to activities most worthy of scrutiny.

We also describe the process of working with practicing analysts to develop and refine our visualization techniques. Through a user-centered process that included a series of formative design and evaluation workshops, we produced a set of visual interfaces that help analysts identify and explain off-normal activities. These workshops motivated our research by framing the questions that analysts ask in the course of investigating anomalies and the work practices that visualization systems need to support.

2 Related Work

Current network monitoring tools can produce potentially overwhelming volumes of data (Conti et al., 2006). To improve the detection of malicious events, and make best use of innate human abilities to integrate observations into explanations, there is a critical need for visual analysis techniques with which security professionals can efficiently interrogate this data. Moreover, visualization can be coupled with more sophisticated data pre-processing steps such that the human analyst is brought into the analysis loop at the appropriate point; rather than creating synoptic visual displays of all activity, for instance, we can first employ data reduction techniques that generate cognitively appropriate visual displays of the highest-value information.

2.1 The Importance of Maintaining Context

Human cognition is fundamentally about putting information in context, and it is context that allows analysts to make judgments about the meaning and importance of observed events. Just as real-world context guides pre-attentive selection of salient

features in the physical environment (Barsalou et al., 1993), contextual selection should extend to visualization environments. To help analysts deal with volumes of alerts and warnings, context-aware systems can triage these events in the same way a human would.

In an examination of the work practices of security analysts, Goodall et al. (2004) found that analysts often use online contextual information sources to keep up on the rapidly changing security landscape. Thompson et al. (2006) suggest that text information, including data from public information sources, be included in visualization applications for network security. We came to a similar finding during our own workshops (see Sect. 3), where analysts reported that there was more useful information online about potential threats than they had time or capacity to process.

2.2 Visualizing Packets and Flows

Many contemporary visualization approaches for network analysis problems focus on node connectivity and traffic flow (e.g., Paulson, 2004; Krasser et al., 2005) or on firewall alert and packet-level visualization (e.g., Conti et al., 2006). NVisionIP (Lakkaraju et al., 2004) exemplifies visualization of the lowest level of network flow data, providing a broad overview of traffic volume. Erbacher et al. (2002) produce an aggregate display that emphasizes higher-level behaviors, such as traffic load and connection patterns to individual systems on a network, and demonstrate that visualization can help analysts detect “interesting” cases quickly.

A primary concern with synoptic visualization of traffic, however, is that subtle nuances in traffic rate, destination, or type can go unnoticed. Kafadar and Wegman (2006), for instance, simply characterize “exotic” traffic as that with particularly high IP address or port frequencies, despite the risk that threats may also be carried in rare, sparse traffic. To address these subtleties, Wegman and Marchette (2003) call for “evolutionary graphics” capable of better communicating traffic change over time.

Ko et al. (1993) provide a foundation for tracking the temporal evolution of network actors, attempting to find the same actor in reports from multiple network resources. Our approach is to aggregate low-level log data to broader behaviors that characterize an actor (malicious or otherwise) over time – while allowing the end user to drill-down to individual records when needed. This technique reduces, in part, the data overload problem by increasing the level of abstraction at which network data can be analyzed. Our behavior models can also help express the temporal patterns and periods in traffic that are important – yet generally overlooked – in anomaly detection (Cordella et al., 2005). Leckie and Yasinsac (2004) take a similar behavior-centered approach, characterizing the activity of actors on a network based on the spread of their sessions during six time periods throughout the day; our technique, detailed in Sect. 4, uses a more precise continuous activity model.

2.3 Visualizing Correlated Activity

In response to the heterogeneous nature of data produced by distributed network sensors, attention has recently been devoted to visual environments that synthesize disparate information into coherent views. SecureScope (D'Amico and Larkin, 2001), for instance, helps situate cyber attacks in organizational and geographic context by visualizing features such as the mission components affected and the geographic location of critical assets. VisAlert (Foresti et al., 2006) visually links the “what, where, and when” of alerts from network sensors, although it depends wholly on the user to detect malicious activities. Hertzog (2006) visualizes relationships between users, applications, ports, and external hosts using parallel coordinate plots; large numbers of outgoing connections are grouped effectively in these graphics, allowing the analyst to readily detect cases where an application is communicating on an unusual port.

Some work on detecting correlates for cyber attacks is motivated by the subtle nature of insider threat. For instance, the vast majority of insider attacks demonstrate extensive advanced planning (Randazzo et al., 2004). Such cases exemplify the need for contextual information – in this case, information on behavioral changes, information access patterns, and so on – that can help detect nascent threats before they are executed. To this end, Stolfo et al. (2003) evaluate email flow characteristics (such as attachment frequency and time of use) to detect changing social cliques and policy violations. A natural extension of this work would apply text analysis techniques, as we do, to identify themes in traffic content that indicate concern.

From a data reduction perspective, there are existing techniques for correlation of multiple logs for improved anomaly and intrusion detection; EMERALD (Porrás et al., 2002), for instance, fuses events at the alert level using Bayesian networks. Such approaches provide a lower-volume, higher-value information stream to the analyst. However, automated correlation of structured log data with unstructured contextual information remains a research challenge, and it is this problem that NUANCE helps address. Furthermore, despite recent advances in visualizing the state of network activity, most visual tools are still reactive – they are only effective at discovering attacks that are incipient. But there is a valuable role for visualization in helping analysts forecast potential events of concern in advance of their observation on the network; a system that processes online discussions can help analysts find out about security events that others are experiencing, for instance. Making use of real-time, open source reporting is especially important in effective notification of, and response to, zero-day attacks.

3 Technical Approach

To establish requirements for visual aids to contextualization, we engaged a group of seven practicing security professionals from our organization in brainstorming, design, and evaluation workshops over the course of a year. The goal was to

understand the kinds of “world knowledge” analysts apply, the nature of the external information sources they use, and their desiderata for visual interaction and discovery mechanisms. While the lessons derived from these sessions drove the design requirements for the NUANCE tools we present in Sect. 4, they also provide general guidelines for visual analysis in computer security applications. Below, we summarize the requirements derived from these sessions, which fell into two categories: understanding context and understanding behavior.

3.1 “I Just Want to Know Where to Focus My Time”

The primary concern that emerged from our workshops was that analysts need help looking beyond anomalies; they want to detect unwanted activity, which they distinguished from the merely anomalous. Said one participant, “my work is exploratory and creative by nature, not engineered or planned.” Signature-based approaches to anomaly and intrusion detection, while an important element of their arsenal, did not always support the need for open-ended discovery. And while the analysts had tools at their disposal that supported unstructured analysis, one challenge for them was knowing where to start looking; they needed the tools to incorporate “focusing” aids that suggested high-value exploration paths.

One of the primary drivers of exploration, we found, was the world of external information that analysts used to keep aware of the changing security landscape. Typically, analysts monitored sources like the SANS Internet Storm Center and US-CERT advisories, but each also defined an idiosyncratic set of online resources that they found helpful in identifying activities to be on the lookout for. Known malicious actors and traffic could be netted by existing signature-based tools, but they were concerned over learning about new exploits for which signatures did not exist.

Our analysts suggested that one technology that would benefit them was a way to associate key words in the online text sources they read with events in the log files they explore. They wanted to be able to quickly find, for event patterns of interest in their logs, any available information online that could help explain them. They suggested that “off-network” text information is even more important in the area of insider threat. Much of the forewarning of insider events was described as “soft”: reports from other staff or changes in the style, content, or pattern of communication. As with external threats that might begin with a reconnaissance phase, insider activities are marked by a number of elements that happen before the event, none of which would be captured in the logs of current monitoring tools. The same content-based approach to associating terms in external text sources with network events could be used for insider threat mitigation if it is applied to network traffic content. While insider threat detection is outside the scope of the present work, the basic techniques we develop could be extended to this area.

A motivating example from finance emerged from one of our workshops to illustrate the role of context in visual analysis. A company’s stock price is a summary signal for the behavior of the people and institutions that own it. Changes in the

signal reflect changes in behavior – buying and selling. News stories about that company can serve as both *indicators* and *drivers* of changes in the stock price signal. As an indicator, for instance, a news article about a company's woes might coincide with an observed downward trend in its stock price. As a driver, an article about the unexpected release of a new product might motivate buyers to purchase the stock, raising its price. And just as a market analyst cannot effectively act if he or she only sees one or the other component of the system (either the price history or the contextual articles), a security analyst needs both signal and context to understand the state and trend of the network. In particular, the better the analyst can understand the relationships between drivers and the observed signal, the better decisions he or she can make; ideally, the analyst can identify contextual events likely to influence the signal before they are reflected in that signal. To proactively respond to potential threats, the security analyst needs to be aware of the current threat landscape and must understand the relationships between those threats and his or her own systems. Visualization can be effective in communicating associations between the two.

3.2 “We Need to Organize Our Hay into Smaller Piles”

The second major area of concern for analysts in our workshops was developing a greater understanding of the nature of the behaviors on their networks. Unsatisfied with tools that gave them a broad overview of the “haystack” that was their network, our analysts were looking for data reduction techniques that increased the information density of their visual displays. This “smaller haystacks” approach, if coupled with contextual information that could at least point them to the right haystack, as they said, would result in efficiency gains in finding the needles of malicious activity.

The first approach to data reduction that the analysts wanted to see was the ability to reduce massive amounts of transaction level data down to a set of behaviors that represent the trends in those transactions over time. These behavior-based views would help them see what normal activity from an individual IP address, group, or location looked like, and would therefore let them detect off-normal conditions more easily. A visual display should show the analyst whether current activity for given actor is within its expected behavior, but can do so at a high level that does not clutter the display with lower-level transaction information (although this should be available through drill-down).

Once the baseline models had reduced large amounts of transaction records to a more succinct behavior, the analysts were looking for tools that would organize these behaviors into “cliques” that would help them detect distributed but related activities. These cliques would be produced at multiple levels of granularity, from the individual IP address, to the organization (company, university, and so on), and region (city, province, country). Organizational cliques can help detect coordinated activities that derive from a location such as an internet café, even if the particular IP addresses involved change over time. In addition to creating cliques on the basis of

shared network attributes, cliques should be created that reflect common behaviors regardless of their location or organization of origin.

Understanding both behavior and context helps analysts achieve what they stated as their ultimate goal, which was to be more proactive in their work. A challenge for visualization, they told us, was to give them the flexibility to be both predictive and reactive; to look ahead for events they *might* experience and to look backward for forewarning they could have had about events they *did* experience.

3.3 Behavior Modeling

The NUANCE visualization approach that emerged from the requirements gathered during our analyst workshops fuses behavioral analysis with contextual information. This section describes the resulting behavior modeling and visualization technique; Sect. 5 introduces our contextual analysis methods.

Rather than build visualizations for individual network transactions, our work takes the approach that what is important is anomalous or malicious *behaviors*, which may be manifested in a series of transactions over time (from seconds to days or even years). NUANCE uses a hierarchical modeling approach to represent actor behavior. At the most granular level is the IP address; a behavioral model can be constructed for every IP and group thereof. NUANCE defines group membership through geographic and organizational attributes retrieved through who is and gazetteers, although custom groups could be created by the analyst. (We note that NUANCE itself does not reconcile spoofed addresses, although it can make use of existing traceback techniques). Groups can also be created at the port level. Sample groups might include “China”, “Zhejiang Province”, “XYZ University” and “FTP traffic from XYZ University”. When a new actor appears on a network (e.g., an IP address that has not been observed before), the hierarchical modeling approach allows it to be assigned baseline behavioral models based on its organization and region until sufficient history has been observed to generate a model of its own.

Because baseline activity for “normal” behavior is not assumed to be the same for all IP addresses, the analyst can examine behavioral profiles aggregated to any group of users – important when trying to detect external threats where the attacker changes IP address over time. We can look at behavior from a particular place, for instance, and note when that origin location’s behavior changes, regardless of which actors are responsible for the change (since some of the actors may be “new” IP addresses).

3.3.1 Model Definition

The NUANCE behavior modeler consists of parsing, statistics, and curve-fitting components. The parser receives real-time network data (such as from NetFlow logs, although any log data can be used) and summarizes each record as a timestamp and a list of groups to which the transaction belongs. Group membership is assigned by

rules such as “Is this packet from a .edu domain?”, “Is this an HTTP packet?”, or “Is this an HTTP Packet from ABC Co.?” The statistics thread monitors output from the parser and maintains an array of sufficient statistics for each IP address and group (hereafter we use the term “actor” to describe both unique IP addresses and groups thereof). For a given actor, our statistical model expresses the expected traffic rate over time as a periodic function. The assumption of periodicity suggests the use of a Fourier series, and to ensure that the expected traffic rate is never negative, we use an exponentiated Fourier series.

Let k be a vector of periods of interest, and let K be the least common multiple of the k_i 's; we currently use $k = (6, 8, 12, 24)$, in units of hours, although k can include any time periods of interest, from minutes to years. Our model is then that session time modulo K is random, with a density function we will now describe.

For any real-valued parameter vectors α and β , we can write the series

$$\varphi_{\alpha\beta}(t) = \sum_i \left(\alpha_i \cos \frac{2\pi t}{k_i} + \beta_i \sin \frac{2\pi t}{k_i} \right).$$

Since $\exp\{\varphi_{\alpha\beta}(t)\}$ is always positive, we can normalize it to arrive at a density function

$$f_{\alpha\beta}(t) = \frac{\exp\{\varphi_{\alpha\beta}(t)\}}{I_{\alpha\beta}},$$

where $I_{\alpha\beta}$ is the normalizing constant

$$I_{\alpha\beta} = \int \exp\{\varphi_{\alpha\beta}(t)\} dt.$$

Every choice of α and β leads to a density $f_{\alpha\beta}(t)$. Now suppose for a given actor we observe transactions starting at times $\{t_1, \dots, t_N\}$, and we wish to choose α and β to best fit the observed data. This we can do using maximum likelihood estimation; we consider $\Pi_n f_{\alpha\beta}(t_n)$ as a function of α and β (since the t_n 's are now fixed), and find the maximum value. The values of α and β leading to this maximal value are then the maximum likelihood estimates of the parameters, and are denoted $\hat{\alpha}$ and $\hat{\beta}$, respectively.

Typically, rather than maximize $\Pi_n f_{\alpha\beta}(t_n)$ directly, it is more convenient to maximize its logarithm. This leads us to the log likelihood

$$\begin{aligned} l(\alpha, \beta) &= \sum \log f_{\alpha\beta}(t_n) \\ &= \sum \varphi_{\alpha\beta}(t_n) - N \log I_{\alpha\beta} \\ &= \sum (\alpha_i c_i + \beta_i s_i) - N \log I_{\alpha\beta}, \end{aligned}$$

where c_i and s_i are the statistics

$$\begin{aligned} c_i &= \sum \cos \frac{2\pi t_n}{k_i} \\ s_i &= \sum \sin \frac{2\pi t_n}{k_i}. \end{aligned}$$

In particular, $(N, (c_i), (s_i))$ constitutes a sufficient statistic. Thus as new data arrive, the number of values that must be stored for estimation remains constant.

For each actor, the sufficient statistics allow us to fit a curve representing the traffic rate as a function of time, with multiple time scales taken into account simultaneously. Once the density function is determined, we can use it to estimate the expected traffic rate at a given point in time. Suppose observed times span from t_0 to t_1 . Our estimated rate can be found using the criteria

$$r(t) = \gamma f_{\hat{\alpha}\hat{\beta}}(t)$$

$$\int r(t)dt = N.$$

Here γ is a normalizing constant. We then have

$$N = \int r(t)dt = \gamma \int f_{\hat{\alpha}\hat{\beta}}(t)dt,$$

so the estimated traffic rate for each actor is

$$r(t) = \gamma f_{\hat{\alpha}\hat{\beta}}(t) = \frac{N f_{\hat{\alpha}\hat{\beta}}(t)}{\int f_{\hat{\alpha}\hat{\beta}}(t)dt}.$$

The NUANCE statistical methods result in evolving behavioral models for each actor on a network. Figure 1 shows a sample of behavioral profiles using this modeling approach. Each colored curve represents a unique actor. Some actors demonstrate characteristically “bursty” behavior, such as the profile labeled “A”, engaging in short sessions of traffic at various times throughout the day. Others exhibit continuously high levels of traffic, such as the profile labeled “B”; this is characteristic of a server, or, if the IP address is external, of a search engine crawler. A low-and-slow port scan would appear “bursty” in this model, because the traffic would appear as short periods of activity separated by long periods of quiet.

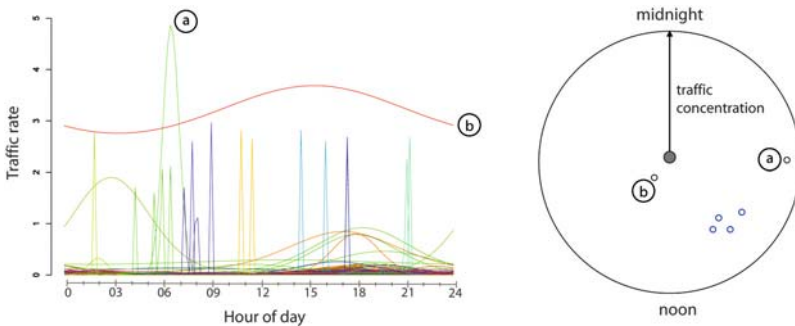


Fig. 1 (left): Diurnal behavior profiles (traffic over time) for a selection of actors; each curve represents a distinct actor, whose activity over all ports is aggregated (port-specific models can be constructed). *Y*-axis scale is packets per second. (right): Visual transformation of profiles to “clock” view allows cohorts of similar actors (blue) to be detected

In the special periodicity case $k = (2\pi)$, an actor's traffic function reduces to the density for a von Mises distribution, which is a continuous distribution describing a set of points situated on a circle. This von Mises plot (at the right of Fig. 1) represents temporal information, with time moving clockwise around the plot, and provides an even higher-level summary of the nature of each actor's behavior. Actors who typically engage in brief bursts of activity (e.g., A) plot nearer to the perimeter of the circle, while those who are more continuously active over the course of time period being examined (e.g., B) plot closer to the center. The position of the actor around the "clock" indicates its typical peak activity time (e.g., 6:00 am for actor A). The von Mises plot in Fig. 1 shows 24 h of activity, but shorter or longer time periods can be selected. The von Mises distribution dramatically reduces the storage and processor time required to generate behavioral models (only three values are required to express each actor's diurnal traffic patterns), allowing models to be constructed on the fly for large segments of the IP address space. Moreover, actor models can be clustered with K-means. These clusters represent behavioral cliques, and contain actors who, regardless of organization or geographic location, exhibit similar behavioral profiles. This clustering approach is useful for detecting distributed attacks that show the same "modus operandi" but would not otherwise be associated into the same actor group. It is also possible to detect actors who change behavioral clique over time, which is often an indicator for insider threat.

We have currently modeled up to 100,000 unique actors successfully in real-time on commodity hardware, and are working to increase by an order of magnitude the number of models the system can process. One limitation to the number of models that can be accurately constructed is that the curve fitter cycles independently over the actors, updating fitted curves as it goes. As new transactions arrive, we walk the model list and tune each model in sequence. As a result, as the number of behavioral models to be stored increases, the time to complete the walk increases and the accuracy of the resulting models decreases. This limitation can be mitigated by parallelizing the curve fitter.

3.3.2 Dynamic Histograms

To detect off-normal behavioral conditions, we compare the activity models described above with empirical traffic rates. When observed traffic varies from the predicted behavioral model by a user-specified threshold, the difference can be visualized for the analyst and appropriate contextual information retrieved. A natural approach for representing observed traffic rates is to use a histogram of transaction (e.g., packet or session) counts. However, in storing these histograms there is a trade-off between high resolution and long memory. As with the modeling component, we aim to maintain constant space (i.e., not infinitely increasing the storage and processing requirements as the amount of data increases), so we require the number of bins in the histogram to be fixed.

To accommodate this trade-off, we maintain high temporal resolution for recent data and drop the usual implicit assumption that bins are adjacent. Each bin in a

NUANCE histogram represents 1 min of activity from each actor for whom a model is also being built. Each minute a new bin is introduced and an older bin chosen at random is dropped. Thus the number of chances a bin has had to be dropped is proportional to its age, so older bins are progressively farther apart. The result is a time-decayed histogram, representing more recent activity on a finer time granularity than older activity. Our approach allows observed data to age in this way with very little overhead.

3.4 Building Context

NUANCE also introduces a new method for associating contextual content with network behaviors. This process involves gathering and filtering text data to construct a vocabulary that describes each actor and group being modeled, a monitoring component that collects real-time content from sources of the analyst's choosing, and classification routines that determine, on the basis of their vocabularies, to which actors incoming content is relevant.

3.4.1 Vocabulary Construction

Using the same metadata by which IP addresses are assigned to groups (organization name, geographic area, and so on), NUANCE actively constructs a text vocabulary to describe each IP and group. The vocabulary is generated by performing an automated web search around those metadata terms; the result of this harvest is a corpus of training documents representative of the actor (alternatively, training documents can come from the content of traffic in which that group is involved). These documents are merged into a single term list, processed for stopwords (terms to exclude) and major terms (frequent, statistically descriptive terms), and converted into a vector that represents topics characteristic of the group. The value of the vocabulary construction approach is that we can make associations between text documents such as security notices and group activity on the basis of more than just keyword matching. That is, we do not look for the term "Philippines" in a news article and automatically associate that article with our traffic model for the Philippines. Instead, the web harvest builds a more complete representation of the topics relevant to the Philippines (and can be tailored to favor security-related web sites over general information sites). The NUANCE vocabulary allows notices of exploits to be associated with actors that might show evidence of using that exploit, even if the notice does not mention that actor at all (as is usually the case, when we do not know from what IP addresses to expect an attack). There is a significant bootstrapping challenge in generating these vocabularies, however, as one must be constructed – and updated – for each IP address and actor group being modeled.

3.4.2 Content Analysis

Once baseline vocabularies have been constructed for each group NUANCE has also created a behavioral model for, the user can specify a set of text feeds to monitor for computer security and geopolitical events of interest. Currently, NUANCE collects text reports every hour from a variety of security and news web sites. These sources include US-CERT, SANS Internet Storm Center, PacketStorm advisories and exploits, online bug trackers, BBC News, New York Times, and a selection of user-specifiable sites such as message boards. Typically, NUANCE ingests about 300 text reports a day, although this is a minimum and can grow as the user adds new sources to the NUANCE “reading list”. The time of publication for each text document is extracted and later used to correlate the document with network events. Just as each document in the training set for a group vocabulary is processed into a term vector, each incoming context document is processed into a similar vector.

3.4.3 Calculating Contextual Relevance

Once vectors for both characteristic group topics and incoming textual events have been created, they can be compared to determine to which groups context events are relevant. Measuring the relevance of a feed relative to a group is based on a cosine similarity metric. Each time a new contextual feed item is received, it is compared against each of the group vocabularies currently in the system. The smaller the angle between each feed item and the group vocabulary, the more similar or relevant the feed is to the group. We currently use a pre-defined similarity score threshold of 80% to determine whether a context item should be associated with a group. It is possible for a given context item to “hit” on multiple groups, meaning that there are potentially multiple behavioral events to which the context alert is relevant. It is also likely that for many incoming context documents, there will be no relevant actor; the goal is simply to create a broad harvest of current security topics so that if a relevant actor appears on the network, the analyst can be notified.

3.5 Visualizing Behavior in Context

Figure 2 shows NUANCE in operational mode. The main application allows the analyst to vertically tile and sort multiple behavioral models (three are shown in this view), facilitating comparison of activities across groups. Selections from an analyst’s “favorite” behavior models are made through a drop-down list at the top of each pane. When the system monitors large numbers of actors, analysts can call up models through a search interface. NUANCE currently defines actors automatically, aggregating transactions by IP address or group of addresses representing a geographic or organizational unit. Analysts do not have to define these actors manually, but we anticipate creating a visual interface that allows them to do so. Models

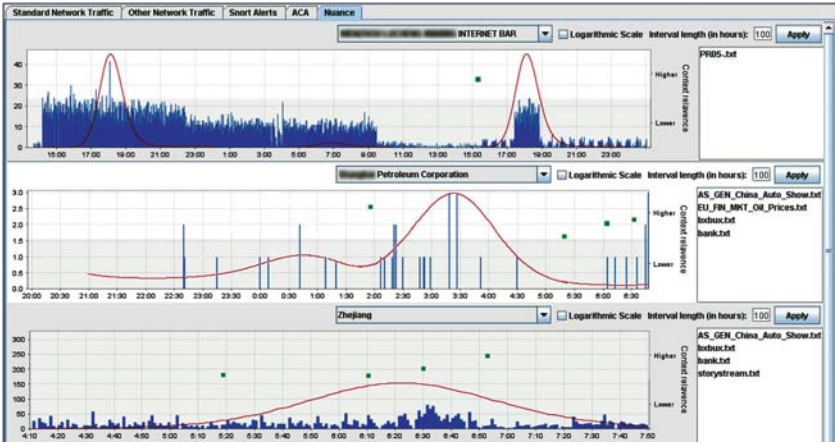


Fig. 2 NUANCE behavioral models for three actor groups. Histograms (blue) show current activity in 1-min intervals. Red curves represent expected behavior for each group. Relevant contextual information is automatically attached to the group at the time it is received (green dots). Selecting a context item in the chart or the on the list to the right of each plot opens it

can also be “pushed” onto the view based on pre-defined anomaly thresholds (i.e., observed behavior differing from expected behavior by more than a specified percentage) coupled with availability of context (i.e., groups with large anomalies plus highly relevant context).

Each NUANCE plot is essentially a timeline showing a zoomable temporal window on the *x*-axis and traffic level over that window on the *y*-axis (here, shown as packets per minute over all ports, although views could be segregated by port). The top plot in Fig. 2 shows approximately the last day and a half of traffic from a particular internet café. Histogram data shows per-minute observed traffic levels, while the red periodic curve represents the current expected behavior pattern for that group. In this example, NUANCE has been processing real-time data for about 3 days, so the models are beginning to tune themselves but have not yet reached full precision (we have found that approximately one week’s worth of data is required to maximize the modeling approach’s ability to represent observed traffic). For traffic from the internet café, the model correctly predicted the two peaks in traffic that occurred over this period, although the model did not predict the generally high level of activity over the first half of the plot. This difference can trigger a flag which adds the group to the analyst’s watch list. At the same time, incoming context feeds that met the scoring threshold for relevance to this group are displayed as green dots on the plot. The right *y*-axis describes the approximate score of each context item, with more relevant items plotting toward the top of the chart. In the case of this internet café, we can see that one contextual item has fused to the group; selecting that item (either the dot on the chart or the item title in the context list to the right of the chart) opens a reader where the analyst can view the context item. This particular context feed warned of a SQL injection attack, and it fused to this internet café because

the vocabulary for that group indicated that similar exploits had been identified as sourcing from the same location. The analyst, without knowing that traffic from this source should be of concern, now has both an understanding of how the traffic has departed from recent historical levels and, through the context item, an explanation of what that traffic may involve. Drilling down into the histogram can expose the underlying transactional information.

The behavioral models in the lower two charts (for traffic from a petroleum company in the middle plot, and from Zhejiang province in China for the lower plot) generally track their group's observed activity, and will continue to refine at each update increment. Here too, context elements help explain changes in the observed activity rate for each. In the middle chart, a news alert about rapidly rising petroleum prices in the region in which this company does business (leftmost green dot, corresponding to the "EU_FIN_MKT_Oil_Prices" story) precedes a rise, roughly 90 min later, in this organization's activity. Just as news items can help explain observed changes in stock prices, our analysts found that contextual items could often help them determine why activity changes might be taking place, leading to a better ability to triage responses.

3.5.1 Situational Awareness Dashboard

Summative evaluation from our analyst team on the NUANCE application suggests that the ability to visually relate context and actor behavior fits best in an *exploration* phase of analysis. That is, analysts can drill down through these charts to uncover the original transaction-level data that went into the histogram and modeling routines and can "play" with the data by comparing actor charts, looking up models for actors of interest, and scrolling forward and back in time to uncover subtleties in traffic.

However, analysts in our workshops wanted to complement this exploratory interface with a similar context-driven visual tool that would serve them in their *monitoring* activities. For many of our analysts, monitoring preceded exploration; they might use a suite of monitoring tools to alert them of suspicious activity over the course of the day, and if an event merited further investigation they would turn to an analysis tool to explore it further. As a result, we developed a prototype monitoring interface that reduces the NUANCE contextualization and behavior modeling approach to a simpler "one-look" view.

Figure 3 shows the enhanced NUANCE view designed to support real-time monitoring. At the center of the display is a von Mises circle depicting the past 24 h of activity (shorter and longer time periods can be selected), containing a point for each actor. Through the method described in Sect. 4.1, each point corresponds to one of the time charts in Fig. 2; the two displays can be linked such that selection of an actor in this view will bring up the relevant histogram and activity model. The location of each actor around the plot perimeter indicates its expected time of appearance, while its location along the radius indicates the nature of its activity, from "bursty" to continuous. The actor models in Fig. 3 have been clustered using K-means; each cluster is represented by a unique color in the central circle.

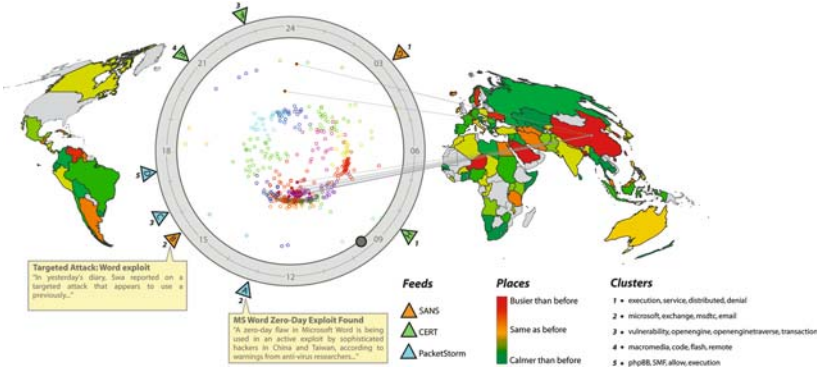


Fig. 3 NUANCE situational awareness tool. Von Mises circle at center shows a point for each actor observed over the last 24 h; colors indicate behavioral cliques. A geographic region (China) is selected, highlighting actors from that location

Surrounding the time wheel is a map that shows the analyst the level of “chatter” about geographic locations in the context feeds being monitored. As more context items mention a location, its country is colored a deeper red. In this example, China has been selected, and the actors representing traffic from China have been highlighted in the circle. Context items relevant to this geographic group then appear as flags around the perimeter of the time wheel at the time they were published. Context items are also clustered in real-time, using standard text-clustering techniques (Hetzler et al., 1998); each flag is numbered with the context cluster to which it belongs. For instance, the two highlighted context items referring to an MS Word exploit correspond to cluster 2, which contains the “Microsoft, exchange, msdtc, email” topics.

The dashboard view simplifies user interaction by restricting the amount of detail shown about each actor. By reducing each behavioral profile to a single point, more actors can be shown at once. In a single view, it is also possible to ascertain both the current geopolitical and security landscape. We are currently exploring visual interactions that help users link high-level concepts derived from the dashboard with the specific activities (as in Fig. 2) that manifest these changes.

4 Future Work

NUANCE behavioral modeling is suited equally well to modeling the activities of external hosts or machines internal to a network. In the internal case, NUANCE models can offer analysts an evolving picture of the expected state of machines on their network. One challenge in modeling external hosts is the large number of potential actors of interest; we do not currently have an adequate mechanism for enabling analysts to navigate through the models that exist and choose which

to display. An additional visual component that helps in this regard (for instance, clustering related models to provide hierarchical navigation) may be called for.

We are currently working on extending NUANCE to support predictive analysis. Once NUANCE has linked a context item with a behavioral profile, it is possible to use machine learning techniques to reinforce the association between the context topics and resulting behaviors. Then, when particular collections of context topics are observed in the future, analysts can be presented with representations of the kinds of behaviors to expect. Associating behavioral profiles with historical security events helps detect those activities that are not in themselves anomalous but that match the longer-term patterns of actors who eventually did perform malicious acts. We are also developing techniques to visualize higher-order threat models that link behaviors observed across distributed actors. This work helps overcome the difficulty in detecting multi-stage attacks emanating from spoofed sources. Finally, to address the challenges inherent in processing streaming network data in real-time, we have begun implementing NUANCE on high-performance computing infrastructures. Distributing the model generation and text classification work across a cluster can improve the accuracy of the models (by reducing the time between updates) and speed the generation of training vocabularies for contextualization as new actors are observed.

5 Conclusions

Based on requirements gathered during a series of formative sessions with practicing analysts, we have developed a new behavioral modeling and contextualization approach that helps users visualize the associations between changes in network activity and explanatory external events. Our efforts to build automated context gathering and classification methods into visualization tools give analysts an improved ability to situate threats in the real world, a practice in which they already engage but for which current tools offer little support.

The NUANCE method's behavioral modeling technique is capable of representing the periodic nature of network activity over multiple time scales, even when periods are not evenly spaced. NUANCE can also create a unique behavioral model for each actor observed on a network, allowing it to present a detailed picture of network activity. Assessing the deviation of each actor's current activity from historical precedent offers specific warning of suspicious activity.

Behavioral models are one technique for accommodating surprise, by helping the analyst understand whether what he or she is seeing fits into historical patterns. However, these models alone will never produce perfect forecasts, especially when dramatic events force a change in behavior (for instance, Hurricane Katrina fundamentally changed the nature of traffic to and from certain regions of the US). Therefore, we incorporate a text content collection and fusion technique that helps analysts discover possible reasons why traffic is behaving as it is. Ultimately, by fusing heterogeneous information sources – including those, such as news feeds,

that are part of the analyst's toolkit but typically ignored in security analysis tools – we can improve the analyst's ability to detect and respond to threats.

References

- Barsalou L, Yeh W, et al. (1993) Concepts and meaning. Chicago Linguistics Society: Papers from the Parasession on Conceptual Representations. K Beals, G Cooke, D Kathman, et al. Chicago, University of Chicago 29:23–61.
- Conti G, Abdullah K, et al. (2006) Countering security information overload through alert and packet visualization. *IEEE Computer Graphics and Applications* (March/April):30–40.
- Cordella LP, Finizio I, et al. (2005) Using behavior knowledge space and temporal information for detecting intrusions in computer networks. *Pattern Recognition and Image Analysis, Pt 2, Proceedings* 3687:94–102.
- D'Amico A and Larkin M (2001) Methods of visualizing temporal patterns in and mission impact of computer security breaches. *DARPA Information Survivability Conference and Exposition II, IEEE Computer Society*.
- Erbacher R, Walker K, et al. (2002) Intrusion and misuse detection in large-scale systems. *Computer Graphics and Applications* 22(1):38–48.
- Foresti S, Agutter J, et al. (2006) Visual correlation of network alerts. *IEEE Computer Graphics and Applications* 26(2):48–59.
- Goodall J, Lutters W, et al. (2004) I know my network: Collaboration and expertise in intrusion detection. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work, CHI Letters*, ACM Press, New York, pp. 342–345.
- Hertzog P (2006) Visualizations to improve reactivity towards security incidents inside corporate networks. *ACM Conference on Computer and Communications Security, Third International Workshop on Visualization for Computer Security*, Alexandria, VA, ACM Press, New York, pp. 95–102.
- Hetzler B, Harris W, et al. (1998) Visualizing the full spectrum of document relationships. *International Structures and Relations in Knowledge Organization, Proceedings of the Fifth International ISKO Conference*, Wurzburg, ERGON Verlag, pp. 168–175.
- Kafadar K and Wegman EJ (2006) Visualizing “typical” and “exotic” Internet traffic data. *Computational Statistics & Data Analysis* 50:3721–3743.
- Ko C, Frincke DA, et al. (1993) Analysis of an algorithm for distributed recognition and accountability. *Proceedings of the First ACM Conference on Computer and Communications Security*, Fairfax, Virginia, United States, ACM Press, New York.
- Krasser S, Conti G, et al. (2005) Real-time and forensic network data analysis using animated and coordinated visualization. *IEEE Workshop of Information Assurance and Security*, West Point, NY.
- Lakkaraju K, Yurcik W, et al. (2004) NVisionIP: Netflow visualizations of system state for security situational awareness. *ACM Conference on Computer and Communications Security, Workshop on Visualization and Data Mining for Computer Security*, Washington, DC, ACM Press, New York, pp. 65–72.
- Leckie T and Yasinsac A (2004) Metadata for anomaly-based security protocol attack deduction. *IEEE Transactions on Knowledge and Data Engineering* 16(9):1157–1168.
- Paulson L (2004) Researchers develop network-security visualization tools. *Computer* 37(4): 17–18.
- Porras P, Fong M, et al. (2002) A mission-impact-based approach to INFOSEC alarm correlation. *Fifth International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, Zurich, Switzerland, Springer, Berlin Heidelberg New York, pp. 95–114.
- Randazzo M, Keeney M, et al. (2004) Insider threat study: Illicit cyber activity in the banking and finance sector, U.S. Secret Service and CERT Coordination Center.

- Stolfo S, Hershkop S, et al. (2003) Behavior profiling of email. First NSF/NIJ Symposium on Intelligence and Security Informatics, Tucson, AZ, Springer, Berlin Heidelberg New York, pp. 74–90.
- Thompson R, Rantanen E, et al. (2006) Network intrusion detection cognitive task analysis: Textual and visual tool usage and recommendations. Proceedings of the Human Factors and Ergonomics Society (HFES 06), pp. 669–673.
- Wegman EJ and Marchette D (2003) On some techniques for streaming data: A case study of Internet packet headers. *Journal of Computational and Graphical Statistics* 12:893–914.

An Interactive Attack Graph Cascade and Reachability Display

L. Williams, R. Lippmann, and K. Ingols

Abstract Attack graphs for large enterprise networks improve security by revealing critical paths used by adversaries to capture network assets. Even with simplification, current attack graph displays are complex and difficult to relate to the underlying physical networks. We have developed a new interactive tool intended to provide a simplified and more intuitive understanding of key weaknesses discovered by attack graph analysis. Separate treemaps are used to display host groups in each subnet and hosts within each treemap are grouped based on reachability, attacker privilege level, and prerequisites. Users position subnets themselves to reflect their own intuitive grasp of network topology. Users can also single-step the attack graph to successively add edges that cascade to show how attackers progress through a network and learn what vulnerabilities or trust relationships allow critical steps. Finally, an integrated reachability display demonstrates how filtering devices affect host-to-host network reachability and influence attacker actions. This display scales to networks with thousands of hosts and many subnets. Rapid interactivity has been achieved because of an efficient C++ computation engine (a program named NetSPA) that performs attack graph and reachability computations, while a Java application manages the display and user interface.

1 Introduction

Attack graphs have been proposed by many researchers as a way to model adversary behavior, identify critical weaknesses, and suggest network changes to improve network security. Researchers and commercial companies have recently developed differing approaches to generating attack graphs (RedSeal, 2007; Skybox, 2007; Ingols et al., 2006; Noel and Jajodia, 2005; Ou et al., 2005). A review of past

L. Williams, R. Lippmann, and K. Ingols
MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02173, USA, e-mail: LCWILL@LL.MIT.EDU, LIPPMANN@LL.MIT.EDU, KWI@LL.MIT.EDU

research is available in (Lippmann and Ingols, 2005a). Attack graphs are constructed by starting an adversary at a given network location and, using information about the network topology and host vulnerabilities, examining how the attacker can progressively compromise vulnerable hosts that are reachable from already compromised hosts. Vulnerability scanners and analyses of filtering performed by firewalls and routers are used to obtain information about host vulnerabilities and to determine host-to-host reachability in a network. Almost all approaches have a method of generating recommendations to patch critical vulnerabilities or make firewalls more restrictive. In addition, most of the existing implementations provide some type of attack graph display. However, the abstract nature of attack graphs has proven to be a serious practical weakness in creating an effective display.

The rest of this paper describes an interactive tool that was designed to overcome the limitations of existing techniques. The following section gives an overview of related work. The next sections present initial design goals, some display types that were considered, and examples of our display for a small test network and a larger field trial network. This is followed by a discussion of future work. The final section concludes.

2 Related Work

2.1 Limitations of Existing Approaches

Previous attempts at effectively displaying attack graphs suffer from several fundamental limitations. First, hosts and other network assets are often positioned in a way that is unrelated to normal network diagrams drawn by system administrators. For example, typical network structures such as subnets, virtual local area networks (VLANS), and other fully connected domains are often not easy to delineate and, even when hosts in subnets are collocated, subnets are positioned in locations that are widely divergent from those that are intuitive to system administrators. Second, when an attempt is made to display a complete attack graph at once, this often creates a confusing tangle of edges that is difficult to interpret. Critical steps that allow an attacker to progress between subnets, exploit a new trust relationship, or jump into a network using a client-side attack may be missed in the explosion of edges that occurs after an attacker reaches the network interior. Finally, host-to-host reachability is not usually displayed even though this is a key factor in explaining why critical attack steps are possible and determining which filtering devices allow reachability and could be modified to prevent these attack steps.

These three limitations initially resulted from the use of automatic graph layout algorithms available in the dot application from the Graphviz software package (Graphviz, 2007; Gansner et al., 1993). Representing hosts as separate nodes and exploitation of vulnerabilities as separate edges often leads to complex graphs that are difficult to display, navigate, and interpret. A number of researchers have suggested methods to simplify these types of attack graph displays (e.g. Ingols et al.,

2006; Noel and Jajodia, 2004, 2005; Swiler et al., 2001) by grouping similar hosts together and representing grouped hosts by single nodes, and by using hierarchical displays. These approaches still result in complex attack graphs that are difficult for untrained system administrators to relate to the underlying analyzed network.

2.2 *NetSPA System*

In previous research we described an efficient approach to generating a new type of attack graph called a multiple-prerequisite (MP) graph that scales well to large enterprise networks. Descriptions of the NetSPA tool that generates MP graphs are available in (Ingols et al., 2006; Lippmann et al., 2006; Lippmann and Ingols, 2005b). NetSPA's network model supposes that an individual host possesses one or more interfaces each of which has a listening address. These interfaces also have zero or more open ports which accept connections from other hosts. Interfaces may have rules that dictate how network traffic may flow to, and through, the interface and its host. A port has zero or more vulnerability instances that may be exploitable by an attacker. Each host interface is connected to a link, representing some combination of hubs and switches connecting a set of interfaces together. An attacker is able to obtain one of four access levels on a host using a particular vulnerability: "root" or administrator access, "user" or guest access, "DoS" or denial-of-service, or "other," indicating a loss of confidentiality and/or integrity. The combination of a host and an access level is an attacker state. A state may provide the attacker with zero or more credentials; vulnerability instances may require zero or more of them. Currently, it is assumed that an attacker obtains a host's reachability if "root" or "user" access is achieved. Reachability and credentials serve as prerequisites for exploitation of a vulnerability instance. Our concept of "credential" is any information used for access control such as a password or a private key. Besides providing access to a host, a vulnerability is characterized by whether it can be exploited remotely from a different host or only locally from the vulnerable host.

An MP graph is limited in size because it contains at most one node for each vulnerability instance, host state, reachability group, and credential. Reachability groups represent collections of hosts that are treated identically by all firewalls. These are determined automatically by firewall rule analysis. Edges are contentless and three node types explicitly represent prerequisites of all attacks. State nodes represent an attacker's level of access on a particular host. Outbound edges from state nodes point to the prerequisites they are able to provide to an attacker. Prerequisite nodes represent either a reachability group or a credential. Outbound edges from prerequisite nodes point to the vulnerability instances that require the prerequisite for successful exploitation. Credentials can be used to model many types of trust relationships such as shared passwords used to administer many hosts or automated remote host control via SSH that requires only root access on an administration machine. Vulnerability instance nodes represent a particular vulnerability on a specific port. Outbound edges from vulnerability instance nodes point to the single state

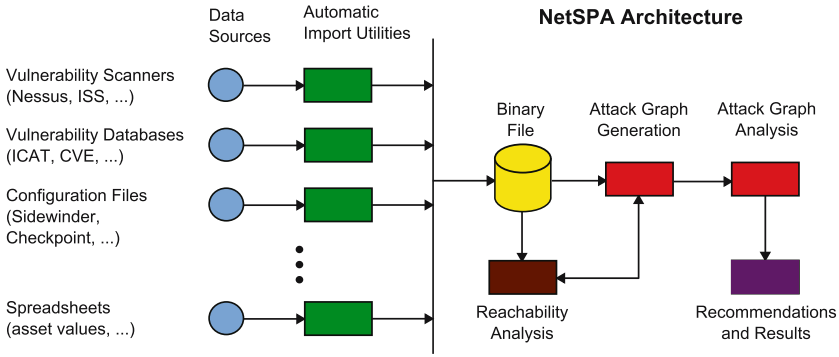


Fig. 1 System architecture of NetSPA tool

that the attacker can reach by exploiting the vulnerability. These three node types in turn define the sole ordering of paths in the graph: a state provides prerequisites, which allow exploitation of vulnerability instances, which provide more states to the attacker.

The NetSPA system is comprised of several software components. The importer, written in PERL, is responsible for reading in raw data such as Nessus scans, firewall rulesets, and NVD database records NVD, 2007, and converting the data into a custom binary file format for later use. A small C program acts as a vulnerability classifier and is designed to identify a vulnerability’s locality (remote or local access) and effect (whether root, user, DoS, or other privilege level is provided). It uses a pattern-matching algorithm that has been trained on a sample vulnerability data set. The classifier was built using the freely available LNKNet tool (Lippmann et al., 1993). The engine, written in C++, is responsible for computing reachability, generating attack graphs, and analyzing the graphs to generate recommendations. It reads the network model from the custom binary file generated by the importer. The block diagram in Fig. 1 gives an overview of the design of the NetSPA system.

Our MP graphs are currently displayed using the Graphviz dot application (Graphviz, 2007; Gansner et al., 1993). These graphs are fast to compute but difficult to interpret for all but the smallest networks. We have developed a simple algorithm to “collapse” many graph nodes together (Ingols et al., 2006), but the simplified graph is still complex and hard for a human to parse. As a result, the current display is of little practical use for nontrivial networks. This motivated our search for a better attack graph display.

3 Technical Approach

Attack graphs are usually displayed as node-link graphs. An excellent review of 15 different general approaches to displaying node-link graphs is available in Munzner (2006). We explored many of these using open-source or trial versions of tools and

became convinced that displaying attack graphs requires a custom approach. Hand-drawn graphs are too time-consuming. The Graphviz dot tool (Graphviz, 2007) and force-directed approaches lead to excessively complex displays unrelated to the underlying network structure. Techniques to expand and collapse parts of large graphs such as space trees (Plaisant et al., 2002) and hyperbolic trees (Lamping et al., 1995) cause global context to be lost when part of a network is expanded and, as a result, are difficult to follow. Treemaps (Johnson and Shneiderman, 1991) are excellent when summarizing data for a small set of hosts, but they do not represent a network's hierarchical structure well. Finally, multilevel cell matrices (van Ham, 2003; Noel and Jajodia, 2005) are difficult for system administrators to interpret and relate to actual networks.

3.1 Design Goals

The most important design goal for a custom display that these previous approaches lack is the ability to highlight and explain critical attack steps where attackers progress between subnets, exploit a new trust relationship, or jump into a network using a client-side attack. This capability justifies and explains the automatic recommendations made by the existing NetSPA tool and makes it more likely that these recommendations will be understood and followed by system administrators. In addition, we desired a display where all hosts that a system administrator would expect to be grouped together, such as those in a subnet or VLAN, are also positioned together. We also wanted to allow system administrators to manually position subnets to reflect an intuitive topology and to be able to perform “what-if” experiments and visually see the effect of following recommendations, installing patches, and making firewall rules more restrictive. Finally, we wanted system administrators to be able to explore reachability so they could understand which hosts are exposed through firewalls and other filtering devices. Reachability is too complex to analyze by hand, especially when firewalls include hundreds of filtering and network address translation (NAT) rules and a visual reachability display greatly simplifies a system administrator's task when trying to understand the security of large networks.

3.2 Initial System Design

Our current display 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 Shneiderman and Aris (2006). The interface also includes a set of display controls and an information pane which shows additional data about hosts and vulnerabilities. A screenshot of the

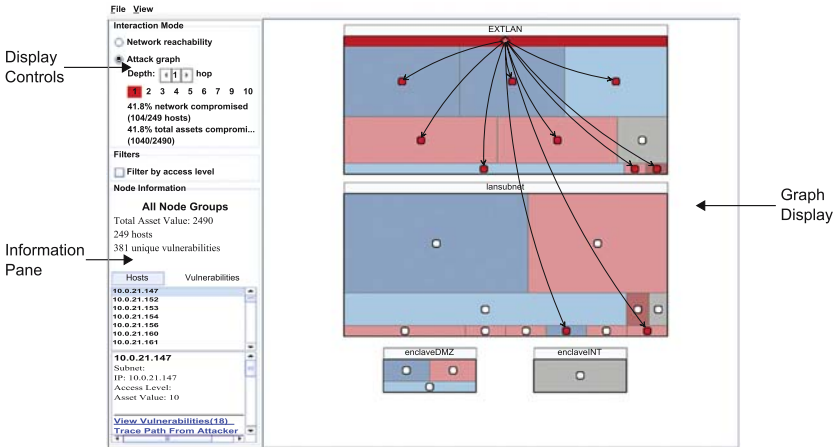


Fig. 2 Interface for attack graph display, with relevant components labeled

entire interface is shown in Fig. 2. The key components are indicated by the text labels.

Our attack graph cascade display attempts to simplify the layout of the MP graph. It shows only the collapsed state nodes from the simplified MP graph, omitting the prerequisite and vulnerability instance nodes. All state nodes attached to the same NetSPA link are grouped together into a rectangular area that typically represents all hosts in a subnet. Initially, subnet groups are successively placed across the display from top to bottom, but they can easily be repositioned and resized to form a more intuitive layout that reflects the conceptual network topology. Each subnet group is labeled by name, and its state nodes are further grouped into collapsed subgroups corresponding to the collapsed state nodes from the MP graph. The inner subgroups are colored according to the level of access of their representative states, and the relative size of each is proportional to the number of hosts it comprises. The subgroups are laid out inside of their respective subnet groups according to the space-filling strip treemap algorithm (Johnson and Shneiderman, 1991). This layout method generally produces reasonable dimensions for the rectangular areas, thus making the subgroups easily discernible.

The user interface is based upon two separate modes of interaction, which determine the meaning of the edges that are drawn between nodes. The edges in the graph display are hidden by default, and each mode enables the user to interactively show or hide a subset of edges. The first mode allows exploration of general reachability within the network. For any particular state node, there is the option to display either incoming or outgoing reachability and the corresponding edges are drawn to all nodes that can reach or be reached from the selected node. Direct interaction with the displayed nodes is supported, and each node provides a context menu containing options for hiding and revealing reachability links. The second mode offers a view of the attack graph, and the edges indicate the shortest attack paths between

sets of nodes. In this attack graph mode, the edges can be incrementally displayed in sets, where each set corresponds to one attacker hop or step. Each hop represents adding to the attack graph links to all hosts that can be compromised from currently compromised hosts. On the first hop, only vulnerable hosts that are reachable from the attacker starting location(s) can be compromised. On successive hops, links to vulnerable hosts reachable from all previously compromised hosts are added to the attack graph. A set of controls is exposed in the side panel that allows the depth of the attack graph to be incremented (and decremented) in this way. We will explore a few simple example networks in the next section to illustrate the features of this new display.

In addition to providing controls for interaction with the display, the accompanying side panel contains an information pane for viewing data that is not graphically represented. This set of information includes hosts represented by each collapsed state group, vulnerabilities present on each host, host IP addresses and asset values, and vulnerability types and descriptions. The informational display is updated whenever a particular state node is selected.

The new display was written in Java and it needed to interface with the NetSPA C++ computation engine to extract information about the network model. We developed a set of bindings between the Java and C++ code using the SWIG toolkit (SWIG, 2007). The Java visualizer is able to load the NetSPA engine as a shared library and drive it programmatically. Because performance in interfacing between Java and C++ is somewhat sluggish, we have taken pains to minimize data transfer requirements, performing analysis on the C++ side or Java side as needed.

The engine and importer required very few changes to support the new Java visualization tool. The importer carries forward some extra data about vulnerabilities which had previously been discarded, and the engine has a few additional methods to aggregate data on the C++ side, minimizing the quantity of calls necessary between C++ and Java. The visualization tool queries the NetSPA library for data on the network model, and it can also request and retrieve reachability computations, collapsed multiple-prerequisite graphs, and recommendations.

3.3 Example Network Results

The first example network shown in Fig. 3 consists of three hosts on an internal network and an external host on the other side of the firewall. The attacker starts on the external network at Host A. The firewall only allows communication between

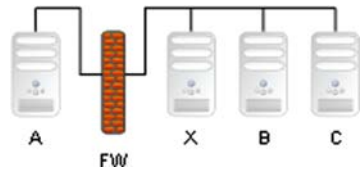


Fig. 3 Example network 1

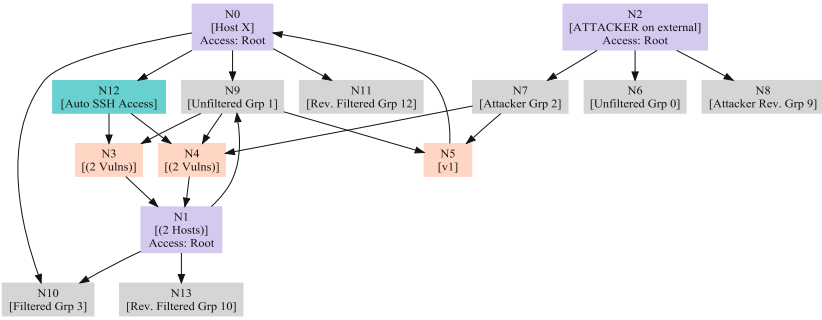


Fig. 4 Original dot display of MP attack graph for network 1

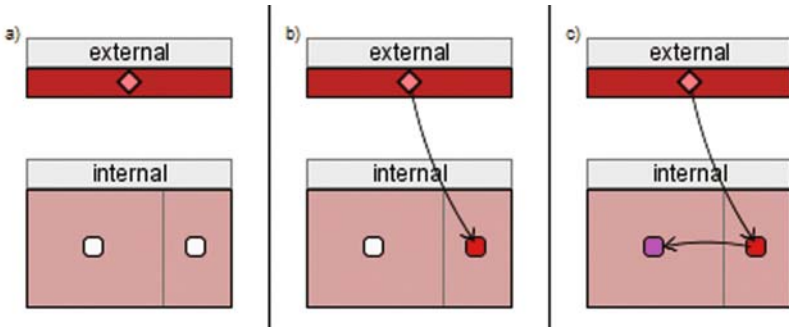


Fig. 5 Simplified display of MP attack graph for example network 1 (a) Before the attack starts, (b) after 1 attack hop or step, and (c) after 2 attack hops

Host X and the external network. Host X is an administrator machine and uses SSH to control Hosts B and C as root. The MP graph models a trust relationship between the administrator and hosts B and C that allows a user with root privileges on the administrator X to automatically log into hosts B and C as root using SSH. The administrator can also be reached via SSH from outside the firewall and there is an SSH vulnerability that allows root level compromise.

Figure 4 shows the MP graph in the original Graphviz dot format. It is relatively small and uncomplicated due to the simplicity of the network. However, the graph does not clearly convey the attacker’s possible paths and it does not at all depict the underlying network structure. The display in Fig. 5a, on the other hand, shows the two distinct subnets (internal and external) and the states contained in each. The pink diamond in the external subnet group represents the attacker’s starting location. The two white squares in the other group represent the collapsed state groups for the internal subnet – one contains both Hosts B and C, and the other contains only Host X. The subgroups are colored pink, indicating root access level for these states. Selecting a node brings up all of this information, such as access level, hosts that make up the group, and exploitable vulnerabilities from the group, in the separate

Fig. 6 Example network 2

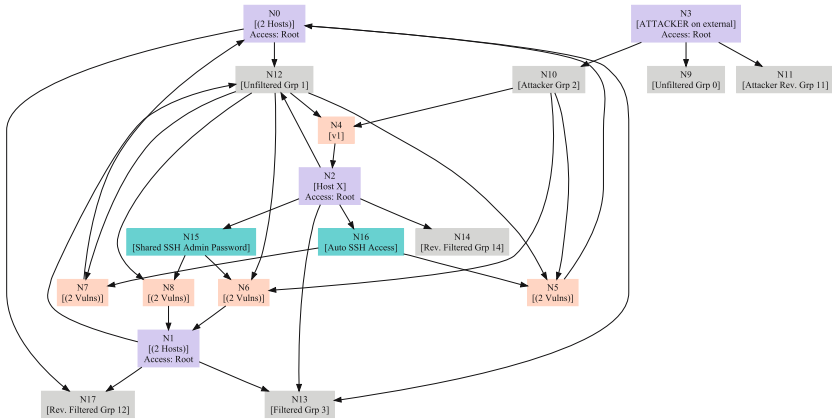
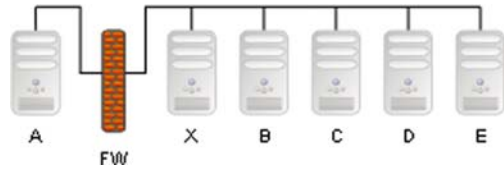


Fig. 7 Original Graphviz dot display of MP attack graph for example network 2

detail pane. Figure 5b,c show the progression of the attack graph. On the first hop, the attacker is able to go through the firewall and compromise Host X. From this point, the attacker can gain access to the remaining hosts and compromises them on the second hop.

Figure 6 presents a slightly more complicated network. It is similar to the first example network, with the addition of two workstation hosts. An administrative password is required to SSH into Hosts D and E as root. Host X possesses the necessary password. It is assumed that an attacker with root privileges on host X can capture the shared password used to administer hosts D and E and this password is modeled as another prerequisite in the MP graph. The remotely exploitable SSH vulnerability on host X has been patched, but there is a web server client-side vulnerability on host X which is the only host allowed to visit web sites in the external network through the firewall. This client-side attack replaces the SSH exploit as the first step of the attack graph in this example.

The Graphviz dot diagram for example network 2 (Fig. 7) shows that the MP graph becomes significantly more complex, even though there is a minimal addition of two hosts and one prerequisite relationship. The alternative display (Fig. 8a) remains just as simple, with only one additional collapsed state group representing root access on Hosts D and E. The layout of the internal subnet group has changed to accommodate the extra subgroup, but the relative sizes have been preserved – the upper two subgroups, each containing two hosts, are twice as big as the lower subgroup, which encloses the single Host X. As illustrated by Fig. 8b, the

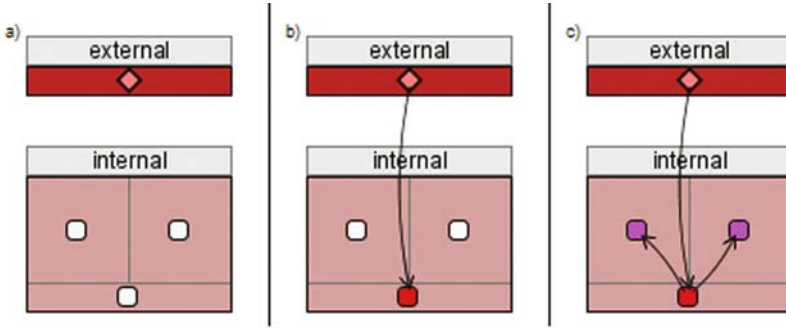


Fig. 8 Simplified display of MP attack graph for example network 2 (a) before the attack starts, (b) after 1 attack hop or step, and (c) after 2 attack hops

first step in the attack graph is identical to the previous example, and the administrator host is compromised using a client-side attack instead of an SSH server attack. Since the administrator host contains the two prerequisites required to gain access to the remaining hosts, these groups are both compromised in the next step shown in Fig. 8c. In this display, the attacker’s progression through the network is succinctly summarized by exposing the shortest, worst-case attack paths. Consequently, an immediate assessment of an attacker’s ability to penetrate a network can be made without having to deal with firewall rulesets and the complex relationships allowing reachability between hosts.

As was mentioned in the previous section, the display also provides a reachability mode. In this mode of interaction, the incoming and outgoing links for a given node indicate which groups can reach and be reached by that particular group of host states. Reachability between hosts is computed independent of vulnerabilities that are present on those hosts, thereby presenting a different view of the network. Due to the simplicity of the two example networks introduced above, the corresponding reachability displays are trivial and rather uninteresting. This functionality will be illustrated in the next section using a more substantial network graph.

3.4 Field Trial Results

To determine how well the new cascade display scales to a larger number of hosts, it was tested on a small operational network, shown in Fig. 9. The network has 250 hosts, 3,777 ports, and 8,585 vulnerability instances. The MP attack graph, presented in Fig. 10, contains 80 nodes and 190 edges. The resulting tangle of edges makes the graph virtually impossible to interpret, and renders it essentially useless. The simplified display of the graph in Fig. 11a contains only 21 collapsed state nodes partitioned into four subnet groups. As mentioned before, the color of the groups indicates the level of access: pink is for root access, dark blue for DoS, light blue for other, and gray for no access (meaning these hosts have no

Fig. 9 Operational field trial network

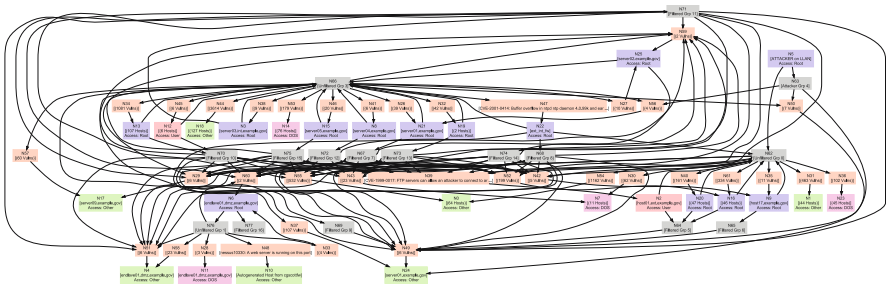
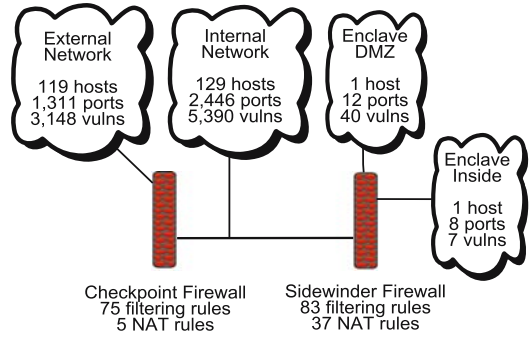


Fig. 10 Original Graphviz dot display of MP attack graph for field trial network

exploitable vulnerabilities). The red area represents the attacker’s starting location in the external network. Subnets in this display were positioned manually to represent the normal hierarchical relation used by system administrators.

The first step in the attack graph is depicted in Fig. 11b. It is immediately evident that the attacker gains access to most of the external network on the first hop, with the exception of a small group of protected hosts. More importantly, the two edges that cross over to the internal network demonstrate that the attacker can gain control of these inside hosts from the outset. Vulnerabilities that allow these transition edges are critical to the security of the internal network. Figure 11c shows the second attacker hop, where the remainder of the internal network is compromised, as well as two states on the enclave DMZ host. The third and final step in Fig. 11d reveals the complete compromise of the host in the DMZ. The enclave inside network has no incident edges and is therefore fully protected.

Figure 12 gives a sample view of general reachability within the trial network. It displays the reachability for the node in the middle right gray subgroup of the external subnet. In Fig. 12a, the incoming reachability is displayed, and the links from all of the nodes in the two larger subgroups imply that the target node in the external subnet is reachable from each of these groups. Figure 12b shows the outgoing reachability for the target node, which is able to reach each of the nodes it points to. None of these links are present in the attack graph display since the hosts represented by this node do not possess any vulnerabilities that are exploitable by the attacker.

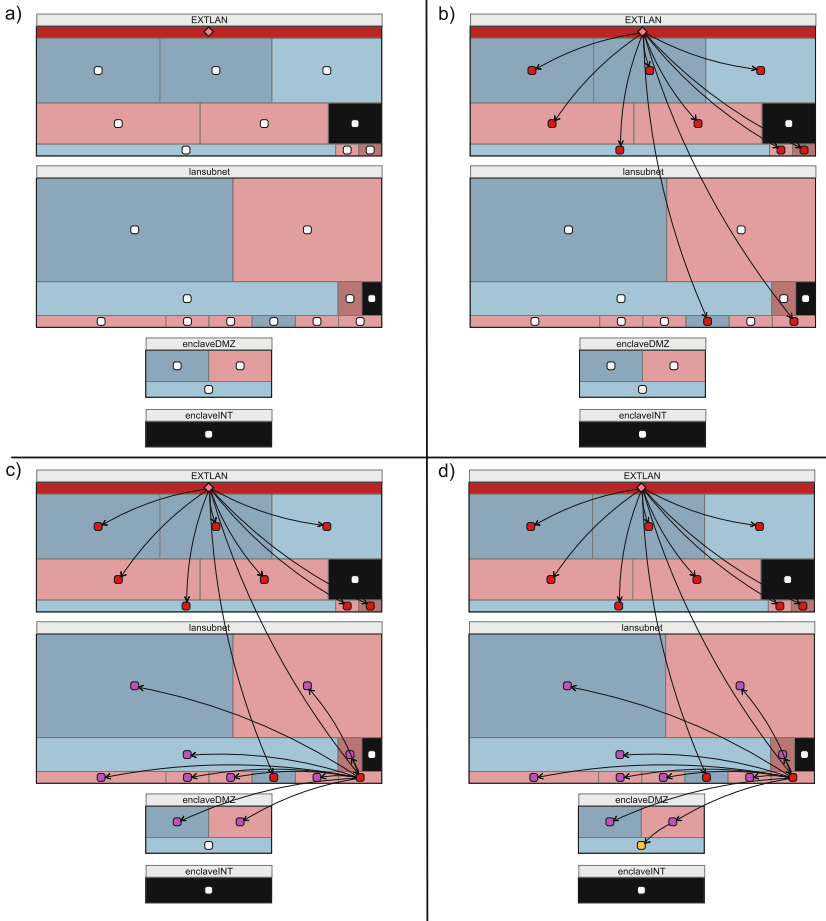


Fig. 11 Simplified display of MP attack graph for field trial network (a) before the attack starts, (b) after 1 attack hop or step, (c) after 2 hops, and (d) after 3 hops

It is also interesting to note that the set of links representing outgoing reachability is not necessarily identical to that representing incoming reachability. This example illustrates the simplicity with which complex routing and filtering rules can be graphically represented and also underscores the need for having separate views for the attack graph and general reachability.

4 Future Work

We have used this initial attack graph cascade display to illustrate reachability and attack graphs for example networks and a field trial network. Security administrators and researchers who have been exposed to this new display are able to understand

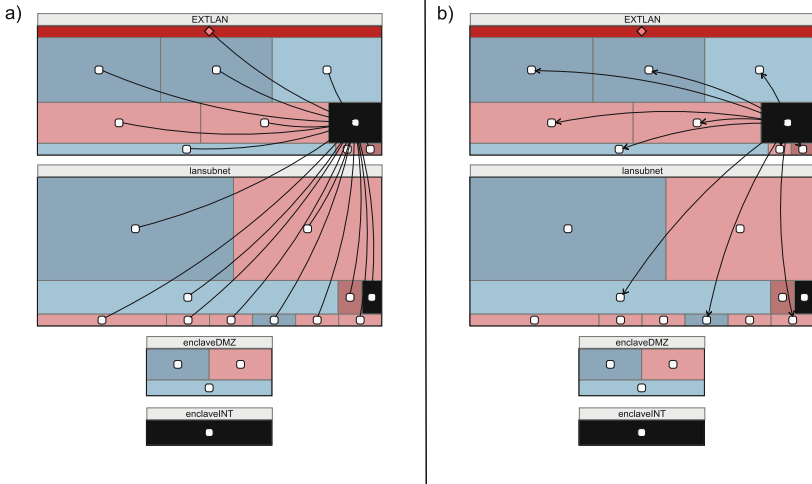


Fig. 12 Reachability display for a single node in field trial network (a) incoming reachability for one node, (b) outgoing reachability for one node

how the display relates to the analyzed network, determine which hosts can be compromised, and also identify those vulnerabilities that are most critical in enabling attackers to progress through networks. This was not true for our original Graphviz displays.

Several enhancements are currently under development for the current attack graph cascade display. In the current implementation, a starting subnet for the attacker can be selected and NetSPA automatically determines a set of worst-case IP addresses for that subnet. Support will be added for manually selecting a single starting IP address or a range of addresses. The display will also be extended to support some limited “what-if?” experiments. A list of recommendations generated by NetSPA will be presented with the option to select and apply one or more. Users can then see the effect of patching vulnerabilities and changing firewall rules when the attack graph is regenerated and the display updated to indicate which nodes are no longer accessible as a result of following a recommendation. A related feature will add support for introducing zero-day vulnerabilities into the network, allowing the user to view the changes in the regenerated graph. These will be vulnerabilities in server or client software versions that a user can propose to proactively plan for future attacks. In addition to the original MP attack graph, NetSPA also generates summary attack graph metrics that will be provided as part of the interactive display. These include calculation of the total assets captured vs. the number of attack hops, the number of unique exploits required by an attacker, and an estimate of the cost of development for the exploits required. Plots of assets captured vs. these measures of attacker effort will supplement the information provided in the attack graph display by presenting a different view of the data. There will also be support for importing historical data about a network and computing metrics representing a past state of

the network. This older data could be presented alongside or overlaid on the current plots, showing how the security of the network has changed over time.

Another goal in this development effort is to ensure that the display application is intuitive and user-friendly. To measure the usability of the graphical interface, user tests will be conducted with system administrators and blue team members. These assessments will aid in identifying additional features that may need to be included or existing functionality that may need to be modified. The resulting feedback will enable us to develop a simpler and more efficient interface for the display. We also plan to apply the cascade display to very large networks with many subnets and determine if some type of user-generated hierarchy or other approach is necessary to display such networks in a simple and uncluttered way.

5 Conclusions

We have developed an efficient attack graph cascade display that is intuitive, produces a compact representation, highlights critical attack steps that lead into new network areas, and displays both attack graph and reachability information. Because the display is easy to understand and interact with, a system administrator can confirm the accuracy of the analysis and the validity of the recommendations. This may encourage the practical use and acceptance of attack graphs as tools to improve network security.

This new display improves over past research and commercial attack graph displays in a few critical areas. First, hosts in subnets are presented in close proximity using a treemap. This makes it easy to associate hosts in the graph with hosts in actual subnets. Hosts in each treemap are automatically grouped based on level of compromise, how the hosts are treated by firewalls, trust relationships the hosts participate in, and prerequisites required to compromise hosts. These groupings provide visual indications of the network security and greatly simplify the display. The color of each group indicates the level of compromise and the size of the treemap rectangle for a group indicates the number of hosts in that group. At least one commercial company (RedSeal, 2007) uses a treemap, but it appears to be used to provide an overall indication of the number of vulnerabilities for all hosts in a network and not as part of an attack graph display. Other approaches to generating attack graphs collapse hosts into groups (e.g. Swiler et al., 2001; Noel and Jajodia, 2004), but our approach automatically forms collapsed groups, results in large computational savings, and leads to groupings and layouts that can be readily understood. A second improvement is that users position subnet treemaps themselves to represent an intuitive notion of the network hierarchy. This requires some extra user effort and may not scale when there are hundreds of subnets, but it results in understandable topologies. A third improvement is that we display all hosts that can be compromised from a specified attacker starting point. Some other displays (e.g. Skybox, 2007) require users to specify both an attacker starting point and a target destination and only show the shortest path between these two points. The cascade display provides an

overall picture of the attack graph and shows how patching specific critical vulnerabilities can protect not just one target, but many. A fourth improvement is that an interface between the Java cascade display software and the NetSPA C++ engine allows rapid interaction with and regeneration of the attack graph display. Computation times for computing attack graphs with 1,000's of hosts are typically less than a few seconds (Ingols et al., 2006) and this allows interactive display and analysis. Computation times for computing recommendations and showing the result of recommendations are similar. Finally, we display reachability information as well as attack graphs by using the same network display and operating it in two separate modes, where edges have a different meaning in each mode.

Acknowledgements We would like to thank Seth Webster and Rob Cunningham for interaction and feedback on initial display designs.

References

- Gansner ER, Koutsofios E, North SC, Vo KP (1993) A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, vol. 19, no. 3, 214–230
- Graphviz (2007) Graph visualization software. <http://www.graphviz.org>. Accessed 6 Sept 2007
- Ingols K, Lippmann R, Piwowarski K (2006) Practical attack graph generation for network defense. *Proceedings Computer Security Applications Conference (ACSAC)*, 121–130
- Johnson B, Shneiderman B (1991) Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the Second Conference on Visualization '91*, GM Nielson and L Rosenblum, Eds. *IEEE Visualization*, IEEE Computer Society Press, Los Alamitos, CA, 284–291
- Lamping J, Rao R, Pirolli P (1995) A focus + context technique based on hyperbolic geometry for visualizing large hierarchies. *Proceedings of ACM CHI '95 Conference*, 401–408
- Lippmann R, Ingols K (2005a) An annotated review of past papers on attack graphs. MIT Lincoln Laboratory, Lexington, MA, Technical Report, 2005, ESC-TR-2005-054
- Lippmann R, Ingols K (2005b) Evaluating and strengthening enterprise network security using attack graphs PR-IA-2. MIT Lincoln Laboratory Project Report, 12 August 2005
- Lippmann R, Kukulich L, Singer E (1993) LNKnet: neural network, machine-learning, and statistical software for pattern classification. *Lincoln Laboratory Journal*, vol. 6, no. 2, 249–268
- Lippmann R, Ingols K, Scott C, Piwowarski K, Kratkiewicz K, Cunningham R (2006) Validating and restoring defense in depth using attack graphs, MILCOM 2006, Washington, DC
- Munzner T (2006) 15 Views of a node-link graph: an infovis portfolio. <http://www.cs.ubc.ca/~tmm/talks.html>. Accessed 6 Sept 2007
- Noel S, Jajodia S (2004) Managing attack graph complexity through visual hierarchical aggregation. *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*. New York, NY, USA: ACM Press, New York, 109–118
- Noel S, Jajodia S (2005) Understanding complex network attack graphs through clustered adjacency matrices. *Proceedings Computer Security Applications Conference (ACSAC)*, 160–169
- NVD (2007) National Vulnerability Database <http://nvd.nist.gov/>. Accessed 6 Sept 2007
- Ou X, Govindavajhala S, Appel AW (2005) Mulval: a logic-based network security analyzer. In *Proceedings of the 14th Usenix Security Symposium*, 2005, 113–128

- Plaisant C, Grosjean J, Bederson B (2002) Spacetree: supporting exploration in large node link tree, design evolution and empirical evaluation. In INFOVIS 2002, IEEE Symposium on Information Visualization, 57–64
- RedSeal Systems Inc. (2007) <http://www.redseal.net/>. Accessed 6 Sept 2007
- Shneiderman B, Aris A (2006) Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, 733–740
- Skybox Security Inc. (2007) <http://www.skyboxsecurity.com>. Accessed 6 Sept 2007
- SWG (2007) <http://www.swig.org>. Accessed 6 Sept 2007
- Swiler LP, Phillips C, Ellis D, Chakerian S (2001) Computer attack graph generation tool, Proceedings of the Second DARPA Information Survivability Conference & Exposition (DISCEX II), Los Alamitos, CA. *IEEE Computer Society*, vol. 2, 307–321
- van Ham F (2003) Using multilevel call matrices in large software projects. *Proceedings Information Visualization (INFOVIZ)*, 227–232

Intelligent Classification and Visualization of Network Scans

C. Muelder, L. Chen, R. Thomason, K.-L. Ma, and T. Bartoletti

Abstract Network scans are a common first step in a network intrusion attempt. In order to gain information about a potential network intrusion, it is beneficial to analyze these network scans. Statistical methods such as wavelet scalogram analysis have been used along with visualization techniques in previous methods. However, applying these statistical methods causes a substantial amount of data loss. This paper presents a study of using associative memory learning techniques to directly compare network scans in order to create a classification which can be used by itself or in conjunction with existing visualization techniques to better characterize the sources of these scans. This produces an integrated system of visual and intelligent analysis which is applicable to real world data.

1 Introduction

This paper presents an intelligent approach to visual characterization of network scans. This characterization process is a useful tool for analysts in counterintelligence efforts against potential network intruders. Scanning a network is a common first step in a network intrusion attempt. The process of scanning a network is usually performed to determine what exists on a network. For example, if an attacker is looking for exploitable web servers, then he or she would attempt to connect on TCP/UDP port 80 to every possible IP address within a particular range. If there is a web server using port 80 at any of these IP addresses, it will probably respond. However, for addresses where there is nothing, or where there is a computer that is not running a web server, there will be no response. Detecting these scans is fairly

C. Muelder, L. Chen, R. Thomason, and K.-L. Ma
University of California, Davis, e-mail: muelder@cs.ucdavis.edu, leich@ucdavis.edu, rdthomason@ucdavis.edu, ma@cs.ucdavis.edu

T. Bartoletti
Lawrence Livermore National Laboratory, e-mail: azb@llnl.gov

easy (McPherson et al., 2004; Simon et al., 2005), but mining them for information about the attacker can be relatively difficult.

An attacker can do several things in an attempt to make such a scan anonymous; for instance, coming from different source addresses or scanning destination addresses in a random order. In fact, it is even possible to perform a scan indirectly by using a fake source address so the scan looks like it is coming from a different computer. Denial of service and worm propagation attacks can also produce scan-like behavior, and since they do not need the target to respond, they often fake their source addresses as well. The port number is also not sufficient for categorizing scans, because both malicious and benign scans can often be run on the same port number. For example, both a web crawler looking for new sites and a worm that targets webservers would target port 80. Therefore, some other metric must be used for categorization purposes.

Experimental results have shown that variations in arrival time of the scanning connections often have a high correlation with particular sources (Parno and Bartoletti, 2004). That is, the timing information produces a digital “fingerprint” that correlates to a particular source. While in theory an attacker could try to further obfuscate a scan through controlled delays, in practice this is quite rare. In fact, doing so would require a customized tool which would make the timing signature even more unique. It is surmised that this correlation between timing patterns and sources is due to a combination of factors, including the connection application software employed, the supporting hardware platform, operating system characteristics, and regular interference from other processes on the source system that compete for these resources. Network factors such as number of hops and properties of the routers are certainly responsible for some degree of the timing structure as well, and make this a particularly interesting and challenging problem in network traffic forensics.

The critical question analysts seek to answer is whether the same source ensemble run in an entirely different network (hence, different source IP address), would exhibit a timing structure that is sufficiently similar. This would uniquely identify the environment, and, by extension, the actor behind the observed activity. Alternatively, the analysts would like to know the degree to which the effects of intervening routers produce characteristic packet timing irregularities for different activities conducted from a constant network location. This latter capability would provide a means to determine the veracity of a given source IP address when faced with potential address-spoofing.

To answer this question, analysts must first be able to correlate scans from different source addresses based on timing information. To accomplish this, the analysts need to compare the timing information of very large quantities of network scans quickly and efficiently. Previous methods have used statistical reduction and visualization to compare these scans. However, the statistical reduction has an inherent data loss and can be susceptible to noise, and the direct visualization techniques can become quite unwieldy as the number of scans increases. Intelligent pattern recognition algorithms are quite good at dealing with these issues. They are good at reconstructing distorted or incomplete data, and they scale well to large numbers of inputs.

So, an artificial intelligence based classification methodology was developed that can be used both alone and in conjunction with existing visualization techniques to better characterize potentially hostile scan sources. Since the raw scan data is not directly conducive to comparison, we first transform and encode it into a format that can be compared. We then use a machine learning algorithm to classify the scans. Finally, we combine these results with statistical and visual techniques, and demonstrate how the complement each other.

2 Related Work

The study of network security has been popular for the last decade. Visualization systems have been developed to visualize and compare the network scan pattern in order to detect the potential for attacks. ScanVis (Muelder et al., 2005) presents a means of facilitating the process of characterization by using visual and statistical techniques to analyze the patterns found in the timing of network scans. The system allows large numbers of network scans to be rapidly compared and subsequently identified. Conti and Abdullah (2004) use a parallel coordinates system to display scan details and characterize attacks.

There are many systems designed for detection of scans. There exist visualization based tools such as PortVis (McPherson et al., 2004), as well as data mining methods such as those presented in Simon et al. (2005), but all these approaches focus on the detection of suspicious activity and not on the analysis of such activity. The work presented here does not focus on the detection of these scans. Rather, it focuses on what can be learned once these scans have already been detected.

Machine learning methods – associative memory models in particular – have been widely applied in the pattern recognition and classification area. Tavan et al. (1990) extend the neural concepts of topological feature maps towards self-organization of auto-associative memory and hierarchical pattern classification in 1990. Stafylopatis and Likas (1992) proposed a technique based on the use of a neural network model for performing information retrieval in a pictorial information system. The neural network provides auto-associative memory operation and allows the retrieval or stored symbolic images using erroneous or incomplete information as input.

Machine learning algorithms have also been directly applied to computer security problems in the past. Girardin and Brodbeck (1998) uses a machine learning technique to analyze log files and look for anomalies. These techniques have also been applied to intrusion detection systems in several approaches (Komlodi et al., 2004; Portnoy et al., 2001; Sinclair et al., 1999). These approaches focus on using the machine learning to aid in the detection of malicious activity, rather than the analysis of it. Other approaches use intelligent classification to analyze individual activities to detect patterns in data (Axelsson, 2004a,b). Unlike all these works, the work presented here does not focus on detection. Rather, it aims to analyze potentially malicious activity to gain information about the source.

3 Technical Approach

The methods used in this paper cover a wide variety of techniques, including statistical analysis, visualization, and artificial intelligence, with the ultimate goal of aiding in counterintelligence efforts by characterizing potential adversaries through their scanning activity. An overview of these techniques is shown in Fig. 1. In order to analyze and characterize network scans, they must first be detected and extracted from the raw network data. This is currently being done through the use of some fairly simple statistical data mining techniques. These techniques consist of statistically detecting a scan in progress, then tracking it backwards and forwards in time in order to find when the scan starts and stops. Most of the scans that are detected and extracted in this manner are caused by unknown sources, and so they form a set of unknown scans. But some of the extracted scans originated from known sources, where the parameters of the scans were carefully controlled. That is, properties such as the source hardware, software, scanning tool, and location on the internet are known. This creates a set of controlled scans, which can then be used to compare against the unknown scans.

Due to the constant scanning activity occurring all the time on the internet, the set of unknown scans can rapidly grow to be quite large, so it is infeasible to compare them against each other by hand. The visualization techniques of Muelder et al. (2005) are one set of methods being used to deal with the scalability issues through statistical analysis and visual presentation of both the controlled and unknown scans. These scans can also be used as inputs to an artificial intelligence algorithm, which is the focus of the approach presented here. The controlled scans are used as training data for an associative memory learning process which generates a weight matrix. This weight matrix can then be used in an associative memory reconstruction process to take the unknown scans and classify them. These classified scans can then either be used directly to characterize their sources, or they can be used in the visualization system to improve its effectiveness at scanner characterization.

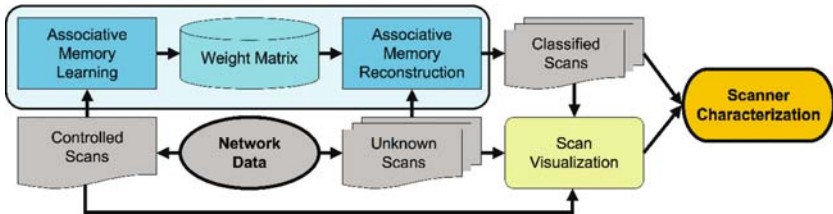


Fig. 1 *The overall methodology.* Known and unknown scans are collected off the network. The known scans are used to train the associative memory in order to create a weight matrix which can then be used to classify the unknown network scans. All of these scans can be visualized, with the overall goal of gaining information about the sources of the scans. The three blue items selected in the upper left are the focus of this paper

3.1 Scan Data and Representation

The scan data used in this paper was generated and collected by the Computer Incident Advisory Capability (CIAC) Group from the network at Lawrence Livermore National Lab (LLNL). The visualization techniques and data formats used are introduced and described in more detail in Muelder et al. (2005). The control data was generated by performing multiple scans, using known tools such as nmap (nmap, 2007) under various settings, on an isolated LAN in order to minimize outside interference. The unknown scans were collected from real network scans targeted at LLNL.

3.1.1 Network Scan Data

Each scan consists of timing information of a scan over a class B network, which contains 65,536 destination addresses. The scan data in its most raw form consists of pairs of destination addresses and times, one for each probe in the scan. Comparing scans in this representation against each other is very ambiguous, since there is no inherent order to this data, so some sort of transformation is necessary. One could choose to order the data according to time or according to address space, and each ordering would yield different results. Alternatively, instead of using the original values, it can often be useful to consider the deltas between subsequent values, in order to emphasize fine details. Various transformations have been performed to create a set of ordered data modes to visualize different aspects of the data (Muelder et al., 2005). There are a plethora of different possible transformations, and several of the modes that were investigated were found to be quite useful in analyzing the scans.

For this research, we chose to use the data mode referred to in our previous work as mode 22 (Muelder et al., 2005). In this data mode each value is derived as the difference between the actual and expected arrival times for the first probe to each destination address, where the expected arrival times are for an ideal scan that starts with the first address, probes successive addresses at a constant rate, and ends with the last address. Because of the fixed size of this transformation, individual scans can be visually represented in detail in a 256×256 grid, where the x and y axes represent the third and fourth bytes of the address, respectively, and the color of each pixel represents the derived value at that coordinate. An example of a scan in presented with this visualization is shown in Fig. 2. As described in the original paper (Muelder et al., 2005), data mode 22 is of fixed size (65,536 values) and is effective at capturing fine details in the scans in the form of intricate patterns. Because of the uniqueness of the patterns under this transformation, data mode 22 is optimal for a pattern matching technique such as is presented here. The techniques described here could be applied to other data modes as well, but that is beyond the scope of this work.

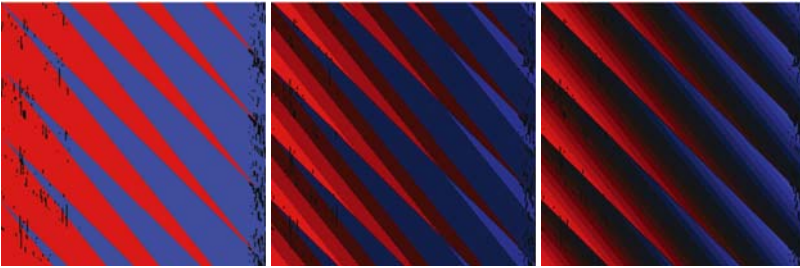


Fig. 2 *Sample network scan using 2, 3, and 4 bits.* The scan is represented by a plot of the third and fourth bytes of the destination IP addresses. Color represents how much earlier or later than expected a probe arrived at that destination. Different bit encodings create different granularities of colors in the scan representation

3.1.2 Binary/Bipolar Coding

The intelligent algorithm we use works on bipolar patterns, which means that every value in the data series is either 1 or -1 . Binary patterns can be converted to bipolar patterns easily by replacing every 0 in the pattern with a -1 . In order to use non-binary data, such as the real valued mode 22 output, the patterns must first be transformed into binary patterns. The number of bits used can be arbitrary: using higher numbers of bits allows for finer grained distinction between patterns than low numbers of bits, but uses more memory and requires more processing time. Whatever bit length is being used, 0 is reserved for dropped or missed packets, and those values are colored black in the graphs. The highest value, $2^n - 1$ is also reserved and not used. This leaves an equal number of values to represent early packets which are colored red, and late packets which are colored blue. In the simplest coding used in this paper, two bits are used for each IP address in the scan. If the captured time of the first probe is earlier than the expected time, we give the neuron value “01” and if it is later than the expected time we give the value “10”. More bits can also be used to break down the continuous range of possible values into a finer set of discrete ranges. Figure 2 shows a particular scan where the data values are encoded using 2, 3, and 4 bits.

3.2 An Intelligent Method

There are two main categories of machine learning algorithms: unsupervised and supervised. An unsupervised algorithm approaches a data set with little to no pre-conceived notions about the classes of items that it contains; it attempts to separate the items into statistically distinct groups on its own. Examples of unsupervised algorithms include k-means clustering and self-organizing maps (MacQueen, 1967; Kohonen, 1989). But, unsupervised algorithms cannot effectively use controlled

data. Since there is controlled scan data available, a supervised algorithm is more readily applicable to the task presented here. A supervised algorithm is given a large number of examples of items in each class it is to detect; each is labeled with the class to which it belongs. Then, when new items are presented, it puts them into the class they are most likely to belong to. Examples of supervised algorithms include neural networks (McClelland et al., 1986; Werbos, 1974), support vector machines (Cortes and Vapnik, 1995; Cristianini and Shawe-Taylor, 2000), and associative memory (Kohonen, 1978). Of these, associative memory in particular was designed for pattern recognition and has been shown to be faster and more effective than most other approaches for this task (Kohonen, 1978). Since the problem presented here consists of matching large numbers of patterns, associative memory is an ideal choice, and so was selected for our approach.

3.2.1 Associative Memory

The concept of associative memory was first proposed by Kohonen (1978). In the human mind, the memory process takes a stimulus and matches it up with what it remembers about that stimulus. The nature of associative memory is an emulation of human memory, and so it works in a similar way. Associative memory takes a stimulus in an input layer of artificial neurons, feeds them through one or more layers of intermediate artificial neurons, then produces an associated result in an output layer of neurons. The associative memory model has many applications, for example, pattern cognition and reconstruction, image processing, face, character and voice recognition, databases, control systems, and robotics. Human memory is quite robust in that it can correct errors and recognize stimuli even when they are incomplete or distorted. Similarly, in associative memory, given a stimulus pattern that is distorted or incomplete, associative memories are often able to reproduce the correct response pattern. It does this by “remembering” a set of known patterns and then matching up incoming unknown patterns against them. There are a variety of types of associative memory models that have been studied in the last two decades, including Hopfield networks (Hopfield, 1982) and bidirectional associative memory (BAM) (Kosko, 1988).

3.2.2 The BAM Algorithm

Bidirectional associative memory was introduced by Kosko (1988). Like all associative memory algorithms, BAM maps patterns from an input layer X to patterns in an output layer Y , where each layer consists of a set of artificial neurons capable of representing the input or output patterns. BAM has the distinction that it is bidirectional, so it can also map patterns from layer Y back to layer X . The patterns in both layers are represented as artificial neurons which can be either 1 or -1 indicating whether they are firing or not. That is, each element $x_{k,i}$ in each pattern

\mathbf{x}_k in the X layer and each element $y_{k,j}$ in each pattern \mathbf{y}_k in the Y layer is either 1 or -1 . While there does exist a continuous version of BAM, it introduces fuzzy logic and non-linearity to the system, so we use the more common discrete BAM here.

BAM is a heteroassociative memory algorithm, which means that the X layer and Y layer of the network have distinct dimensions. This makes it useful for our application since our X layer is very large, while our Y layer can be much smaller. In this work the X layer patterns are the network scans in a bipolar encoding which are of size $m = 65,536 * b$ (where b is the number of neurons used to encode each value) and the Y layer patterns are IDs which are of a size n sufficient to store the number of control patterns we are interested in. While there could be 2^n potential IDs in layer Y, this would not be reliably recalled by BAM (Kosko, 1988). In our system we use 32 neurons in the Y layer by default, but allow the user to set this.

Several methods of generating IDs were tried. The first method was to make each ID a bipolar encoding of the binary representation of an incremental number: $\mathbf{y}_1 = (-1, -1, \dots, -1, -1, 1)$, $\mathbf{y}_2 = (-1, -1, \dots, -1, 1, -1)$, $\mathbf{y}_3 = (-1, -1, \dots, -1, 1, 1)$, etc. The next method was to make each ID a bipolar encoding of the binary representation of powers of two: $\mathbf{y}_1 = (-1, -1, \dots, -1, 1)$, $\mathbf{y}_2 = (-1, -1, \dots, -1, 1, -1)$, $\mathbf{y}_3 = (-1, -1, \dots, 1, -1, -1)$, etc. However, the best results were when the IDs were generated pseudorandomly, seeded with the index of the control pattern. The first two methods only use a small portion of the weight matrix because they only use $\log(n)$ and n bits, respectively, where n is the number of control scans. The advantage of the third method is that it utilizes the entire range of bits in the Y layer regardless of the number of control scans. This is useful because BAM can get “confused” if scans that are very different have IDs that are very similar (as measured by the Hamming distance). As long as the number of bits used for the ID is large enough, the potential for collisions, or the Hamming distance being too small, can be avoided.

The BAM network itself consists of an $m \times n$ matrix of weights going between each of the m neurons of the X layer and the n neurons of the Y layer. This matrix is derived from the controlled pairs of patterns in the X and Y layers, and then used later to map patterns from the X layer to the Y layer and vice versa. The weight matrix \mathbf{W} is calculated by

$$W_{i,j} = \sum_{k=1}^K x_{k,i} * y_{k,j}, \quad (1)$$

where $x_{i,k}$ is the value of the i th neuron in layer X and $y_{k,j}$ is the value of the j th neuron in layer Y of the k th pair of controlled patterns. Calculating this matrix trains the network, which can then be used to classify unknown scans. Since calculating each entry of this matrix is fairly simple, generating the entire matrix can be done fairly quickly.

Classification is performed by starting with a pattern in one layer, then mapping it between the X and Y layers using the matrix until it converges to a constant pair of patterns. Each mapping from the X layer to the Y layer is calculated by

$$y_{k+1,j} = \begin{cases} 1 & \text{if } \sum_{i=1}^m W_{i,j} * x_{k,i} > 0 \\ y_{k,j} & \text{if } \sum_{i=1}^m W_{i,j} * x_{k,i} = 0 \\ -1 & \text{if } \sum_{i=1}^m W_{i,j} * x_{k,i} < 0 \end{cases} \quad (2)$$

for $0 < j \leq n$, where $\mathbf{x}_k = \{x_{k,i} | 0 < i \leq m\}$ is the current pattern in the X layer. And each calculation from the Y layer to the X layer is calculated as

$$x_{k+1,i} = \begin{cases} 1 & \text{if } \sum_{j=1}^n W_{i,j} * y_{k,j} > 0 \\ x_{k,i} & \text{if } \sum_{j=1}^n W_{i,j} * y_{k,j} = 0 \\ -1 & \text{if } \sum_{j=1}^n W_{i,j} * y_{k,j} < 0 \end{cases} \quad (3)$$

for $0 < i \leq m$, where $\mathbf{y}_k = \{y_{k,j} | 0 < j \leq n\}$ is the current pattern in the Y layer. Eventually this system will converge (as shown by Kosko (1988)), and the resulting patterns are output. In our case, the input is one of the scans into the X layer, and once it converges, the output is the resulting ID in the Y layer. If the output exactly matches the ID of one of the control patterns a match has been successfully made, otherwise it remains unclassified.

3.2.3 BAM Results

Application of the BAM algorithm to actual network scans yields valuable results. As depicted in Fig. 3, we found that the BAM algorithm, when trained on a single example of multiple kinds of patterns, effectively categorizes the unknown scans. Even when patterns are somewhat different due to varying amounts of noise or distortion, they are still associated with the correct training pattern. This shows that the techniques presented here are able to effectively classify many network scans according to a set of training scans. Thus, given a good set of training scans, this technique can classify large numbers of unknown scans.

But classification in this manner is limited by the control data that is available. For example, this technique could not be used to detect and classify a new worm, since no control data would be available for that particular worm. Even creating control data for known malicious attacks such as worms can be difficult due to policy issues regarding launching attacks against one's own network. However, the techniques presented here would easily extend to any kind of scan-like data provided one has access to or can generate such control data. Additionally, if there are a few particular unknown scans that warrant further investigation, the BAM algorithm could

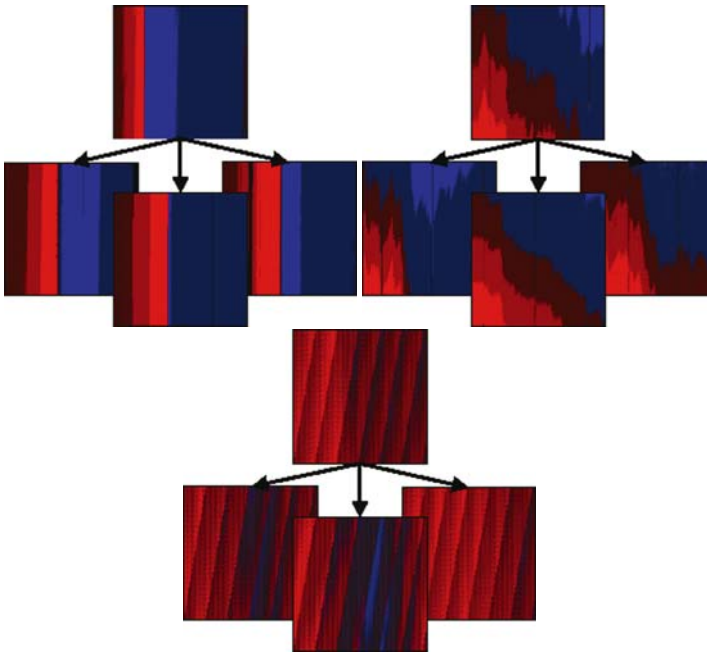


Fig. 3 *Classification examples.* The system matches numerous scans to their corresponding training scans. Three training patterns using 3 bits for encoding are shown. Below each are several example classifications that were made

be trained on those select unknown scans to try and determine if similar scanning behaviors have happened in the past.

3.3 Visualization Integration

Once scan patterns have been classified by matching them up with their associated control patterns, it is useful to present these results visually to the user. This is particularly important as the number of classified scans increases and becomes too unwieldy to analyze by hand. For this work, it was decided that these results could best be visualized as a coloring of an existing graph generated by statistical means as in ScanVis (Muelder et al., 2005). So, the scans were run through a wavelet analysis to generate scalograms, and these scalograms were compared against each other to generate a graph representation.

The combination of the intelligent and statistical techniques is effective because the wavelet techniques and the associative memory techniques measure different properties of the data. For instance, the wavelet analysis can detect a particular feature within a pattern, but would not carry any information about where in the scan this feature occurs. The associative memory, on the other hand, would reliably

match scans based on where a feature occurs in a scan, but would also match a slightly different feature that occurs in the same area. Thus, the two approaches can complement each other well when used together.

3.3.1 Wavelet Analysis

While transforming the scans into a regular form makes them comparable, a direct comparison would still not yield useful results. For instance, if an attacker scanned every other network address one day, then came back the next day and scanned the addresses that were skipped the first time, then a direct comparison would reveal no similarity, even though the patterns would be nearly identical; they would just be out of phase from each other. However, there are several algorithms utilizing frequency analysis that are useful for handling this kind of data, such as Fourier transforms and wavelet analysis. Although network scan patterns can exhibit periodic or quasi-periodic structure, they often contain gaps, aperiodic aberrations, and regions where the relative phase of the periodic structures has shifted. These are things that Fourier analysis has been found to handle poorly (Graps, 1995).

So, wavelets are used because they are relatively resistant to both phase shifts and noise. This means that similar patterns will have similar wavelet scalograms, even if the patterns are shifted slightly or different parts of the pattern are missing, as can be seen in Fig. 4a. But dissimilar patterns will still produce different scalograms, as can be seen in Fig. 4b. There are several variations on the wavelets that can be used, several of which are enumerated in (Muelder et al., 2005). For this work we chose to use the following wavelet. Given a series of $N = 2^n$ items $D_0 = (d_{0,1}, d_{0,2}, \dots, d_{0,N})$, we calculate recursively:

- $D_k = (d_{k,1}, d_{k,2}, \dots, d_{k,2^{n-k}})$,
- $S_k = (s_{k,1}, s_{k,2}, \dots, s_{k,2^{n-k}})$,
- $\sigma_k = \sum \frac{S_k}{2^{n-k}}$,

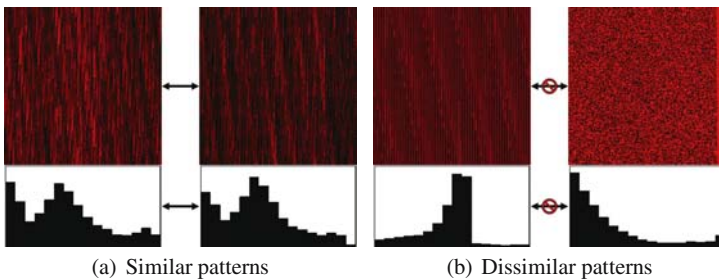


Fig. 4 Wavelet scalograms. Applying wavelet scalograms creates a representation that is directly comparable, at the cost of data loss (Muelder et al., 2005)

for $0 < k < n$, where

- $d_{k,i} = \frac{d_{k-1,i} + d_{k-1,i+1}}{2}$,
- $s_{k,i} = \begin{cases} -1 & \text{if } d_{k-1,i} > d_{k-1,i+1} \\ 1 & \text{if } d_{k-1,i} < d_{k-1,i+1} \\ 0 & \text{if } d_{k-1,i} = d_{k-1,i+1} \end{cases}$.

At each recursion the σ values are the mean of the corresponding data series, and they estimate the variance at each resolution. Once these wavelet scalograms are calculated, they can be directly compared to one another. The downside to using these wavelets is that they lose fine details in the data, such as where a particular feature occurs in the pattern. Still, they can be used to create a good approximate overview from which other information can be shown.

3.3.2 Graph Representation

In order to represent large numbers of these scans at once, a graph representation of the scans was used, just as in Muelder et al. (2005). In this view, each scan is a node in a complete graph, and each edge is calculated as the inverse Euclidean distance squared between the wavelet scalograms of the two scans it connects. That is, for an edge between scans \mathbf{A} and \mathbf{B} , the weight $W(\mathbf{A}, \mathbf{B})$ is:

$$W(\mathbf{A}, \mathbf{B}) = \frac{1}{1 + \|\mathbf{A} - \mathbf{B}\|^2}.$$

A force-directed layout, LinLog (Noack, 2004), is then applied to this graph, which groups similar scans together according to the edge weights between them. Edges that are below a threshold are dropped for clarity. This creates an overview in which large scale trends in the wavelet analysis, such as clusters, can be seen.

We can also use it to display the results of applying the BAM algorithm by coloring the nodes according to their classification. An example of this is shown in Fig. 5, where the classification process was run on a small set of controlled scans. In this example, the set of scans was classified according to three training scans, of which several samples are shown. In this image it can be seen that scans that have the same color usually have very similar patterns. It can also be seen that some scans did not match any of the controlled patterns, which indicate anomalous patterns which could be looked at more closely.

However, of even more interest is that the BAM classification and the clusters according to wavelet analysis do not have a 1-to-1 correlation. There are more than one cluster that are colored the same according to BAM, indicating a difference between these scans that wavelet analysis picked up that BAM did not. And there are clusters that contain a scan or two of a different color than the rest of the scan, indicating scans that contain a difference that BAM detected that wavelet analysis missed. Thus, the two techniques complement each other well in this manner.

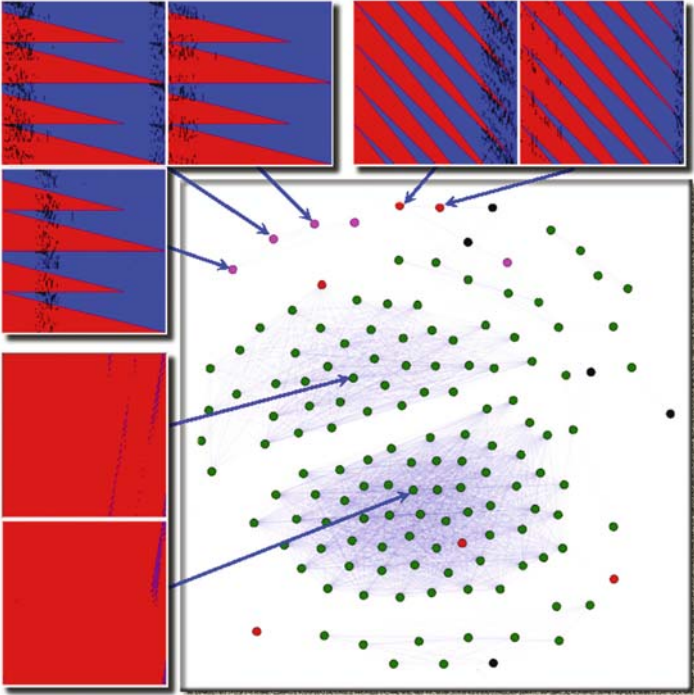


Fig. 5 A graph representation. A graph of a set of controlled scans is generated and laid out according to the wavelet scalogram analysis, as in ScanVis (Muelder et al., 2005). BAM was used to classify these control scans according to three selected scans. The results of using BAM is represented by coloring the nodes of this graph. Examples of scans from each classification are shown around the outside. Outliers which were not classified by BAM are shown as black nodes

3.4 A Case Study

Figure 6 shows the results of applying the techniques presented here to color a graph of over 800 unknown scans collected off the network. One pattern that is readily apparent is the large number of unclassified scans, which are colored black in the image. This is indicative of the large number of different kinds of scans active on the internet that have not yet been identified. However, by applying an associative memory approach, these scans have been highlighted by a process of elimination, indicating a set of scans that would be beneficial to look at in more detail. Using wavelets alone would not have identified these scans.

Another prevalent pattern is that the nodes that were clustered by the statistical analysis are generally classified the same according to the associative memory. What is of interest is that some of the scans within a cluster are classified differently. This is indicating that the machine learning algorithm is picking up on a detailed pattern that is lost by this statistical analysis. As is shown in Fig. 7, the difference in the patterns is not necessarily even difficult to discern by eye. This shows that machine

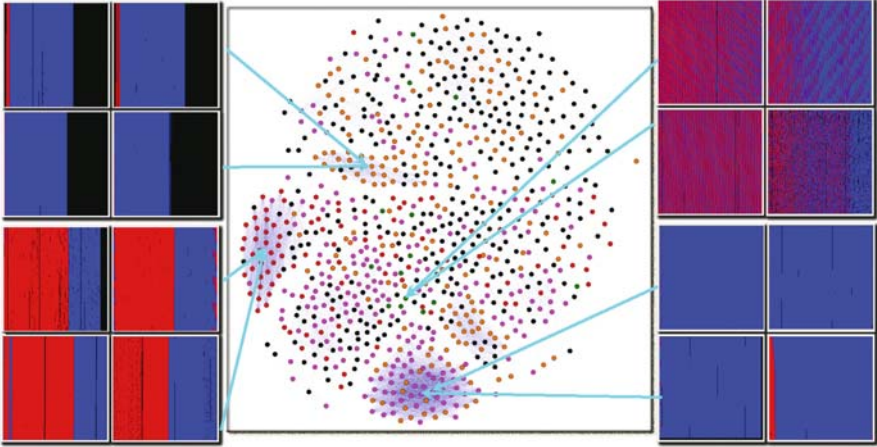
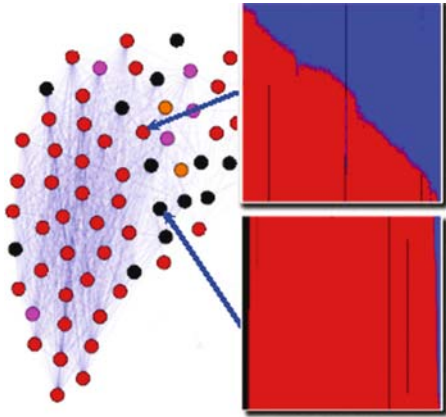


Fig. 6 A graph showing the results of the BAM classification to unknown scans. As in Fig. 5, a graph was made as in ScanVis (Muelder et al., 2005) and the nodes are colored according to a BAM classification. In this case, BAM was trained with four different patterns, so it classified nodes into four groups. Examples from each classification are shown. Outliers are shown as black nodes

Fig. 7 A close up of one cluster. Most nodes are colored red, and their detail view looks like the detail view on top. But some are not, and look different as is shown in the detail view at bottom. This distinction was not picked up by the wavelet analysis



learning techniques can quickly lead the analyst to find pattern details that were lost by the statistical analysis.

4 Future Work

One possible extension of this work would be to investigate looking at the differences between the original scans and the controlled scans that the associative memory classified them as. That is, by subtracting out the part of the pattern that

is controlled, it should be possible to isolate the distortion patterns that are created by uncontrollable effects such as router delays. Doing so should greatly enhance the capability to correlate and identify such common but minute effects, even when the underlying scans are vastly different due to large scale effects such as the scanning tool used.

Another aspect that could be considered is the application of unsupervised algorithms to this task. These would produce a clustering without the need for controlled data. This could lead to an even more effective tool for finding outliers in the graph, since each true outlier would have a unique color instead of all being grouped into one class.

Finally, it would be useful to pursue a tighter integration of intelligent techniques such as BAM with visualization systems such as ScanVis. We have demonstrated the effectiveness of such a combination in one direction, by presenting the results of applying the BAM algorithm to a graph such as that in ScanVis. It could be useful to go the other direction, and take some information learned from ScanVis and quickly feed it into the training of an algorithm such as BAM. That is, when an anomaly or trend is detected, it would be convenient to be able to select the scans of interest as training data for BAM directly from within ScanVis, and have BAM subsequently classify the rest of the graph automatically.

5 Conclusions

In the system presented in this paper, a machine learning method, BAM, has been used to perform intelligent network scan pattern reconstruction and classification. When given a set of controlled scan patterns and a noisy or incomplete pattern, the results show that the system can successfully return the complete pattern. These restored patterns are much more convenient for further studies, such as pattern comparison or pattern clustering, for the purpose of correlating malicious network activities. This paper has also shown the effectiveness of combining machine learning techniques with existing visualization and statistical techniques to create a useful visual representation for dealing with large numbers of network scans. Therefore, the results naturally lead to the feasibility of applying associative memory models in reconstruction and recognition of network scan patterns.

From an operational standpoint, one of the most important tasks in cyber security is “damage assessment”. When damage (a successful intrusion, or a detected data-exfiltration activity) is discovered at one point, security managers must immediately seek evidence of this activity more broadly, where it may not yet have been discovered. Unfortunately, this assessment is typically limited to a search for the same (outsider) IP address, under the naive assumption that the hostile agent will be using the same IP address for their activities directed at range of targets. If the intruder or outside agent employs different IP address, either in time (evolving) or in space (intentionally to avoid naive correlation) then these broader damage assessments will fail.

A system by which an intruder's "pattern", such as exemplified by scan timing characterization, could be used to quickly "re-identify" an intruder (irrespective of IP address) would greatly enhance the effectiveness of damage assessment activities, enabling detection where none was previously possible. Our system demonstrates that associative memory techniques and visualization together form an effective basis for such a characterization system. The system could quickly be trained upon the known intruder evidence, and then sought more broadly to determine the true scope of damage.

Acknowledgements This research is sponsored by the NSF CyberTrust program.

References

- Axelsson, S.: Combining a bayesian classifier with visualisation: understanding the ids. In: VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 99–108 (2004a)
- Axelsson, S.: Visualising intrusions: Watching the webserver. In: Proceedings of the 19th IFIP International Information Security Conference (SEC2004). IFIP, Toulouse, France (2004b)
- Conti, G., Abdullah, K.: Passive visual fingerprinting of network attack tools. VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 45–54 (2004)
- Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995). DOI <http://dx.doi.org/10.1023/A:1022627411411>
- Cristianini, N., Shawe-Taylor, J.: *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK (2000)
- Girardin, L., Brodbeck, D.: A visual approach for monitoring logs. In: Proceedings of the 12th Usenix System Administration Conference, pp. 299–308 (1998)
- Graps, A.: An introduction to wavelets. *IEEE Computational Sciences and Engineering* **2**(2), 50–61 (1995)
- Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. In: Proceedings of the National Academy of Sciences, vol. 79. National Academy Press, Washington, DC (1982)
- Kohonen, T.: *Associative Memories: A System Theoretic Approach*. Springer-Verlag, Berlin Heidelberg New York (1978)
- Kohonen, T.: *Self-Organization and Associative Memory*. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1989)
- Komlodi, A., Goodall, J., Lutters, W.: An information visualization framework for intrusion detection. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI) (2004)
- Kosko, B.: Bidirectional associative memories. *IEEE Trans. Syst. Man Cybern.* **18**(1), 49–60 (1988)
- MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Symposium on Math, Statistics, and Probability, vol. 1, pp. 281–297 (1967)
- McClelland, J.L., Rumelhart, D.E.: *The PDP Research Group: Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA (1986)
- McPherson, J., Ma, K.L., Krystosk, P., Bartoletti, T., Christensen, M.: Portvis: A tool for port-based detection of security events. In: ACM VizSEC 2004 Workshop, pp. 73–81 (2004)
- Muelder, C., Ma, K.L., Bartoletti, T.: A visualization methodology for characterization of network scans. *Visualization for Computer Security, IEEE Workshops*, pp. 4–4 (2005)

- nmap: (2007). URL <http://insecure.org/nmap/>
- Noack, A.: An energy model for visual graph clustering. *Lect. Notes Comput. Sci.* **2912**, 425–436 (2004)
- Parno, B., Bartoletti, T.: Internet ballistics: Retrieving forensic data from network scans. Poster Presentation, the 13th USENIX Security Symposium (2004)
- Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: *ACM Workshop on Data Mining Applied to Security (DMSA)* (2001)
- Simon, G., Xiong, H., Eilertson, E., Kumar, V.: Scan detection: A data mining approach. Technical Report AHPCRC 038, University of Minnesota – Twin Cities (2005)
- Sinclair, C., Pierce, L., Matzner, S.: An application of machine learning to network intrusion detection. In: *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, p. 371. IEEE Computer Society, Washington, USA (1999)
- Stafylopatis, A., Likas, A.: A pictorial information retrieval using the random neural network. *IEEE Trans. Software Eng.* **18**(7), 590–600 (1992)
- Tavan, P., Grubmuller, H., Kuhnel, H.: Self-organization of associative memory and pattern classification: recurrent signal processing on topological feature maps. *Biol. Cybernet.* **64**(2), 95–105 (1990)
- Werbos, P.J.: Beyond regression: New tools for regression and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University, Division of Engineering and Applied Physics (1974)

Using InetVis to Evaluate Snort and Bro Scan Detection on a Network Telescope

B. Irwin and J.-P. van Riel

Abstract This paper presents an investigative analysis of network scans and scan detection algorithms. Visualisation is employed to review network telescope traffic and identify incidents of scan activity. Some of the identified phenomena appear to be novel forms of host discovery. Scan detection algorithms used by the Snort and Bro intrusion detection systems are critiqued by comparing the visualised scans with alert output. Where human assessment disagrees with the alert output, explanations are sought by analysing the detection algorithms. The Snort and Bro algorithms are based on counting unique connection attempts to destination addresses and ports. For Snort, notable false positive and false negative cases result due to a grossly oversimplified method of counting unique destination addresses and ports.

1 Introduction

The Internet is a hostile network environment. Firewalls shelter users from the continual “storm” of probing activity (scanning) pervasive throughout the Internet. The greater proportion of this activity is generated by self-propagating malicious software such as worms (Pang et al., 2004; Yegneswaran et al., 2003). While there is value in detecting and tracking scan activity, evaluating the success of automated detection methods can be a complex exercise. This paper aims to illustrate that visualisation can contribute to the critique of algorithmic scan detection. In particular, two widely deployed open source intrusion detection systems (IDS), Snort 2.1.6 (Roesch, 1999) and Bro 1.1d (Paxson, 1999), are assessed.

The remainder of this introduction discusses the value of performing scan detection. Section 2 highlights the efforts of others with regard to network monitoring,

B. Irwin and J.-P. van Riel

Rhodes University, Grahamstown, South Africa, e-mail: b.irwin@ru.ac.za, g02v2468@campus.ru.ac.za

scan detection, and network security visualisation. The InetVis visualisation tool and key features are described in Sect. 3. Section 4 discusses the approach used to review network traffic and compares it to IDS scan detection output. It also describes the samples of network traffic. The results, including visualised examples, are presented in Sect. 5. Future work is outlined in Sect. 6, and the outcomes are concluded in Sect. 7.

1.1 The Merits and Difficulties of Scan Detection

To begin a justification of this research, we need to address the question, “of what value is scan detection?”

1.1.1 Arguments Against Scan Detection

Firstly, scanning activity is too prevalent to warrant concern with every incident (Yegneswaran et al., 2003). Secondly, in production network monitoring scenarios, detecting successful intrusions is of paramount concern, but scans merely signify vague intent. Thirdly, as stated by several authors (Gates et al., 2006; Jung et al., 2004; Simon et al., 2006), current scan detection algorithms have poor accuracy and generate too many false positives. Scan detection is a specialised case of anomaly detection. Many authors argue that anomaly detection methods are less accurate than signature-based methods (Axelsson, 2000; Kemmerer and Vigna, 2002; Verwoerd and Hunt, 2002). Furthermore, algorithms cannot be too complex, as they need to be efficient enough for real-time monitoring. For these reasons, scan detection is often left disabled or ignored in production environments.

1.1.2 Arguments for Scan Detection

Having considered arguments against scan detection, there are at least some motivations for performing scan detection.

Firstly, there is the potential to detect and contain worm activity without relying on signatures. Infected hosts and worm activity can be identified based on the scan patterns they produce. Compared to matching traffic against a large signature database, a general scan detection algorithm could be more scalable and detect zero-day worm attacks for which no signatures exist. Scan detection can also be employed as an application of “extrusion” detection – monitoring internal hosts to detect compromised systems that attempt malicious outbound connections (Bejtlich, 2005). Readily identifying compromised internal hosts can facilitate rapid response and recovery.

There is a rationale for performing scan detection on inbound traffic. Both the Snort and Bro IDS have intrusion prevention mechanisms to trigger the injection

of new firewall rules as a response to malicious network events. As a pro-active response, once a source is identified as a scanner, subsequent connections can be blocked to prevent future exploit attempts. However, there are at least three caveats to this defence mechanism, namely denial-of-service (DoS), false positives, and distributed attacks. A malicious third party could affect DoS by initiating a scan that spoofs the address of a legitimate host. Similarly, due to the poor accuracy of scan detection algorithms, benign traffic may be misclassified as scan activity, blocking legitimate access. The third concern is that distributed attacks from multiple sources will defeat this blocking strategy, as it relies upon tracking and blocking individual malicious sources. A more cautious approach would be to maintain a list of “suspicious” hosts that, due to their scanning activity, warrant more attention, as alluded to by Verwoerd and Hunt (2002). An IDS can then assign more resources to intensive checks against this reduced set of hosts deemed suspect by virtue of their previous scanning activity.

2 Related Work

The context of this research involves network monitoring methods, intrusion detection theory, and information visualisation techniques. This section relates several contributions that the authors believe to be significant.

2.1 Intrusion Detection and the False Positive Problem

One difficulty with intrusion detection is the possibility of falsely identifying legitimate traffic as intrusive. The false positive rate is a major factor that limits the effectiveness of IDSs, an issue is well addressed by Axelsson (2000). He takes an established statistical argument known as the “base rate fallacy” and applies it to the problem of intrusion detection. It is presumed that a significant proportion of traffic in a production network is benign and, relative to this, the incidence rate of malicious activity is low. Even with high accuracy, a large volume of benign traffic and a low incidence of malicious traffic can result in an overwhelming number of false alarms.

2.2 Network Telescopes

Network telescopes are constituted by segments of unassigned IP address space where unsolicited traffic is passively captured. This averts the false positive problem because legitimate traffic will never be observed. The observations tend to be limited to scans and backscatter, because network telescope IP addresses do not initiate or respond to traffic. Similarly, honeypot networks attempt to capture only malicious

activity but, unlike telescopes, actively respond to traffic to solicit more information and improve the scope of the observations.

Harder et al. (2005) provide an example of analysing traffic from a small network telescope. They perform some statistical analysis and include some static graphics. In preliminary work, the authors (van Riel and Irwin, 2006), examine telescope traffic phenomena with an emphasis on interactive graphical analysis. However, Moore et al. (2004) argue that small network telescopes, such as a class C network, are too small to infer statistical generalisations about the Internet. Pang et al. (2004) perform a large-scale study on class A and B networks using both active and passive measurement methods. Lastly, using large telescopes, a seminal study by Moore et al. (2006) discusses the task of inferring DoS backscatter.

2.3 *Classifications of Network Scan Activity*

In describing and characterising scan activity, several synonymous terms are used in the literature. This paper adopts the definitions used by Snort documentation (Caswell and Hewlett, 2007), as it offers a broad set of categories for classifying scanning activity:

port-scan: a “one-to-one” scan where a source host attempts multiple connections to a single target (destination) host on a number of distinct destination ports. This type of scanning is also broadly termed *service discovery*, or *vertical scanning* (Yegneswaran et al., 2003).

port-sweep: a “one-to-many” scan on a given destination port, where a single host attempts to connect to multiple destination hosts. This can also be referred to as *host discovery*, *address scanning*, *vulnerability scanning*, or *horizontal scanning* (Yegneswaran et al., 2003). Host discovery can also be conducted with ICMP.

These definitions describe probing activity originating from one source alone. To evade detection, both service discovery and host discovery can be coordinated from multiple sources in a distributed manner – referred to as *distributed scanning*. Snort has the capability to detect distributed and decoy port scans (many-to-one). A single host may spoof multiple source addresses as decoys to obscure its real identity. Lastly, *stealth scans* use a variety of methods to attempt to evade detection. Stealthy techniques include distributed scanning, scanning slowly, and using specialised TCP flags – see the Nmap reference (Lyon, 2007) for more details.

2.4 *Algorithmic Approaches to Scan Detection*

Scan detection is a form of anomaly detection. The proficiency of a scan detection algorithm will be determined by how it characterises scan activity and differentiates it from normal traffic. Distinguishing between various scan types requires modelling the distinctive characteristics of traffic patterns produced by each type. One general assumption is that scan activity will generate a high number of failed connection

attempts (Lau, 2004; Caswell and Hewlett, 2007). Both Snort and Bro apply this as a base assumption in their algorithms. In essence, they simply count failed connection attempts and alert if thresholds are reached, though more sophisticated approaches are being developed. For example, advanced statistical approaches can be taken (Gates et al., 2006; Jung et al., 2004; Leckie and Kotagiri, 2002), or data mining classifiers employed (Simon et al., 2006). For the purposes of this evaluation, the scope is limited to the default algorithms offered by Bro and Snort.

2.5 Network Security Visualisation

Vision is a parallel and pre-attentive cognitive process, whereas auditory cognition (used to understand text and speech) is a serial process (Ball et al., 2004; Wickens et al., 1983). Hence, visualisation is a superior medium for correlation and pattern matching tasks such as observing anomalous traffic patterns caused by scans. However, despite these cognitive advantages, scalability is often cited as a limitation (Ball et al., 2004; Valdes and Fong, 2004; Goodall et al., 2006; Foresti et al., 2006).

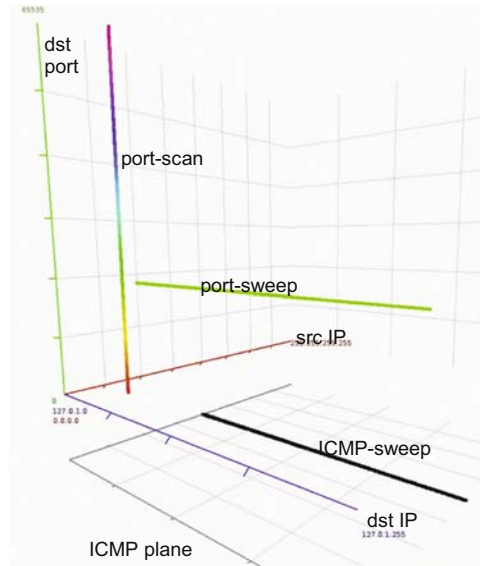
Stephen Lau's "Spinning Cube of Potential Doom" visualisation is a primary reference for developing this work (Lau, 2004). The basic 3-D scatter-plot of source IP, destination IP and destination port is well suited to displaying traffic patterns and in particular, scanning activity. Scatter-plots have a scalability advantage because points consume a minimal amount of display space. Lines are commonly used in network visualisation (Ball et al., 2004; Yin et al., 2004; Toledo et al., 2006) to provide a natural metaphor for connectivity, but use display space less efficiently.

In building on Lau's original concept, several other visualisations provided useful ideas. An enumeration of key influences follows. Valdes and Fong discuss a scalable approach with similar plots to the "Cube of Potential Doom", but in 2-D (Valdes and Fong, 2004). The "space-shield" described by Fisk et al. (2003) offers features like time-animated replay at variable replay rates, and immersive navigation. The parallel axes visualisation by Yin et al. (2004) discusses focusing on subsets of the data, often termed "drilling down". Their work also includes the concept of a time-window (Ball et al., 2004) grey-out older events to provide historic context. Etherape highlights the occurrence of new events by momentarily enlarging the thickness of lines (Toledo et al., 2006). Lastly, Kuchar et al. (2006) motivate the importance of time as an attribute for correlating data.

3 InetVis Network Traffic Visualisation

InetVis, short for Internet Visualisation, is primarily designed for reviewing network telescope traffic (van Riel and Irwin, 2006). Figure 1 illustrates the plotting scheme and basic types of scans. Lau's original visualisation plotted TCP traffic and InetVis extends this by including support for the UDP and ICMP protocols. The input for Lau's visualisation is Bro connection log files, whereas InetVis captures live traffic

Fig. 1 The InetVis 3-D scatter-plot, exhibiting common scan types. Points are plotted according to the source IP (*red-axis*), destination IP (*blue-axis*), and destination port (*vertical green-axis*). TCP and UDP traffic is plotted together in the main bounding box and ICMP traffic is plotted to the plane below. For address scanning, a port-sweep appears as a horizontal line, and similarly an ICMP-sweep appears as a line in the ICMP plane. A full port-scan appears as a vertical line. These example scans were generated with Nmap – for more Nmap scan examples, refer to previous work in (van Riel and Irwin, 2006). The default is to colour points by destination port with a rainbow colour gradient



or reads packet traces in the common Libpcap format (supported in Snort, Bro, Tcpdump, and Wireshark). InetVis offers a wealth of dynamic and interactive features intended to facilitate exploration of packet capture data.

3.1 Key Features and Enhancements

When characterising network scans, the order, and timing of probe packets is significant. InetVis includes several features to enhance the viewer's chronological sense of network activity – the time-window, replay-position, time-scale, transparent ageing, and pulse features. The time-window is relative to the replay position and acts as a filter by excluding events that are before or beyond the bounds of the window. The time-scale adjusts the replay rate to either slow or speed up playback. In conjunction, these controls manage the time frame in terms of position, size, and progression through the stream of packets.

Each feature can be independently controlled to dynamically adjust the time frame. For each control, quick adjustments can be made by slider-bars that scale the adjustment effect to the appropriate range of time. For example, in lower ranges, adjustments of the time-scale and time-window are in the order of milliseconds and seconds for fine control. Conversely, the upper ranges are adjusted in the order of hours (for the time-scale) or days, weeks, and months (for the time-window).

Transparent ageing and pulse effects can be enabled to enhance chronological salience. With transparent ageing, older events are faded out and appear diminished in contrast to newer events that appear solid (fully opaque). This creates an emphasis on newer events while maintaining a lingering sense of historic context. Enabling the brief pulse effect draws attention to newly introduced events by enlarging their points momentarily. This also helps the viewer to notice reoccurring events.

Other features allow the user to explore, isolate, and focus on interesting traffic phenomena. These features entail navigation, scaling the plot, a logarithmic plot, filtering, colour schemes, and recording output. Immersive navigation allows the user to explore within the data by moving, rotating, and zooming the view in 3-D space. The user can form additional insight into subtle patterns by viewing the data from different perspectives. To avoid disorientation, the navigation is bounded and ensures that the user cannot lose sight of the visualisation.

To explore the data in more detail, and deal with the effects of over-plotting (where points overlap each other), the plot can be scaled down into sub-networks or a smaller port range. Source and destination networks are specified in CIDR notation and the destination port range is specified by an upper and lower bound. By default, the plots linearly map the chosen network and port ranges onto their respective axes. For the destination port axis (vertical green-axis), a logarithmic scale can be applied instead of linear mapping. This resolves over-plotting that occurs at lower range due to the high proportion of traffic that targets the well-known and registered ports. For fine control, the user can adjust the base of the logarithm to control the amount of expansion at the lower port region – the greater the base, the greater the expansion at lower ports, with the effect of more contraction at higher ports.

Uninteresting traffic can be filtered with BPF syntax (as used in Tcpcdump). The syntax is complicated, but facilitates powerful and flexible filtering options that can operate on any field in the packet data. The filters are applied dynamically by rereading the packet trace data (with the caveat that large capture files incur some delay in applying the change). Colour can convey additional information and the user has a choice of several colour schemes, such as colouring by source port, source address, packet size, protocol, and so forth. Colour change interactions are dynamic, and can aid in viewing attribute relationships in the packet data.

To record output, InetVis supports capturing image snapshots or frame sequences for rendering video clips. The packets in view can also be dumped to a trace file in Libpcap format, allowing the user to review traffic with other utilities, such as Tcpcdump and Wireshark.

4 Investigative Methodology

Scanning activity is observed with InetVis, characterised, and compared to alert output produced by the Snort and Bro. In the first phase of exploration, the network telescope traffic is freely explored with InetVis. In this step, events of interest are discovered, characterised, isolated, and recorded. Case-by-case, each event is then

processed with Snort and Bro to test the accuracy of the scan detection algorithms. The third investigative phase proceeds in reverse. The full network telescope dataset is processed with Snort and Bro, and samples of the alerts are reviewed with InetVis. To explain false positive and false negative cases, the source code of the scan detection algorithm is reviewed. Lastly, to verify cases and explanations, specific test traces are created with Nmap. Fig. 1 shows an example of three basic scan types generated with Nmap. The test cases are used to confirm assessments about how the Snort and Bro scan detection algorithms function.

4.1 Network Telescope Traffic Capture

Traffic captured from a network telescope provides a sample of Internet scanning activity. A single monitoring host passively captures traffic destined for the class C network. The dataset consists of monthly packet traces captured during 2006. Due to some downtime (mainly caused by power outages), the data contains a few gaps which amount to 20.2 days (5.5%) for the 2006 year. The accumulated set of traces for 2006 contains in excess of 6.6 million IP packets. The data composition by protocol and the number of packets is 65% TCP, 20% UDP, and 15% ICMP.

The full dataset is visualised in Fig. 2. From this image, it is evident that the predominant phenomena are address scans in the form of port-sweeps and ICMP-sweeps. Conversely, port-scans are a rarity, as most sources first establish the presence of a host before expending the time to conduct a port-scan.

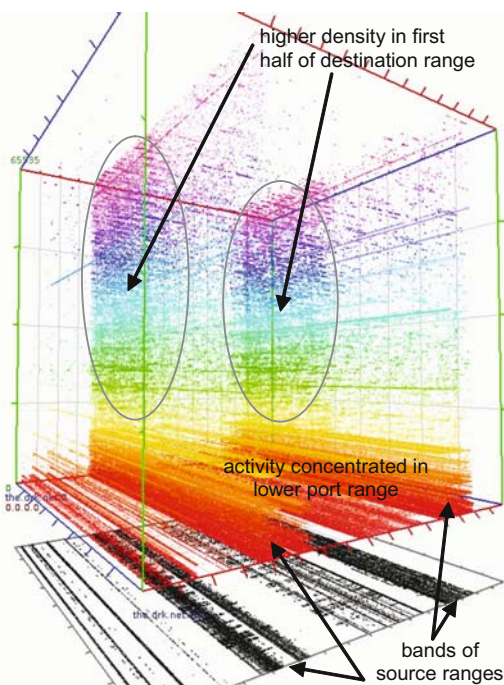
4.2 Scan Detection Configuration and Processing

The default Snort and Bro configurations were modified to focus on scan detection features. This streamlined the alert output and avoided unnecessary processing. Several iterations of configuration and testing were performed on the network telescope data as well as generated scan examples.

4.2.1 Snort Configuration

For Snort, only essential pre-processors (`flow`, `frag3`, `stream4`, `sfport-scan`) were loaded to support scan detection. All rule-sets, except `scan.rules` were disabled. Snort's `sfportscan` scan detection pre-processor is set to "low" sensitivity by default. The low setting requires a sub-minimum number of negative responses, and is named the "priority count" threshold (Caswell and Hewlett, 2007). This low setting is inappropriate for analysis with network telescope traffic, as the data will not contain any negative responses. Alternatively, the medium and high sensitivity settings do factor in negative responses, but they do not require them.

Fig. 2 6.6 Million packets captured during 2006 from a class C network telescope. Even with the large number of points, InetVis is able to maintain moderately interactive frame rates using mid-range graphics hardware (e.g. 8 FPS with a Nvidia GeForce 7600GS). The majority of port-sweeps are concentrated in the lower end of the port range. Note the banding effect by source (*red-axis*) which shows that a large proportion of activity comes from two sections of the IPv4 address space. Another interesting observation is the higher concentration of scattered activity in the first half of the destination address range (*blue-axis*)



Hence, all Snort processing on the data was conducted with either the medium or high sensitivity setting. In the context of a network telescope, disregarding negative responses is acceptable, since all of the traffic captured is unsolicited. The `sfportscan` configuration was also modified to include the detection of ACK scans (discussed further in Sect. 5.2.)

4.2.2 Bro Configuration

To focus on scan events, the default Bro “light” policy was streamlined by removing unnecessary policies intended for application protocol analysis. Bro was used to provide results in two ways. As with Snort, the initial configuration mode was used to test Bro’s scan detection with potential false positive and false negative cases. The defaults were tested as well as adaptations to attempt to match the respective threshold options for the medium and high sensitivity levels found in Snort.

The second configuration mode was designed to investigate the distribution of unique addresses targeted by address scans. The Bro scan detection policy is highly configurable, allowing exact and multiple threshold levels to be specified. This set-up entailed specifying a set of threshold values at regular intervals. Of particular interest is the unique destination address count. As an address scan progressed through the address range, it triggered an alert at each threshold. A script was written to parse the alert file, counting how many scans surpassed each threshold level.

Since a single scan triggers alerts at each threshold it passes, all but the highest alert for that scan should be counted, while previous alerts must be discounted. In addition, different time thresholds were investigated. Unlike Snort's fixed pre-sets, Bro scan detection time-outs can be redefined to an arbitrary value. Results generated from this are presented in Sect. 5.1.

4.3 Graphical Exploration and Investigation with InetVis

The network telescope data was explored month by month. To form an overview of all the events, a fast replay rate of $86,400\times$ (a day per second) was typically combined with a time window of 7 days. A month's traffic could be skimmed over in roughly 30 s. Alternatively, a static view of all the traffic was viewed with a 30-day time window. This mode suited the observation of slow scans and pseudo-random patterns formed over long periods. Patterns were identified as pseudo-random when some facet of non-randomness could be noticed – for example, scattered packets that ended up forming diagonal lines (as discussed later in Sect. 5.2). To reduce the clustering effect in the lower port range, the logarithmic scale was applied to the destination port axis. Various colour schemes were tested when searching for possible correlations.

Rapid replay rates and large time windows were suitable for observing events that progressed slowly. The details of fast events became more evident by reducing the replay rate and time window. Identified events could be focused on by scaling the view and setting the ranges on axes, namely the source sub-network, destination sub-network and destination port range. This “drilled down” into subsets of the data, facilitating a clearer perspective of the event. By reducing the range of data viewed, smaller scale events became more evident. Further reduction of the time window and replay rate was used to analyse very rapid events. In addition, BPF filters were applied to isolate events of interest.

Once incidents were isolated, they were recorded to capture files for further analysis and testing. The captured files were processed with Snort and Bro. For obvious scans identified with InetVis, the failure to alert indicated a false negative. Furthermore, alert output was inspected to check that detected scans were correctly characterised by the detection algorithm. Alternatively, the Snort and Bro alert logs for each month could be inspected and then investigated with InetVis. Using the alert information to set the appropriate replay position, ranges, and filters eased seeking out the event.

5 Results and Analysis

Network telescope traffic review with InetVis enabled the observation of many anomalous traffic patterns that could not be noticed with IDS alert output. The results presented in this section focus on particular findings that illustrate possible

flaws in the Snort and Bro scan detection algorithms. Much of the discussion entails two attributes used to characterise scans; the unique destination IP address count, and the unique destination port count.

5.1 Address Scans and the Distribution of Unique Addresses

Snort and Bro scan detection algorithms count unique destination ports and addresses. A combination of thresholds determine if scanning activity has occurred and setting appropriate threshold levels is a question of parameter optimisation. Snort’s “high”, “medium”, and “low” sensitivity pre-sets are hard-coded threshold combinations with limited scope for optimisation.

With the flexibility afforded by Bro, multiple threshold levels were tested at varied time-out values. The bar chart in Fig. 3 shows the distribution of address scan alerts categorised by the number of unique destination addresses that were targeted. The chart exhibits a full range of thresholds from 9 to 256, grouped by intervals of 8. Essentially, it shows the number of address scans that reached a higher number of unique destination addresses. Furthermore, each address threshold interval is sub-categorised by “light”, “default” and “heavy” time-outs. The expiry time-outs are 1 min, 5 min, and 10 h, respectively.

Firstly, note the right-hand tail of the distribution in Fig. 3. This shows that the greater proportion of the scanning activity targets almost all the addresses in the class C network telescope. The plot in Fig. 4 expands the density of activity in this upper range for both the TCP and UDP address scans (unfortunately, Bro 1.1d does not readily facilitate multiple threshold levels for ICMP). Once again, the greater proportion of scans cover nearly the entire address range. Interestingly, compared to UDP, TCP exhibits this characteristic to a greater extent.

Returning to Fig. 3, the lower range of the distribution also exhibits a tail, but to a lesser extent. Presumably, the tail is caused by miscellaneous non-scan activity. This suggests the obvious – setting the IP address threshold too low increases the number of false-positives. Combined with heavy time-outs, a low unique destination

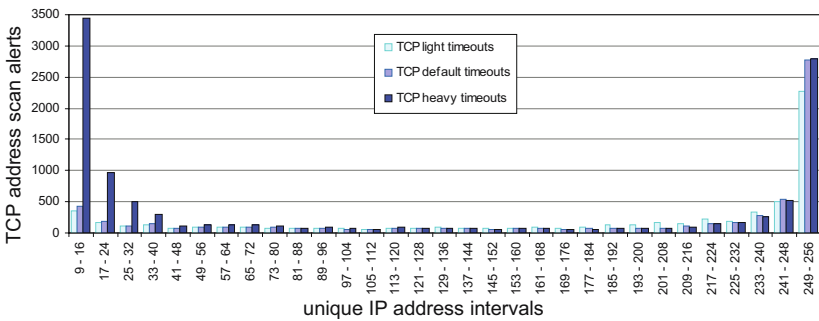
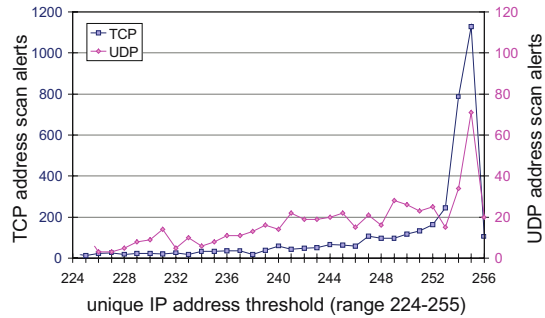


Fig. 3 Number of Bro scan alerts categorised by unique address intervals and time thresholds

Fig. 4 The number of TCP and UDP address scan alerts are plotted (y-axis) for the upper range 224–256 of unique destination address thresholds (x-axis). The count for TCP scan alerts count range is on the left, and ranges between 0 and 1,200. For UDP on the right, the range is 0–120 ($10\times$ smaller than TCP)



IP threshold will result in excessive false positives. While lighter time-outs avoid this to some extent, they miss slower stealth scans.

Another difficulty with time-outs seems somewhat counter-intuitive initially. One might expect that heavy time-outs would pick up more scanning activity. However, careful observation of the upper ranges in Fig. 3 shows that for some intervals the “light” and “default” values are higher, bar the final interval where the “default” and “heavy” values are significantly higher. These offsets can be explained in two ways: either, heavy time-outs count several individual scan incidents as one longer scan, or, if a scan’s timing between packets is inconsistent, lighter time-outs miscount one long scan as two or more smaller scans.

In summary, higher unique destination IP thresholds will pick up the majority of scanning activity while avoiding false positives. Time-out thresholds should not be set too long, nor too slow. Clearly, the algorithm needs an improved method of timing activity, so as not to confuse multiple scans as one, or one scan as multiple. A similar issue was discovered with Snort, which also reports one long scan as multiple shorter scans.

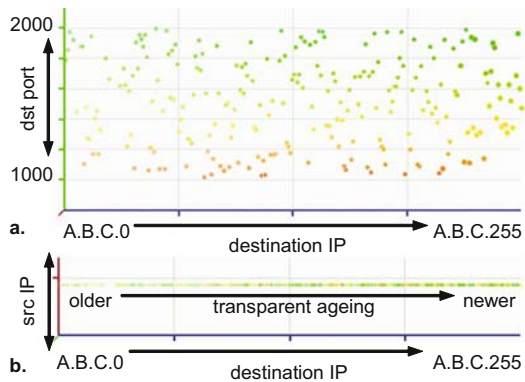
5.2 Scans Discovered and Characterised with InetVis

Multitudes of scanning incidents have been identified by human inspection with InetVis. Selected examples are presented to illustrate false negative and false positive issues with the Snort and Bro scan detection algorithms.

5.2.1 Pseudo-Random Phenomena

Some traffic captured from the network telescope exhibited pseudo-random patterns when visualised with InetVis. In general, there are three foreseeable explanations for the pseudo-random phenomena. They are caused by miscellaneous network configuration errors, DoS backscatter, or subversive stealthy scanning techniques. Evasive scanning methods employ randomisation, dispersion, patience or a combination

Fig. 5 Rapid 50 ms pseudo-random host discovery with probe packets dispersed by destination port. The image is shown with a 75 ms time-window, transparent ageing, point-pulse for new events, and coloured by destination port. (a) Front-view showing destination port vs. destination IP. (b) Top-view showing source IP vs. destination IP



thereof. Very slow scans are likely to fall outside of the bounds of detection time-window thresholds. The authors believe that if a scanning method is sufficiently well dispersed (randomised), it can occur rapidly while evading detection, as shown in Fig. 5.

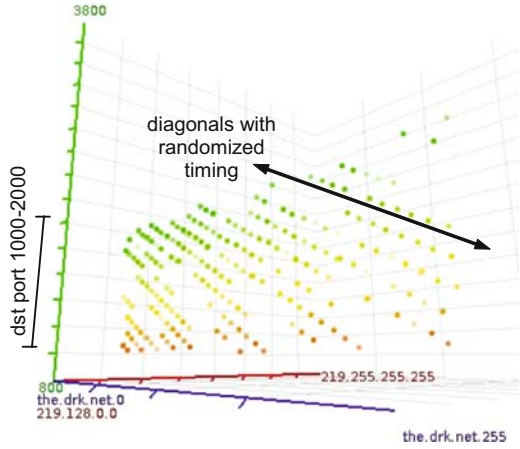
There are two orthographic projections. Figure 5a provides a frontal 2-D view exhibiting a dramatically fast scatter of packets randomly dispersed about the destination port range 1,000–2,000. The transparent fading of older packets shows the address range is traversed in a linear progression. Within 50 ms, each of the 232 packets targets a unique destination IP. Given the fast transmission rate, a few packets may have been lost. Figure 5b shows a top view, emphasising how almost the entire address range is covered.

Upon closer inspection, all packets originate from source port 80 and have SYN/ACK TCP flags set. Assuming the normal connection establishment procedure, a SYN/ACK response indicates that the port is accepting connections. However, the telescope does not initiate any connections. Two explanations are raised: (1) The observed traffic could be backscatter from a web server undergoing a DoS attack where the telescope’s source address was forged (spoofed) – DoS attacks commonly spoof source addresses to hide the identity of the attacker (Moore et al., 2006). (2) Alternatively, this is a form of stealthy host discovery, constituting an address scan.

Supposing the phenomena were backscatter, it is curious that the spoofed addresses were not randomised and selected from the greater address range of over 4 billion IPv4 addresses. Instead, 232 consecutive addresses are probed in a linear fashion – this can be noted from the transparent ageing effect in Fig. 5. Added to this, each packet targets a unique address, and this leads the authors to favour the address scan explanation.

Although the pseudo-random dispersion in Fig. 5 is not a port-sweep by strict definition, it could be a well-adapted alternative to ICMP ping-sweeps. ICMP echo requests and responses are sometimes administratively filtered (by routers and firewalls) to safeguard against attackers who leverage ICMP as a reconnaissance tool. By using TCP source port 80 and setting the TCP flags to SYN/ACK, connection state unaware firewalls will pass this type of traffic. If a destination host receives

Fig. 6 Pseudo-random diagonal phenomenon dispersed about the destination port range 1,000–2,000. The view is in 3-D perspective projection with a 36 h time-window and coloured by destination port. The *red axis* in the background represents the source address range, which is scaled to a/9 network block (half a class A). In the foreground, the *blue axis* represents the destination address range – the network telescope’s range (“the.drk.net”). The *vertical green axis* represents the destination port range from port 800 to 3,800



an unexpected SYN/ACK packet for a connection it did not initiate, the standard response is to send an RST packet back to the source. Doing so confirms the presence of a host, while no response may indicate that the address is not used (unless the destination network policy does not follow RFC 793 and, similar to the case for ICMP echoes, administratively filters out TCP RST packets).

The pseudo-random phenomenon is not an isolated incident. Repeated incidents occur, and several other slower forms have been observed, characteristically bounded in the destination port range 1,000–2,000. The phenomenon in Fig. 6 bears some resemblance to that in Fig. 5, but note the obvious diagonals. The incident occurs in a much longer time frame, 36 h rather than 50 ms. Furthermore, the progression across the address range is randomised and repetitive, as not every packet targets a unique destination IP address.

The Snort `sfportscan` algorithm does not alert on the activity in Fig. 5, a possible false negative. By contrast, the Bro scan algorithm does alert on this kind of pattern (provided the packets are altered to SYN packets, as Bro handles SYN/ACK packets differently). Bro detects this activity because, unlike Snort, its algorithm does not consider the destination ports when identifying address scans. Arguably, this leaves it more susceptible to false positives. While Fig. 5 illustrates a somewhat ambiguous case for address scanning, the incident in Fig. 6 is less likely. With extended time-outs, Bro detects Fig. 6 as an address scan whereas Snort does not produce any alerts. Whether or not such activity should be classified as address scans or backscatter remains debatable.

5.2.2 Multiple Synchronous Sweep Scans

While it is not completely clear if the examples in Figs. 5 and 6 should be detected as scans, a far more obvious case of address scan (port-sweep) activity is shown in

Fig. 7 Simultaneous port-sweeps. Front-on 3-D perspective projection shown with 150 s time-window and transparent ageing. Colour by destination port. Slightly oversized points at the end of scans highlight the most recent packets. The six port-sweeps occur on ports 42 (WINS name service), 139 (SMB/CIFS over NetBios), 445 (SMB/CIFS over TCP), 4,899 (radmin windows based remote administration tool), 5,900 (VNC remote desktop), 6,101 (Verias BackupExec)

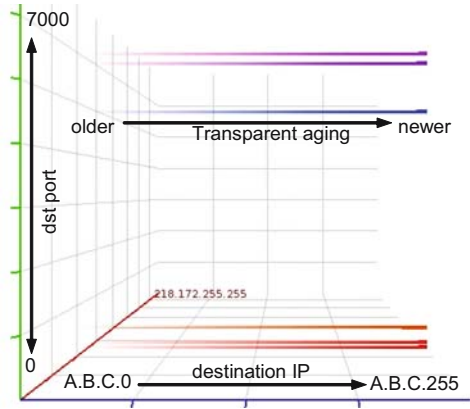


Fig. 7, where six simultaneous scans linearly traverse the address range. This is a multi-vector attack, where each port is associated with one or more vulnerabilities. Remarkably, Snort’s `sfportscan` detector produces no alerts for the port-sweeps, while Bro somewhat misreports the activity.

5.2.3 The Snort False Negative Case

Tested with Nmap, Snort is capable of detecting single instances of port-sweeps. For the multi-port-sweep example, it might be assumed that Snort’s `sfportscan` module would produce either six separate port-sweep alerts, or a single alert for the whole event. Yet no alerts were produced.

Explaining this false negative required review of Snort’s source code. The fault is attributed to an over-simplified implementation for counting unique destination addresses and ports. For each source address, instead of maintaining a set of unique destinations, only the previous destination address is kept in memory, and a similar case applies to tracking destination ports. The current destination is compared with just the previous destination, and increments the unique counter if they do not match. This fails to consider the complete history of destinations within the chosen detection time-window. Effectively, it makes the poor assumption that a port-sweep or port-scan will be efficient and not strike the same destination twice.

In Fig. 7, as multiple port-sweeps progress simultaneously, there is alternation between six ports. This causes repeated hits, and the unique port count is continuously incremented instead of remaining at six. The port count functions as a maximal threshold when detecting address scans. If the unique destination port count is above the threshold, the algorithm rejects the possibility of a port-sweep, since traffic apparently occurs on too many distinct ports – the feature that prevents Snort from alerting on the pseudo-random activity in Fig. 5. Thus, a false negative results from over-counting because the algorithm uses both minimal and maximal

Fig. 8 Snort `sfportscan` log output for a false positive case. The test Nmap scan only targeted two addresses, yet the unique “IP count” is 30. Furthermore, the “IP count” is clearly inconsistent with the “Scanned IP Range” field which specifies a range of two addresses from 127.0.1.2 to 127.0.1.3

```

Time: 05/30/07-12:45:09.413192
event_id: 30
160.0.0.1 -> 127.0.1.3 (portscan) TCP
Filtered Portsweep
Priority Count: 0
Connection Count: 30
IP Count: 30
Scanned IP Range: 127.0.1.2:127.0.1.3
Port/Proto Count: 1
Port/Proto Range: 32000:32000

```

thresholds to distinguish port-sweeps from port-scans. This false negative case also applies to port-scans, as multiple alternating port-scans will go undetected by Snort.

The Bro IDS does produce alert output for the example in Fig. 7, but fails to identify the complete event. In the specific case, it alerted at each set threshold, but only for one out of the six ports, thereby missing the five other scans.

5.2.4 The False Positive Corollary

The Snort flaw discussed above also generates false positives. A false positive can arise if, for a given source, connection attempts repeatedly alternate between two destination addresses, or two destination ports. Instead of recognising the recurring connection attempts to previous destinations, `sfportscan` continually increments the counters for unique destination addresses and ports. The counter then surpasses a minimal threshold which triggers either a port-sweep or port-scan false alert. As proof of concept for this flaw, a special packet trace was generated with Nmap. Multiple alternating TCP SYN connection initiation packets were sent to just two destination addresses on the loop-back interface. The packets were simply alternated 20 times between 127.0.1.2 and 127.0.1.3, totalling 40 packets. This was sufficient to test the pre-sets for the thresholds. Figure 8 provides an example of `sfportscan` log output for this false positive. Another observation was that Snort logs the event as soon as the threshold is reached, thereby failing to report the full extent of a scan.

6 Future Work

This work tested the default scan detection in Snort and Bro. As alluded to in Sect. 2.4, other scan detection algorithms have been devised. Bro includes an implementation of the algorithm by Jung et al. (2004) and may offer the flexibility to re-implement and compare other algorithms (Gates et al., 2006; Simon et al., 2006; Leckie and Kotagiri, 2002) for future analyses.

To ease the process of conducting visual analysis, the authors are devising semi-transparent visual overlays to represent detected scans. The detection of scans can then be seen against the backdrop of the network traffic. To complement this, the intention is to include support for reading scan alert output and automatically forward the replay position to detected scans. Automatic focus of the scan event would also be desirable.

Another worthwhile investigation would quantitatively assess the performance advantage of Snort's simplified pseudo-unique destination counter. In conjunction, one might look at the accuracy cost of this simplification by judging how many false positives it generates. While Bro maintains a set of all previous destinations, this adds complexity and makes the scan detection more resource-intensive (in terms of memory consumption and processor time). Of course, this kind of analysis should make use of production traffic to test real-world performance.

7 Conclusion

This work exhibits the practical application of visualisation to the problem domain of scan detection. InetVis re-implements and extends Stephen Lau's original visualisation concept, adding several enhancements for visualising network telescope traffic and scanning activity. Special attention is paid to chronological salience, allowing the exact order of probing packets to be observed. Several months of network telescope traffic were explored and investigated with InetVis. From select scan incidents, false positive and false negative cases were established for the Snort `sportscan` module, and this inaccuracy was attributed to a unique destination-counting flaw. While Bro did not suffer from this flaw, it too failed to report the full extent of scan activity, as shown with simultaneous port-sweeps in Fig. 7. Pseudo-random phenomena were discussed in Sect. 5.2, and Fig. 5 could be a stealthy form of host discovery. It illustrates the difficulty of dealing with ambiguous traffic patterns that could be a form of ACK scanning or backscatter.

InetVis showcases the advantages of using visualisation, and the 3-D scatter-plot proves to be a suitable choice for displaying scan activity. Without InetVis, the authors would have a weaker understanding of scan phenomena, and would not have discovered these issues in the Snort and Bro scan detection algorithms.

References

- Axelsson S (2000) The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security* 3(3):186–205
- Ball R, Fink GA, North C (2004) Home-centric visualization of network traffic for security administration. *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, 55–64. ACM Press, New York, USA

- Bejtlich R (2005) *Extrusion Detection: Security Monitoring for Internal Intrusions*. Addison-Wesley, Reading, MA
- Caswell B, Hewlett J (2007) *Snort Users Manual*, Version 2.6.1. http://www.snort.org/docs/snort_manual.pdf. Accessed 3 April 2007
- Fisk M, Smith SA, Weber PM et al (2003) Immersive network monitoring. PAM '03: 2003 Passive and Active Measurement Conference. <http://woozle.org/~mfisk/papers/pam03.pdf>. Accessed 14 November 2007
- Foresti S, Agutter J, Livnat Y et al (2006) Visual correlation of network alerts. *IEEE Computer Graphics and Applications* 26(2):48–59
- Gates C, McNutt, JJ, Kadane JB et al (2006) Scan detection on very large networks using logistic regression modeling. ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications, 402–408. IEEE Computer Society, Washington, DC, USA
- Goodall JR, Lutters WG, Rheingans P et al (2006) Focusing on context in network traffic analysis. *IEEE Computer Graphics and Applications* 26(2):72–80
- Harder U, Johnson M, Bradley JT et al (2005) Observing internet worm and virus attacks with a small network telescope. PASM '05: Proceedings of the Second International Workshop on the Practical Application of Stochastic Modelling. ENTCS, 151(3):47–59. doi:10.1016/j.entcs.2006.03.011
- Jung J, Paxson V, Berger AW et al (2004) Fast portscan detection using sequential hypothesis testing. SP '04: Proceedings of the 2004 IEEE Symposium on Security and Privacy, 211–225. IEEE Computer Society, Los Alamitos, CA, USA
- Kemmerer RA, Vigna G (2002) Intrusion detection: a brief history and overview (supplement to *Computer Magazine*). *Computer* 35(4):27–30
- Kuchar OA, Hoeft TJ, Havre Susan et al (2006) Isn't it about time? *IEEE Computer Graphics and Applications* 26(3):80–83
- Lau S (2004) The spinning cube of potential doom. *Communications of the ACM* 47(6):25–26
- Leckie C, Kotagiri R (2002) A probabilistic approach to detecting network scans. NOMS '02: Network Operations and Management Symposium, 359–372. IEEE Computer Society, Washington, DC, USA
- Lyon G (2007) Nmap reference guide (man page). <http://insecure.org/nmap/man/>. Accessed 11 June 2007
- Moore D, Shannon Collen, Voelker GM et al (2004) *Network telescopes: technical report*. CAIDA, San Diego. <http://www.caida.org/publications/papers/2004/tr-2004-04/tr-2004-04.pdf>. Accessed 10 April 2007
- Moore D, Shannon C, Brown DJ et al (2006) Inferring internet denial-of-service activity. *ACM Transactions Computer System* 24(2):115–139
- Pang R, Yegneswaran V, Barford P et al (2004) Characteristics of internet background radiation. IMC '04: Proceedings of the Fourth ACM SIGCOMM Conference on Internet Measurement, 27–40. ACM Press, New York, USA
- Paxson V (1999) Bro: a system for detecting network intruders in real-time. *Computer Networks* 31(23–24):2435–2463
- Roesch M (1999) Snort – lightweight intrusion detection for networks. LISA '99: Proceedings of the 13th USENIX Conference on System Administration, 229–238. USENIX Association, Berkeley, CA, USA
- Simon GJ, Xiong H, Eilertson E, et al (2006) Scan detection: a data mining approach. SDM '06: Proceedings of the Sixth SIAM International Conference on Data Mining, 118–129. SIAM, Philadelphia, USA
- Toledo J et al (2006) EtherApe: A Graphical Network Monitor. <http://etherape.sourceforge.net/>. Accessed 23 April 2006
- Valdes A, Fong M (2004) Scalable visualization of propagating internet phenomena. VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, 124–127. ACM Press, New York, USA

- van Riel J-P, Irwin BV (2006) InetVis, a visual tool for network telescope traffic analysis. Afrigraph '06: Proceedings of the Fourth International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, 85–89. ACM Press, New York, USA
- Verwoerd T, Hunt R (2002) Intrusion detection techniques and approaches. *Computer Communications* 25(15):1356–1365
- Wickens C, Sandry D, Vidulich M (1983) Compatibility and resource competition between modalities of input, central processing, and output. *Human Factors* 25(2):227–248
- Yegneswaran V, Barford P, Ullrich J (2003) Internet intrusions: global characteristics and prevalence. *SIGMETRICS Performance and Evaluation Review* 31(1):138–147
- Yin X, Yurcik W, Treaster M, et al (2004) VisFlowConnect: netflow visualizations of link relationships for security situational awareness. *VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, 26–34. ACM Press, New York, USA