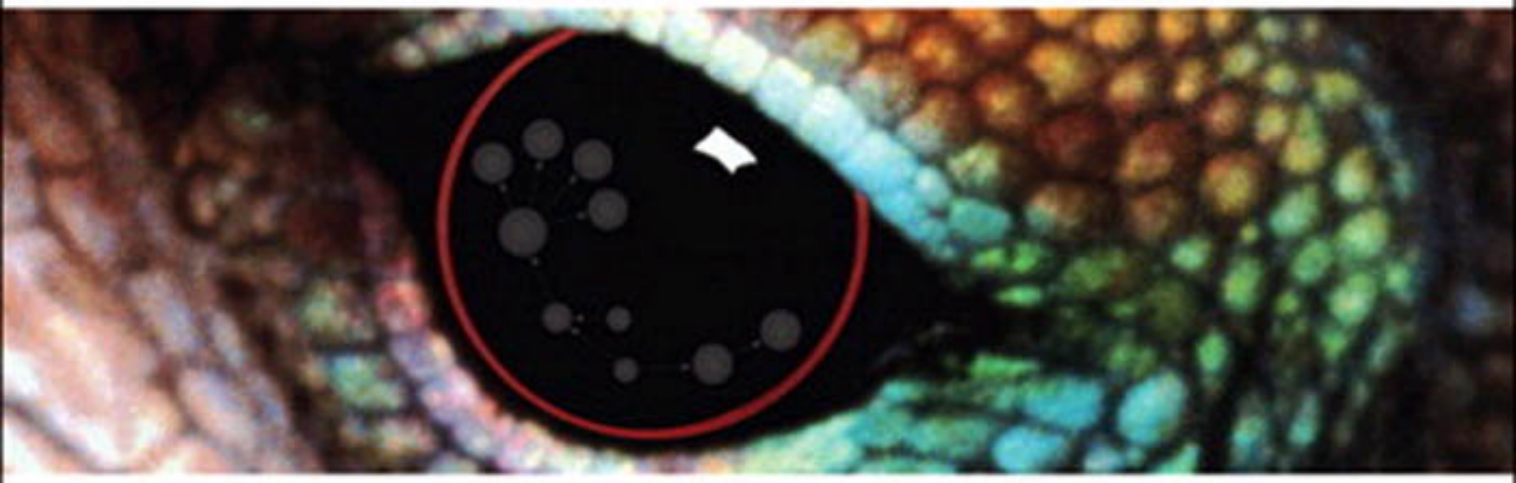
"Collecting log data is one thing, having relevant information is something else. The art to transform all kinds of log data into meaningful security information is the core of this book. Raffy illustrates in a straight forward way, and with hands-on examples, how such a challenge can be mastered. Let's get inspired."

—**Andreas Wuchner,** Head of Global IT Security, Novartis

APPLIED SECURITY VISUALIZATION



RAFFAEL MARTY

Library of Congress Cataloging-in-Publication Data:

Marty, Rafael, 1976-
Applied security visualization / Rafael Marty.
p. cm.
Includes index.
ISBN 0-321-51010-0 (pbk. : alk. paper) 1. Computer networks—Security measures 2. Information visualization. 3. Computer security. I. Title.

TK5105.59.M369 2008
005.8—dc22

2008023598

ISBN-13: 978-0-321-51010-5
ISBN-10: 0-321-51010-0
Text printed in the United States on recycled paper at RR Donnelley, Crawfordsville, Indiana.
First printing August 2008

Editor-in-Chief
Karen Gettman

Acquisitions Editor
Jessica Goldstein

Senior Development Editor
Chris Zahn

Managing Editor
Kristy Hart

Project Editor
Andy Beaster

Copy Editor
Keith Cline

Indexer
Erika Millen

Proofreader
Jennifer Gallant

Publishing Coordinator
Romny French

Multimedia Developer
Dan Scherf

Book Designer
Chuti Prasertsith

Composition
Nonie Ratcliff

Graphics
Tammy Graham
Laura Robbins

Preface

This book is about visualizing computer security data. The book shows you, step by step, how to visually analyze electronically generated security data. IT data must be gathered and analyzed for myriad reasons, including GRC (governance, risk, and compliance) and preventing/mitigating insider threats and perimeter threats. Log files, configuration files, and other IT security data must be analyzed and monitored to address a variety of use-cases. In contrast to handling textual data, visualization offers a new, more effective, and simpler approach to analyzing millions of log entries generated on a daily basis. Graphical representations help you immediately identify outliers, detect malicious activity, uncover misconfigurations and anomalies, and spot general trends and relationships among individual data points. Visualization of data—the process of converting security data into a picture—is the single most effective tool to address these tasks. After all…

A picture is worth a thousand log entries.

To handle today's security and threat landscape, we need new analysis methods. Criminal activity is moving up the network stack. Network-based attacks are becoming more sophisticated, and increasingly attacks are executed on the application layer.

Criminal techniques have adapted. Are you prepared to deal with these new developments? Are you aware of what is happening inside of your networks and applications? In addition to monitoring your networks, you must make sure you are taking an in-depth look at your applications. Because of the vast amount of data that requires analysis, novel methods are needed to conduct the analysis. Visualization can help address these complex data analysis problems.

WHAT THIS BOOK COVERS

Follow me on an exciting journey through security data visualization. We will start with the basics of data sources needed for security visualization. What are they? What information do they contain, and what are the problems associated with them? I then discuss different ways to display data in charts or more complex visualizations, such as parallel coordinates. You will learn which graphical methods to use and when. The book then takes you through the process of generating graphical representations of your data. A step-by-step approach guarantees that no detail is left out. By introducing an **information visualization process,** visualization of security data becomes a simple recipe, which I apply in the core of this book to analyze three big areas of security visualization: perimeter threat, compliance, and insider threat. These chapters are hands-on and use-case driven. Open source visualization tools and libraries are discussed in the last chapter of the book. You can find all the tools introduced on the accompanying CD. Without dealing with installations, you can immediately start analyzing your own security data.

The book is a **hands-on** guide to visualization. Where it covers theoretical concepts and processes, it backs them up with examples of how to apply the theory on your own data. In addition to discussing—step by step—how to generate graphical representations of security data, this book also shows you how to analyze and interpret them.

The goal is to get you excited and inspired. You are given the necessary tools and information to go ahead and embed visualization in your own daily job. The book shows example use-cases that should inspire you to go ahead and apply visualization to your own problems. If one of the chapters covers a topic that is not your responsibility or focus area (for example, compliance), try to see beyond the topic specifics and instead explore the visualizations. The concepts may be valid for other use-cases that you want to address.

WHAT THIS BOOK DOESN'T COVER

This book covers visualization of computer security data. I do not discuss topics such as binary code or malware analysis. I don't get into the topics of steganography (the art or science of hiding information in images) or system call visualizations. This book is about time-based data and system status records. The data visualized is data you use to operationally secure an organization.

This book is not a compendium of security data sources and possible visual representations. It uses existing visualization methods—charts, parallel coordinates, treemaps, and so on—that are supported by many tools and applications. The book is composed of a sample set of data sources and use-cases to illustrate how visualization can be used.

AUDIENCE

I wrote this book for security practitioners. I am introducing new ways to analyze security data to the people who can implement them. Whether you are analyzing perimeter threat issues, investigating insider crimes, or are in charge of compliance monitoring and reporting, this book is meant for you.

The reader should have a basic understanding of programming to follow the Perl and UNIX scripts in this book. I assume that you are familiar with basic networking concepts and have seen a log file before. You don't have to be an expert in IT security or compliance. It helps to have an understanding of the basic concepts, but it is definitely not a prerequisite for this book. Most of all, I want you to read this book with an open mind. Try to see how visualization can help you in your daily job.

STRUCTURE AND CONTENT

This book follows a simple organization. It introduces basic visualization and data graphing concepts first. It then integrates those concepts with security data and shows how you can apply them to security problems. In the following list, I briefly describe each chapter:

- Chapter 1: Visualization

  Visualization is the core topic of this book. The first chapter introduces some basic visualization concepts and graph design principles that help generate visually effective graphs.

- Chapter 2: Data Sources

  Visualization cannot exist without data. This chapter discusses a variety of data sources relevant to computer security. I show what type of data the various devices generate, show how to parse the data, and then discuss some of the problems associated with each of the data sources.

- Chapter 3: Visually Representing Data

  Data can be visualized in many different ways. This chapter takes a closer look at various forms of visualizations. It first discusses generic graph properties and how they can help encode information. It then delves into a discussion of specific visualizations, such as charts, box plots, parallel coordinates, links graphs, and treemaps. The chapter ends with a discussion of how to choose the right graph for the data visualization problem at hand.

- Chapter 4: From Data to Graphs

  This chapter introduces the information visualization process. It is a step-by-step process that guides you through how to take the data and generate a graphical representation of it. It also discusses how to interpret the resulting visual representation. In addition, the chapter discusses ways to process data with various tools, such as UNIX scripts or Perl.

- Chapter 5: Visual Security Analysis

  Visually analyzing security data can be separated into three classes: reporting, historical analysis, and real-time monitoring. Historical analysis I discuss in four sections: time-series visualization, correlation graphs, interactive analysis, and forensic analysis. These are the topics discussed in this chapter.

- Chapter 6: Perimeter Threat

  This chapter is a collection of use-cases. It starts out with a discussion of use-cases involving traffic-flow analysis. Everything from detecting worms to isolating denial-of-service attacks and monitoring traffic-based policies is covered. The use-cases are then extended to firewall logs, where a large firewall log is analyzed first. In a second part, firewall logs are used to assess the ruleset to find potential misconfigurations or security holes. Intrusion detection signature tuning and wireless access log analysis are the next two use-cases that deal with network layer data. The remainder of the chapter looks at application layer data. Email server logs are first analyzed to find open relays and identify email-based attacks. A second part then looks at social network analysis using email transaction logs. The chapter closes with a discussion of visualizing vulnerability scan data.

- Chapter 7: Compliance

  This chapter first introduces compliance in a log analysis context. I discuss the basics of control objectives and policies and show which federal or industry regulations require companies to analyze and collect their logs. I then show how visualization can help analyze audit data for compliance. Going through this process, it becomes necessary to start mapping the log files against business processes to weigh their importance. This leads into a risk management discussion and shows how risk-centric security visualizations can be generated. The chapter finishes up with a discussion of two compliance use-cases: the visualization of separation of duties in an application context and the monitoring of databases.

- Chapter 8: Insider Threat

  Instead of looking from the outside in, insider threat focuses on monitoring inside the perimeter. This chapter first introduces the topic and discusses different aspects of it, such as who a typical insider is. The chapter then introduces a detection framework that helps assess and monitor individuals. Through the use of so-called precursors, we can then identify potential malicious insiders and find users behaving suspiciously. Visualization is a key component of the insider detection process.

- Chapter 9: Data Visualization Tools

  After a short introduction to different data formats used by visualization tools, this chapter then surveys visualization tools and libraries. The chapter then introduces about 20 tools and open source visualization libraries that you can use in your own programs. All of these tools are also available on the accompanying CD, the Data Visualization and Analysis Linux (DAVIX).

COLOR

Color is a key property of information visualization. Unfortunately, the cost of printing a book in color is quite high. This is why the images in the book are printed in black and white. However, because color is an important graph property, the book contains an insert of 16 color pages in the middle of the book. This insert is a collection of figures from throughout the book that illustrate how color enhances the readability of the visualizations. The following table lists the figures that are featured in the color insert.

Color Insert Table Figures that appear in the color insert

Figure Number	Page Number
Figure 3-1	68
Figure 3-17	86
Figure 3-27	95
Figure 3-39	116
Figure 4-10	141
Figure 4-11	143
Figure 4-12	146

continues

Color Insert Table Figures that appear in the color insert *(continued)*

5 Visual Security Analysis

The beginning of this book introduced all the building blocks necessary to generate graphs from security-related data. I have discussed some of the data sources that you will encounter while analyzing security data. The discussion showed what information each of the sources records and what some of the missing information is. I discussed the different graphs and how to most effectively apply them to your problems, and after that I introduced the information visualization process, which guides you through the steps necessary to generate meaningful visual representations of your data. As the last step of the information visualization process, I briefly touched on the analysis and interpretation of the graphs generated.

This chapter elaborates on that concept and shows different ways of analyzing security data using visual approaches. I separate the topic of graph analysis into three main categories:

- Reporting
- Historical analysis
- Real-time monitoring

Reporting is about communicating and displaying data. Historical analysis covers various aspects of analyzing data collected in the past. The motivations and use-cases are manifold. They range from communicating information to investigate an incident

(or other type of problem) to analyzing the data to comprehend the underlying data and situations reported. We discuss the topic of **historical analysis** by separating it into three subcategories:

- Time-series visualization
- Correlation graphs
- Interactive analysis
- Forensic analysis

After looking at historical analysis, we follow up with **real-time monitoring,** which, in the context of visualization, heavily uses the concept of **dashboards.** I discuss some of the main criteria for building effective dashboards for communicating security data. A topic closely related to dashboards is **situational awareness.** I discuss the importance of real-time insight into key security-relevant areas and how it can be used to drive decisions and react to, or proactively address, upcoming problems.

Each of the four historical analysis sections focuses on **analysis.** This material is not about how specific sets of data can be represented in a graph; we have already discussed that in the beginning of the book. Here the focus is on how to represent different sets of data to analyze and compare them. What are some of the common ways of analyzing security data? In some cases, we will see that the analysis requirement will influence how to create visualizations of the individual datasets, creating a closed loop with the graph-generation process.

REPORTING

One of the most often used techniques to communicate and display data is the **report.** Reports are not the prime example for showing the advantages of visualization. However, visualization in the form of graphs is a commonly used tool to improve the effectiveness of reports. Reports are a great tool for communicating and summarizing information. Reporting ranges from status reports, which show, for example, the type of network traffic the firewall blocked during the past seven days to compliance reports used to prove that certain IT controls are in place and operating.

A report can consist of just text, a simple graph, a combination of a graph and text, or a collection of graphs with optional text. A report based only on text is not something I discuss here. This book is concerned with visual communication of information and not textual encoding. One of the key properties of a report is that it focuses on past information. Reports are generated ad hoc, when needed, or on a scheduled basis. For example, I am using a weekly report to see the amount of blocked traffic per protocol that targeted

my laptop. The report is emailed to me so that I get a feeling for what the attack landscape looked like during the past week. Instead of using textual information, I am using graphs to summarize the data visually. Figure 5-1 shows an example of such a report.

The prevalent data source for reports is a database. The advantage of having the data in a database—over, for example, a file—is that SQL can be used to process the data before the report is generated. Operations such as filtering, aggregating, and sorting are therefore easy to apply. In some cases, reports can be generated directly from log files, which I did for the report in Figure 5-1. However, this might require some more manual data processing. The one type of data source that is not well suited to reporting is real-time feeds. Reports are static in nature and represent a snapshot in time. In contrast to reports, dashboards are designed to deal with real-time data. More about dashboards a little later. Apart from the real-time data sources, most other data sources are well suited to reporting.

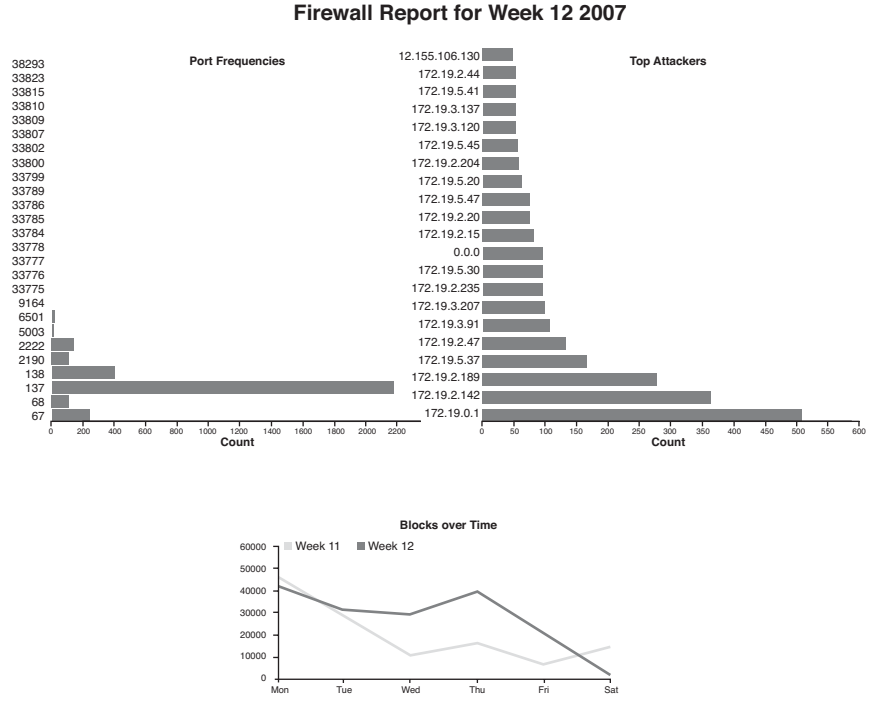


Figure 5-1 A sample report of traffic targeting my laptop. The charts show protocol distributions, the top attackers, and the number of blocked traffic incidents over the past seven days.

The goal of a report is to communicate information. The audience should be able to read the report and understand immediately what is shown. It should not be the case that additional information or explanations are needed to understand a report. Therefore, graphs that are complicated by nature are not well suited to reports. Simple graphs are preferable. That is why bar charts and line charts are great candidates for inclusion in reports. Sometimes scatter plots or time tables can be used, too. All the other graphs, such as link graphs, treemaps, parallel coordinates, and all three dimensional graphs, generally need more explanation or the capability for the user to interact with the graph to make it effective and useful. Bar charts and line charts are by far the most familiar graphs. Everybody has seen them used for many different data visualizations. There are, as always, exceptions to the rule. In addition to choosing the right graph to visualize your data, make sure that you apply the graph design principles with regard to size, color, shape, data-ink ratio,[1] and so on to make sure the graphs are easy to read.

REPORTING TOOLS

Tools to generate reports can be divided into three main categories. The first category consists of security reporting solutions, such as security information management (SIM) and log management tools. These solutions are capable of not just generating reports, but also taking care of all the processing to get the data into a report, such as collection, normalization, storage, and so forth. These tools focus on security events and generally ship with a set of predefined reports for specific reporting use-cases. Unfortunately, most of these tools are not cheap. Those SIM tools available as open source are limited in their capabilities and generally lack adequate support for data feeds.

The second category consists of general-purpose reporting solutions. Microsoft Excel and OpenOffice spreadsheets, Crystal Reports, Advizor, and gnuplot fall into this category. These tools do not deal with data collection. In addition, these types of tools are not built for security data and therefore might not offer some of the functionality necessary. For example, functions to format or process IP addresses are generally not available. However, these tools offer a great variety of graphic capabilities and are generally easy to use and operate. Other drawbacks that you might find annoying fairly quickly are that they operate on static data and that the generation of a new report cannot be automated.

[1] Edward Tufte talks about the data-ink ratio, which is defined as the amount of ink essential to communicate the information divided by the total amount of ink actually used in the chart. The "extra ink" is used to elaborate or decorate the graph but is not necessary for communicating the data information. See also Chapter 1, "Visualization."

The third category consists of programming libraries. There are dozens of such libraries, both commercially available and open source. Most libraries support the common programming languages, such as Java, PHP, and Perl. In Chapter 9, "Data Visualization Tools," I discuss some of the open source libraries that can be used to generate reports. One of the libraries I use fairly often is *ChartDirector*, which is available at www.advsofteng.com. The great benefit of libraries is that you can script the generation of reports and embed them into your own tools. This makes libraries the most flexible tool for report generation. You might pay for the flexibility because of the learning curve associated with working with the library and building the coding framework to use it.

ISSUES AND PROBLEMS

What are some of the problems or issues to watch out for when generating reports? One ubiquitous challenge is that too much data is available. It is important to filter and aggregate the data meaningfully and then apply the right graph type to represent the data. Doing so will help prevent cluttered graphs and make sure large amounts of data are dealt with efficiently. A point that I cannot stress enough about the entire world of visualization is that we have to keep the audience in mind with every decision we make. Who is going to look at the graph? A technically savvy person? A business person? If I generate a report for myself, I don't have to add much meta data for me to correctly interpret the graph. After all, I have been the one generating the report. I should know what is in it. However, if I were going to generate that same report for someone else, I would likely have to add some meta information so that the other person could understand the report.

REPORTING MACHINE ACCESS—AN EXAMPLE

Let's take a look at two sample charts that are frequently used to report machine access. Figure 5-2 shows a user login report where the number of logins is indicated for each user. The horizontal bars, rather than the normal vertical ones, helps keep the labels legible. Especially for long labels, this solution yields better results. The information for this type of graph can be collected from operating system or application logs. It would not hurt to even combine those two types of logs (operating system and application) into one single chart. The chart gives you insight into the behavior of the most active users, as well as the general user population accessing machines. Many things can be identified easily with this chart. For example, you might want to look out for direct **root** access. Good security practice is that root should never directly access a machine; instead, `sudo` or `su` should be used to execute commands that require root privileges. Also look for users who show an abnormally high login count. Pay attention in particular to the dark

portion of the bars encoding the failed logins; they should not be too long. You should be alarmed if the number of failed logins is almost 50 percent compared to the total number of logins, as is the case for a couple of users in Figure 5-2. Why would there be so many failed logins? Depending on your exact setup and the machines or applications for which you are collecting login information, you might want to look out for other things. It might even make sense to configure the graph to highlight some of those instances with a special color (for example, the root logins, as shown in Figure 5-2).

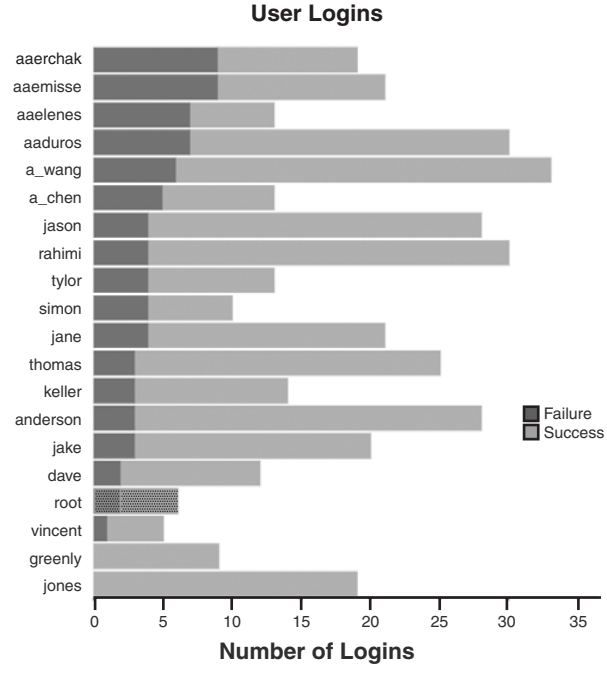


Figure 5-2 A sample report showing the number of logins per user. The chart also encodes whether the logins were successful and where they failed.

GENERATING BAR CHARTS

Bar charts are easy to generate with, for example, Microsoft Excel or a similar tool. You could also use ChartDirector (see Chapter 9) to write a script that can automate the process of generating a bar chart.

To generate a bar chart that shows the number of successful and failed logins per user (see Figure 5-2), we need to first collect the operating system logs that show login activity. On Linux, you find the logins in `/var/log/messages` or `/var/log/auth.log`. A successful login looks like this:

```
May 18 13:56:14 splunker sshd[4359]: Accepted publickey for rmarty from
76.191.199.139 port 33617 ssh2
```

A sample failed login looks like this:

```
May 18 21:00:02 raffy sshd[4484]: Failed password for
raffy from 127.0.0.1 port 50679 ssh2
```

You then use a regular expression to extract the user names from these records:

```
cat auth.log | perl -ne 'if(/sshd\[.*for.*from/) {
s/.*(Failed|Accepted).*for (?:invalid user )?(\w+)
from.*/\1,\2/; print;}'
```

The Perl command first looks for only SSH logins and then extracts whether the login succeeded or failed, along with the user name. The output of the preceding command looks like this:

```
Failed,cvsuser
Failed,cvsuser1
Failed,mana
Accepted,mysql
```

To generate a bar chart from this data, either load the data into Excel or write a Perl script that utilizes ChartDirector to do so. An example Perl script that shows how to use ChartDirector for a slightly different use-cases is shown later in this chapter.

Continuing with login information, in some cases you are not interested in the distribution of logins based on users, but you need to know the distribution per machine. This is shown in Figure 5-3. The chart encodes failed logins with black bars, and the bars

are sorted to show the most failed logins on top. The second bar from the top clearly sticks out. Why is the percentage of failed versus successful logins for this machine about 90 percent? This is probably a case worth investigating.

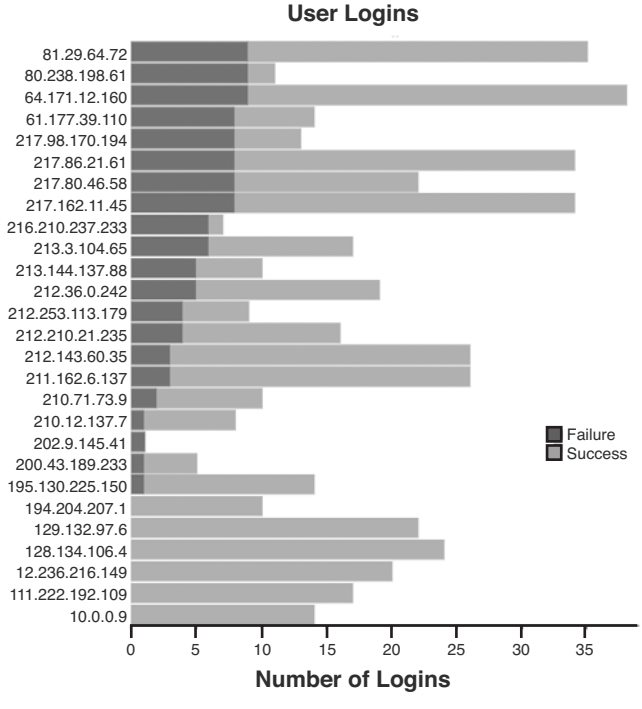


Figure 5-3 A sample report showing the number of logins to various machines. The chart uses color to encode whether the login succeeded or failed.

These reports are great tools to communicate among different teams. Security analysts frequently deal with log files and are comfortable with textual log data: syslog, packet captures, and so on. Using these logs to communicate with other teams, such as the operations team, is not very efficient. You will quickly realize that the operators are generally not keen on even trying to understand the log files you send them. Using graphical reports instead can work wonders. I know of an example where the security team used to hand log files to the operations people whenever a worm-infected machine was found, with little effect. The operators took their time trying to understand what the log files documented, and in a lot of cases returned them with a note that they were not able to find the problem. Upon the introduction of graphical reports, requests to clean worm-infected machines were handled in record time. People understood the reports. It is easy

to make claims in a graphical report that even management understands without further explanation. Figure 5-4 shows a sample graph that illustrates this scenario.

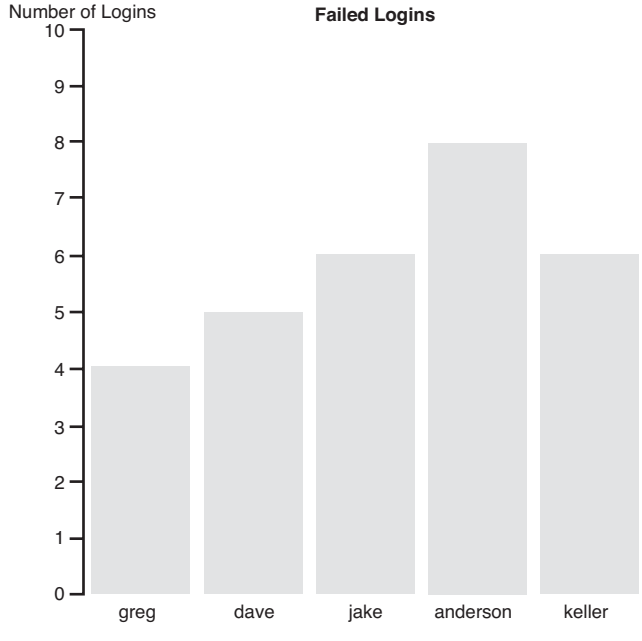


Figure 5-4 A sample graph showing the number of failed logins to various machines per user.

During the analysis of the last three charts, you might have felt an urge to compare the logins to an earlier state. How do the logins of this week compare to the ones of last week? This is where historical analysis, which I discuss in the next section, comes into play.

HISTORICAL ANALYSIS

Historical analysis can be categorized into four main areas: time-series visualization, correlation graphs, interactive analysis, and forensic analysis.

TIME-SERIES VISUALIZATION

In the preceding section, I mentioned that it is often useful to compare present data with past data. An entire branch of statistics is concerned with **time-series analysis,** which is

the analysis of data that was collected over time. For our purposes, it is not necessary to discuss those statistical principles. This section focuses on how visualization can help analyze time-series data.

What is time-series data? It is simply data that was collected over time. For example, if you are recording all logins to your systems and you not only record the username but also the time the login occurred, you have a time series. There are two goals in time-series analysis:

- Develop a model to predict future values.
- Gain an understanding of and analyze the recorded data. What are the variances in time? Does the data show any anomalies? Does the data show any trends? What is the underlying reason or root cause that generated the log entry?

Predictive analysis is always controversial. You have to make sure that you are in a closed system where you are aware of all external factors. Otherwise, external factors suddenly start showing up and may influence the data you are analyzing. This will skew your analysis, and the results will not make sense anymore. In log analysis, the use-cases surrounding predictive analysis are somewhat limited anyway. It is hard to come up with measures that you would want to predict. What good is it to predict the number of failed logins that will occur tomorrow?

To illustrate the influence of external factors in predictive analysis of computer security data, consider the case where you are tying to predict the next attack. You have data about past incidents. Maybe you even conducted a pretty good root-cause analysis and collected log files surrounding the incident. That is nice, but is this enough to actually predict the next attack? Did you really think of all the data points you have to consider, such as the machine's vulnerabilities, the possible attack paths, misconfigurations of the systems, and so forth. How do you take all these factors into account for your predictive model? I am not saying that you could not do it. I am saying that it is hard and you have to be extremely careful when doing so to not forget any of the important factors. I am not going to further discuss this problem, but instead focus on the other part of time-series analysis: understanding the recorded data.

I review five methods to analyze past data: **time tables, multiple-graph snapshots, trend lines, moving-average charts,** and **sector graphs.** These methods are not generally used for statistical time-series analysis. Most of the time, some other statistical method is used, but I discovered that these methods lend themselves quite well to analyzing logs files.

Time Tables

One of the graph types introduced in Chapter 3, "Visually Representing Data," was the time table. This type of graph is inherently meant for time-series visualization and lends itself nicely to analyzing and identifying three scenarios:

- Gaps in activities
- Periodicity of activities
- Temporal relationships

An example that shows all three—gaps, periodic behavior, and temporal relationships—is shown in Figure 5-5. The graph plots activity for a set of different target ports. The top series shows port 445 activity. Port 445 is used for all kinds of Microsoft Windows services, such as share access and Active Directory queries. Two clear gaps can be identified in the first half of the graph. Without knowing more about the dataset, it is hard to determine why there are such significant gaps. If this data was collected on a desktop, it could show that the desktop was idle for a while. Maybe the user stepped away for a break. If this was a server that should be under fairly constant load, these gaps might be a bad sign.

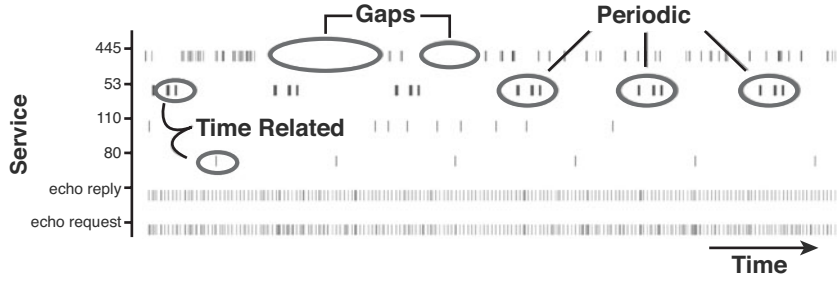


Figure 5-5 Timetable graph showing periodic behavior, as well as gaps in behavior, and time-related activity.

The next data series shows port 53, DNS-related traffic. This looks like a fairly interesting pattern. First thing to note is the periodicity. Six clusters repeat themselves. Internal to the clusters, there seem to be three groups. Note that the markers are fairly thick, representing more than just one event. What could this be? DNS is an interesting protocol. Normally, a client is configured with multiple DNS servers. If the first server on the list does not respond, the second is contacted; if the second one also fails to answer, the third in the list is being used. Only then will the DNS resolver return an error. Three servers do not always have to be configured. A lot of clients are configured with just one DNS server. The DNS traffic in Figure 5-5 could be representing such a scenario where the first two DNS servers fail to answer. This would also explain the multiple marks for each of the three groups. DNS tries three times for each server before giving up. Assuming this is the right interpretation, this also explains the temporal relationship with the port 80 traffic. After every DNS clusters, there is consistent activity on port 80.

This could indicate that the last DNS lookup was successful and thereafter a Web session was initiated.

I could not find an open source visualization tool that would generate a graph similar to the one in Figure 5-5. I therefore used a commercial tool, Advizor, which offers a graph called a timetable.

Multiple-Graph Snapshots

Probably the most straightforward approach to analyzing data over time is to take snapshots at different points in time and then compare them. With some graph types, it is even possible to combine data series of different time frames in a single graph. For example, with line charts, separate lines can be used for different aspects of the data. Figure 5-6 shows an example where each line represents a different server. In this example, there are three servers, and at regular intervals the number of blocked connections to those servers is counted.

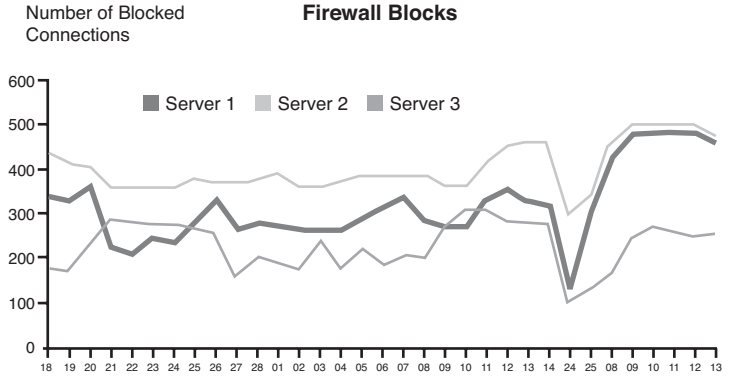


Figure 5-6 Comparing values over time can be done using multiple data series in a single chart; for example, a line chart can be used to do it, as shown in this figure.

When using multiple charts to compare data over time, make sure you are following these quite obvious principles. Figures 5-7 through 5-10 show, for each principle, how things look if the principle is not followed.

1. Compare the same data types: apples with apples. For example, do not try to compare different data, such as failed logins last week with successful logins this week (see Figure 5-7).

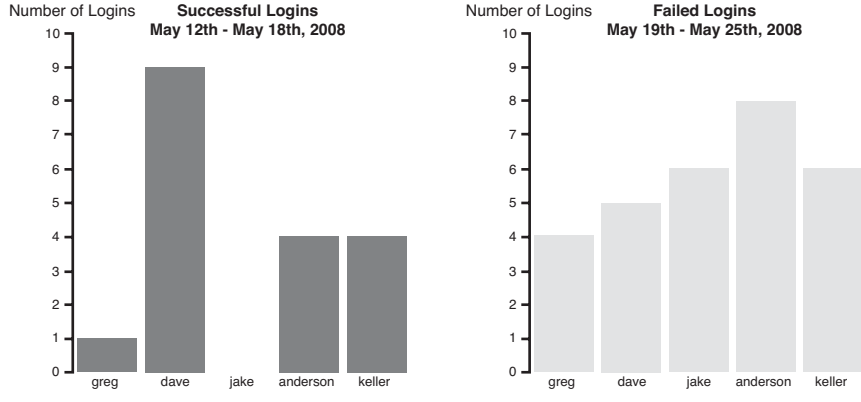


Figure 5-7 Compare the same data types: apples with apples. You would not compare failed logins with successful logins.

2. Compare the same exact values. For example, when monitoring logins, you should keep the same usernames on the graph, even if they have null values. Comparing disjoint sets of usernames is neither efficient nor very useful (see Figure 5-8).

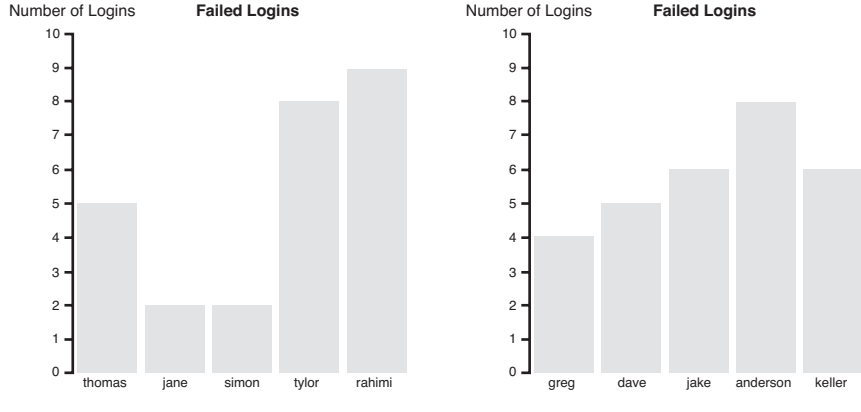


Figure 5-8 Compare the same exact values. You would not compare across disjoint sets of users.

3. Compare the values in the same way. Do not sort the charts by the values of the dependent variable, and especially do not do it for one of the charts and not the other(s). Use the same sort of value of the variable for each instance being compared (see Figure 5-9).

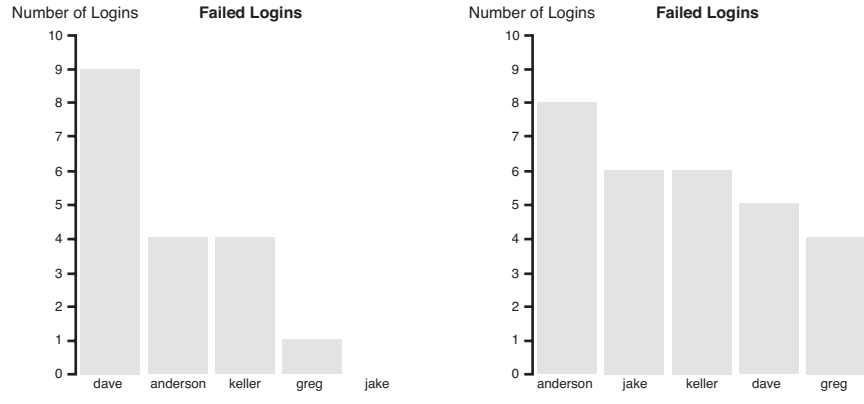


Figure 5-9 Compare the values in the same way. Use the same sorting. You would not sort by the values of the dependent variable in one instance and not the other.

4. Use the same scale on all the graphs. If one graph uses a scale from 1 to 100 and the other from 1 to 10, a bar filling up 100 percent means completely different things (see Figure 5-10).

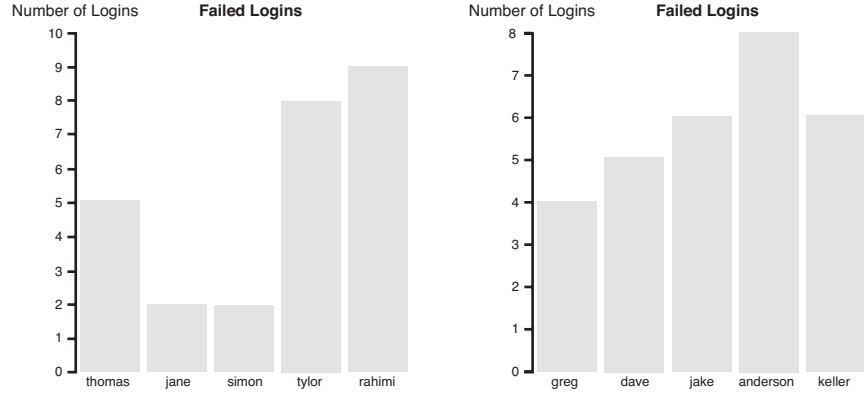


Figure 5-10 Use the same scale on all the graphs.

Figure 5-11 shows an example with three graphs showing user activity at three different points in time. Note how all the four principles are being used. All graphs compare successful logins over the same period of time, a week. They use the same usernames for each of the graphs, although some users failed to log in during specific weeks. The values of the variables appear in the same order, and the scale is kept the same.

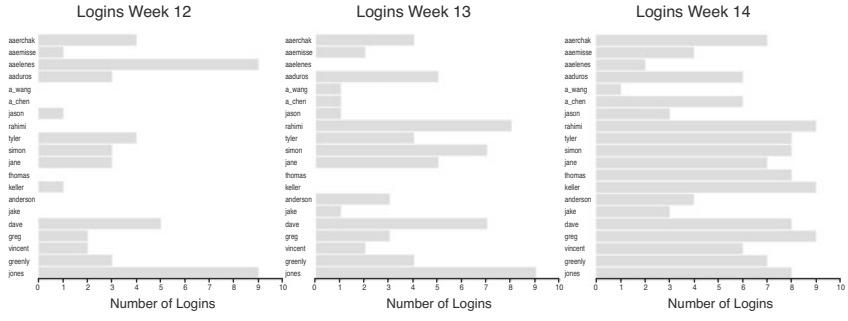


Figure 5-11 Three snapshots of successful logins at three different points in time. The four principles for point-in-time comparison are being followed.

As you can see in Figure 5-11, all the logins have increased over time, except for aaemisse. This could be a significant sign. On the other hand, this person might have been on vacation.

Figure 5-11 uses a bar chart to compare values over time. Some other charts are fairly well suited for this type of analysis too, whereas others are horrible candidates. Link graphs are probably the graphs least suited for analysis using snapshots over time. The problem with link graphs is that the layout significantly changes even if the underlying data is fairly similar. This results in graphs that look completely different even though the data might be almost the same. Some layout algorithms try to take care of this problem, but I am not aware of any tool that would leverage them.

Treemaps are tricky, too. To make them easy to compare with each other, you need to make sure that the data hierarchy is fairly stable. They are most valuable if the data hierarchy is staying completely stable and just the color is changed. With varying degrees of success, you can also try to change the size of the individual boxes, but it makes comparing multiple treemaps significantly harder.

What about scatter plots? Well, they are actually quite well suited for comparison with each other. The same is true for parallel coordinates. However, for scatter plots it is important to keep the axes the same; and in the case of parallel coordinates, it is important not to overload the graphs. In general, parallel coordinates are better suited for interactive analysis than static snapshots. In some cases, they work really well for static analysis, such as in cases where the dataset is fairly specific.

Trend Lines

Almost in the realm of predicting future values is the determination of a trend line for a data dimension. What is a trend line? A trend line indicates the general direction, or the

trend, the data takes over time. Figure 5-12 shows an example with three data series. Each series represents the same data but for different servers. For every day of a week, the number of attacks targeting each server is plotted, along with a trend line for each server. The attack trends for the different servers are all slightly different. Server 3 seems to be in pretty good shape. The attacks are generally in the low numbers, and the trend is decreasing. For the other two servers, it does not look as good. Server 1 shows an even worse trend than server 2. Server 2's trend is rising quickly. However, it is not rising as quickly as the trend for server 1. If I had to prioritize work, I would make sure server 1 is secure!

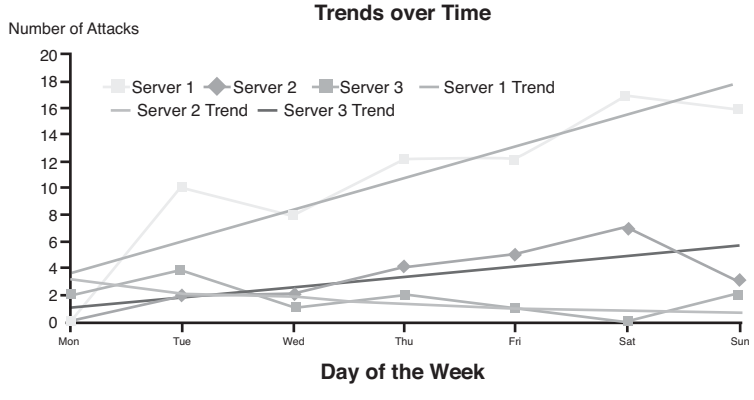


Figure 5-12 A line chart of activity over time. Three datasets are shown. Each refers to a different server that was targeted with attacks. Each of the datasets has its trend line plotted in the graph.

NOTE

Graphing software typically uses a statistical technique called regression analysis to find a linear trend line. Regression analysis finds a line through the data points that minimizes the average squared distance between each of the data points and their corresponding point on the line. This is also referred to as least squares regression.

You might be able to make a prediction, also called an extrapolation, of what the data will look in the future, based on a trend line. In Figure 5-12, you would extend the trend line to the right and see where it ends up for future points in time. The values you end

up with would quite certainly not be exact. It is likely that the predicted value would not be the same as the actual value. However, it represents a best guess or an educated guess as to what the actual value would be. There is also the possibility that the trend is going to change over time, depending on external factors, such as changes in usage patterns, firewall rules that change, and so on. Essentially, be careful when you are making future predictions. On the other hand, your prediction based off of the trend line is better than a "seat of the pants" prediction (that is, one that is not data based).

CREATING TREND GRAPHS

To generate a trend graph like the one in Figure 5-12, I use ChartDirector. I wrote a little tool that you can find in the AfterGlow distribution that plots line charts with corresponding trend lines. The input for the script needs to be in CSV format. The first column is the x-axis label, and the following columns each encode an individual data series. An example looks like this:

```
Mon,10,1
Tue,12,2
Wed,14,10
Thu,1,20
Fri,2,40
Sat,0,10
Sun,2,2
```

The weekdays are the labels on the x-axis, and the two following columns encode two data series that will be plotted. If you save this data in a file, you can then run the trendline.pl tool that uses ChartDirector to generate a trend graph:

```
cat data.csv | ./trendline.pl -t "The Title" -a
```

The `-t` switch is used to define a title for the graph, and the `-a` switch instructs the `trendline` script to draw a trend line.

Any graph type other than line charts is not well-suited for trend analysis. One of the dimensions needs to be time. The other data dimension can be used for one of two possibilities. The first possibility is any categorical variable in your log: target ports, users,

the originating network where a connection came from, or IDS signature name. Count the number of occurrences for a given time period and compare that value over time. The second possibility is to use a continuous variable or data dimension, such as the total number of bytes or packets transferred. Especially for network flow data, these are useful metrics.

A fairly interesting analysis that can be gained from a trend line is a feeling for how anomalous your events are. The distance between each of the data points to their trend is a measure of their anomaly. If you find a point that is very far away (also often referred to as **outlier**), you have found a significantly anomalous event that might be worth investigating. Be careful with this analysis, however. The data dimension you are investigating needs to have a relationship with time before you can claim any particular data points are anomalies. If your data points appear to be spread randomly, the data dimension under investigation is not likely to have any relationship to time.

You can also make use of a **confidence band** to summarize the size of the errors or distances between the individual points and their trend, as shown in Figure 5-13. If the value of interest falls within the confidence band, you agree to disregard the deviation from the baseline. If not, you can call it an outlier. This is just a visual tool to aid in detecting anomalous entries.

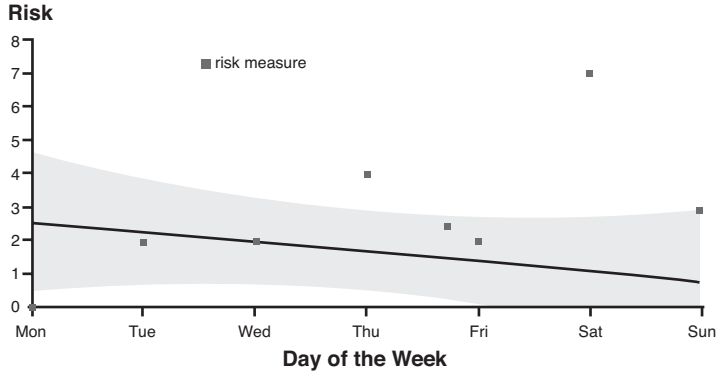


Figure 5-13 A trend line with a confidence band indicates the baseline that is used to plot new values against. If the new values leave the confidence band, they are labeled as anomalous.

GRAPHING CONFIDENCE BANDS

A confidence band summarizes the size of the errors between individual data points and their trend. A 95 percent confidence band, for example, implies a 95 percent chance that the true regression line fits within the confidence bands. I am using a ChartDirector script to graph data with a confidence band. Below is some sample code you can use to plot a line chart with a confidence band. The code assumes that you have a file with one data point per line. The data has to be sorted. Each data row is plotted one after another, assuming an increasing x-coordinate. Here is the code:

```
#!/usr/bin/perl
use perlchartdir;
use strict;

my @label; my @data;
# read input data.
my $index=0;
while (<>) {
  chomp;
  push @label,$index++;
  push @data,$_;
}
# prepare the chart
my $c = new XYChart(600, 300);
$c->setPlotArea(45, 45, 500, 200, 0xffffff, -1,
  0xffffff, $perlchartdir::Transparent,
  $perlchartdir::Transparent);
$c->xAxis()->setLabels(\@label);
$c->addScatterLayer(\@label,\@data);

my $lineLayer = $c->addTrendLayer(\@data, 0x444444);
$lineLayer->setLineWidth(1);
$lineLayer->addConfidenceBand(0.95, 0x80666666);
# generate the chart
$c->makeChart("confidenceband.png");
```

Trend Line Graphing Example

Let's walk through a simple example of how to generate a time series graph from iptables log files. I am interested in an analysis of all the blocked outgoing traffic. To do so, I will use a line graph for the last four days of blocked iptables traffic. The graph shows the traffic distributed over 24 hours and does so for each day as an individual data series. The result is shown in Figure 5-14. But let's start at the beginning by looking at an iptables log entry:

```
May 25 20:24:27 ram-laptop kernel: [ 2060.704000] BLOCK any out: IN= OUT=eth1
SRC=192.168.0.15 DST=85.176.211.186 LEN=135 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF
PROTO=UDP SPT=9384 DPT=11302 LEN=115 UID=1000
```

To generate the desired graph, we need the date and the hour from this entry. All other information we can disregard for now. To extract this information, I use the following command:

```
sed -e 's/^... \(..\) \(..\):.*/\1,\2/' iptables.log | uniq -c |
awk '{printf("%s,%s\n",$2,$1)}' | sort -r
```

The output looks something like this:

```
24,10,1484
24,11,2952
24,14,105
25,20,471
26,02,255
```

The first column is the date, the second one the hour of the day, and the third one indicates how many packets the firewall blocked during that hour. To graph this in a line chart, I use following Perl code that utilizes the ChartDirector library to draw the graph in Figure 5-14:

```
1   #!/usr/bin/perl
2   use perlchartdir;

3   # The labels for the x-axis, which is the hour of the day
4   my $labels = ["0" .. "24"];

5   # reading input
6   my $i=0;
```

```
7   while (<>) {
8     chomp;
9     # input needs to have three columns: Day,Hour of Day,Count
10    split/,/;
11    if ($current ne $_[0]) {$current=$_[0]; $i++;}
12    # @data is a day x hour matrix, which contains the count as
13    # the entry.
14    $data[$i-1][$_[1]]=$_[2];
15  }

16  # Generate a line chart and set all the properties
17  my $c = new XYChart(600, 300);
18  $c->setPlotArea(55, 45, 500, 200, 0xffffff, -1, 0xffffff,
      $perlchartdir::Transparent, $perlchartdir::Transparent);
19  # The x-axis labels, which are the hours of the day.
20  $c->xAxis()->setLabels($labels);
21  $c->addLegend(50, 30, 0, "arialbd.ttf", 9)
      ->setBackground($perlchartdir::Transparent);
22  my $layer = $c->addLineLayer2();

23  # Iterate through the days
24  for $i ( 1 .. $#data+1) {
25    $aref = $data[$i];
26    # Making sure no NULL values are present, otherwise
27    # Chartdirector is going to seg-fault
28    for $j ( 0 .. $#{$aref} ) {
29      if (!$data[$i][$j]) {$data[$i][$j]=0};
30    }
31    # Use a grayscale palette to color the graph.
32    my $color = $i * (0x100 / ($#data + 1));
33    $color=($color*0x10000+$color*0x100+$color);
34    # Add a new dataset for each day
35    $layer->addDataSet($aref, $color, "Day ".$i);
36  }

37  # Output the graph
38  $c->makeChart("firewall.png");
```

To run the script and generate the graph, save it as **firewall.pl** and execute it with `cat out.csv | ./firewall.pl`. The output is going to be an image called `firewall.png`. Make sure you install the ChartDirector libraries before you execute the script. The Perl code itself is not too difficult. It basically takes the CSV input, splits it into multiple columns (line 10), and creates a two-dimensional array (`@data`), which is later used for graphing. The code on lines 17 to 21 prepares the graph with axis labels and so forth.

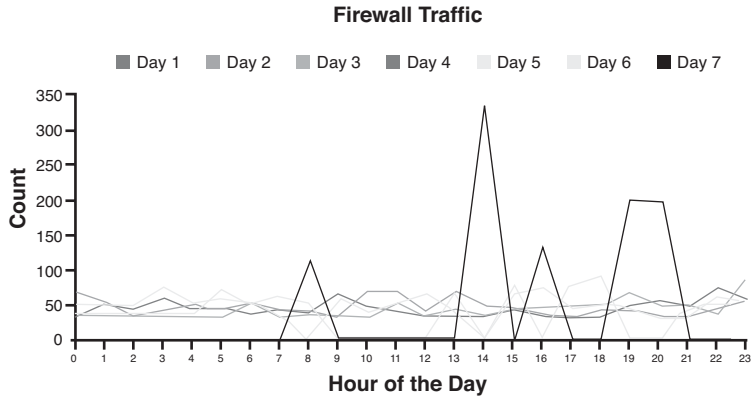


Figure 5-14 A sample report generated with the firewall.pl script, showing firewall events over 24 hours, split into individual series by day.

The final step is to go through each row of the data array, make sure there are no NULL values (lines 28 to 30), and then plot the value as a line in the graph. The color computation (lines 31 to 33) is somewhat fancy. I wanted to use grayscale colors for the graph. The two code lines for the color assignment make sure that each line gets a unique gray tone.

Figure 5-14 shows firewall activity for six consecutive days. The traffic is plotted over a 24-hour period. We can see that for most days, the traffic volume is fairly constant over the day. However, day 1 shows a completely different pattern. It shows spikes at 8 in the morning, at 2 p.m., at 4 p.m., and between 7 p.m. and 8 p.m. This seems a bit strange. Why would there be no traffic for certain times, and why are there huge spikes? Was there maybe some kind of infrastructure outage that would explain this phenomenon? This is worth investigating, especially because all the other days show regular behavior.

If you are interested in pursing these ideas some more with statistical methods, have a look at the next section, where I discus **moving averages.**

Moving-Average Charts

Trend lines are only one way to look at how your time-series data is evolving over time. Another method that is commonly used in stock price analysis is a **moving-average analysis.**[2] A moving average helps to smooth data values. Smoothing data values has the effect that individual outliers show up as less extreme. They are adjusted in terms of the

[2] More on moving average analysis of stock prices can be found at http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages.

rest of the data. It therefore makes it easier to spot trends. This is especially useful for volatile measures (that is, measures that change a lot over time).

How (and more important, why) would you look at moving averages? They are useful for analyzing trends of various measures and are an alternative to trend lines. Moving averages are more precise, and the analysis methods I show you here can prove useful in decision making based on time-series data.

As a decision maker, you need to know when exactly to make a decision based on a set of measures. You can try to look at a trend line, but the trend line is too generic. It does not react well to change. If you are the holder of a certain stock, you want to know when to sell. When monitoring attacks targeting your network, by either looking at firewall logs or intrusion detection logs, you need to know when the number of attacks starts deviating too much from the normal amount so that you know to start investigating and addressing a potential problem. I show you ways to make that decision.

MONITORING RISK TO DRIVE DECISIONS

One application of moving averages is to monitor risk associated with the number of unpatched machines on a network. Based on the risk readings, we can manage resource allocations to mitigate the exposures. Unfortunately, the budget for patching machines is limited, and we need to know when to make an investment in patching and when to take resources away from the patching efforts. We also want to use the risk measures to guide strategic investments in new security measures that can help contain the risk if a high watermark is crossed.
I am loosely defining risk in this context as follows:

> The **total risk,** due to unpatched machines, is the sum of risk over all machines. The risk of an individual machine is the sum of the exposure for each of the vulnerabilities of that machine multiplied by the business value of that machine.

This risk changes over time. It increases when new vulnerabilities are discovered, new machines and applications are introduced to the network that are not fully patched, and when reconfigurations happen. The risk decreases when patches are deployed, machines are removed from the network, operating systems are upgraded, and when machines are consolidated.

We can use this risk metric as a trigger for various actions:

- If the risk *increases,* we institute new countermeasures to reduce or mitigate it. Sample countermeasures are hiring new resources, training people, and purchasing new solutions.
- If the risk *decreases,* we can potentially reallocate resources away from patching machines.

We could use absolute risk thresholds to trigger these actions. Huge fluctuations in an observed metric trigger the thresholds constantly and are therefore a challenge to deal with. We need a more strategic trigger. If we add 20 new servers that are not completely patched, we would likely increase the risk significantly and trigger an action. However, a couple of days later, these machines will be patched, and we might trigger a low threshold. This continues constantly, and we are reacting to every change. This is where we are going to rely on moving averages to help us smooth some of the spikes.

Simple Moving Average

A moving average is computed by taking the average of data values over the last *n* values, where *n* defines the period for the moving average. For example, a 5-day moving average is computed by adding the values for the past 5 days and then dividing the total by 5. This is repeated for every data value. This process smoothes individual outliers and shows a trend in the data. The result of this procedure is illustrated by analyzing the risk associated with unpatched systems in a large network (see sidebar). A graph showing the risk of unpatched machines is shown in Figure 5-15. The figure shows the actual data values along with their moving average. You can see that moving averages are **lagging** indicators. They are always "behind" the actual data values.

By defining a high and a low threshold, we can determine when an activity has to be triggered. You can see in Figure 5-15 that for the moving average the spikes are smoothed and thresholds are less likely to be crossed as compared to the raw data, unless there is a real trend in the data. Crossover points of the data line and the moving average line mark a potential decision point. When the moving average crosses the data line, the data is significantly moving against the moving average, and therefore breaking away from the norm.

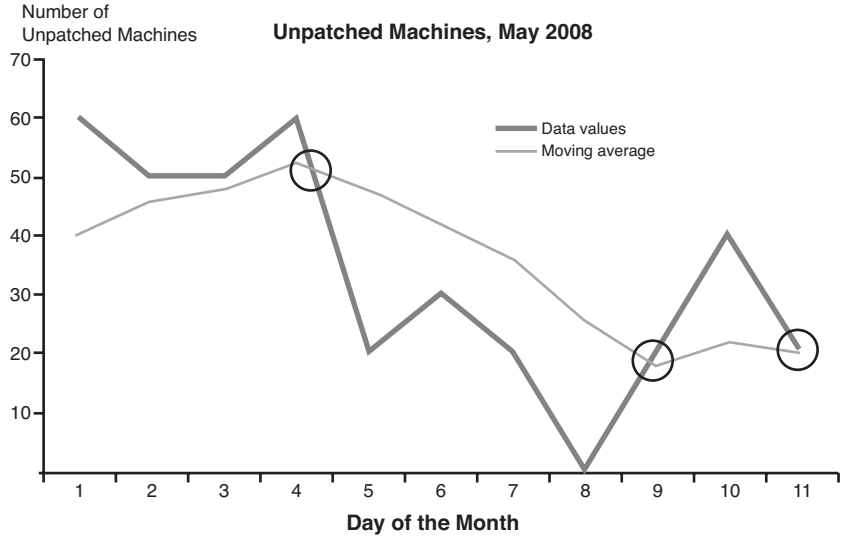


Figure 5-15 Example of applying a simple moving average to see the risk associated with unpatched machines. The three crossover points indicate a change in trend development.

Advanced Moving Averages

The issue with moving averages is that the lag is significant. Various methods help address this problem. For example, **exponential moving averages**[3] (EMA) are used to reduce lag by applying more weight to recent values relative to older values. The result of calculating an EMA on the data is shown in Figure 5-16.

Instead of EMAs, which address the problem of lag, we can also use a dual moving average analysis, where two moving averages of different time periods are used. The crossover points indicate upward trends when the shorter period moving average moves above the longer term moving average; it indicates a downward trend otherwise.

You might realize that this type of comparison is still fairly poor for the given data. There are three decision points. The second and third one seem to be not well placed. Just because there was one value that was higher on May 10 does not necessarily mean that things have changed for good. We need a better method than the simple moving average to reduce the number of decision points.

[3] For more information about exponential moving averages, have a look at http://en.wikipedia.org/wiki/Moving_average.

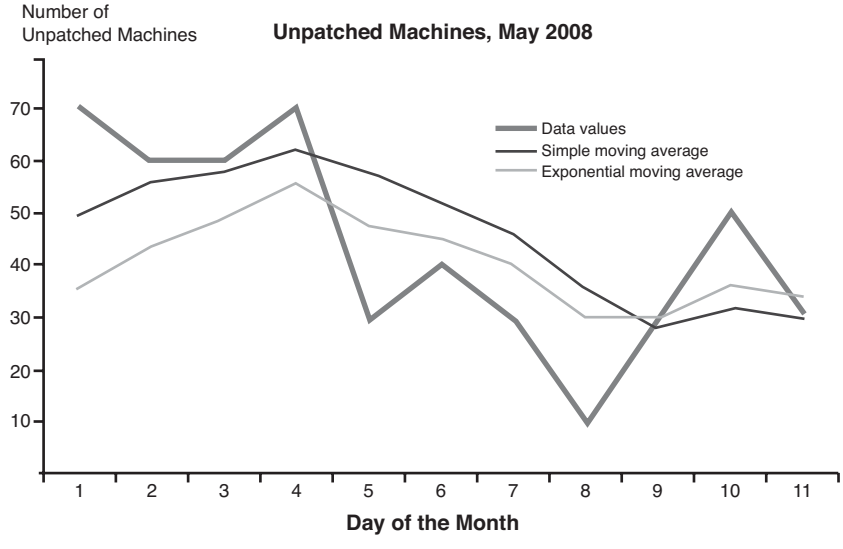


Figure 5-16 Example of an exponential moving average, compared to its simple moving average.

A more sophisticated analysis can be done by using a moving-average convergence/divergence analysis (MACD). It addresses some of the shortcomings of the simplistic methods I introduced earlier. It takes two moving averages, one over a longer period of time and one over a shorter period of time, and computes a measure based on the difference of the two. I borrowed this analysis from stock analysts.[4] A sample MACD chart is shown in The challenge is to come up with a good time period for the two moving averages. The period depends on your use-case and your data. The shorter the period, the quicker you are prompted to make a decision. The longer the period, the less reactive the analysis is to local spikes. To give current values more weight than older ones, an EMA can be used for both of the moving averages.

Figure 5-17 was generated with Excel, by calculating the individual values for each of the EMAs and then plotting them in the graph. I do not discuss the MACD analysis and the chart in Figure 5-17 any further. Some people say that the analysis is pure voodoo. It is difficult to define the right time periods for the EMAs. Changing them can significantly change the location of the decision points. Each application has its own optimal values that need to be determined.

[4] A great explanation of the MACD in stock analysis can be found at http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_average_conve.

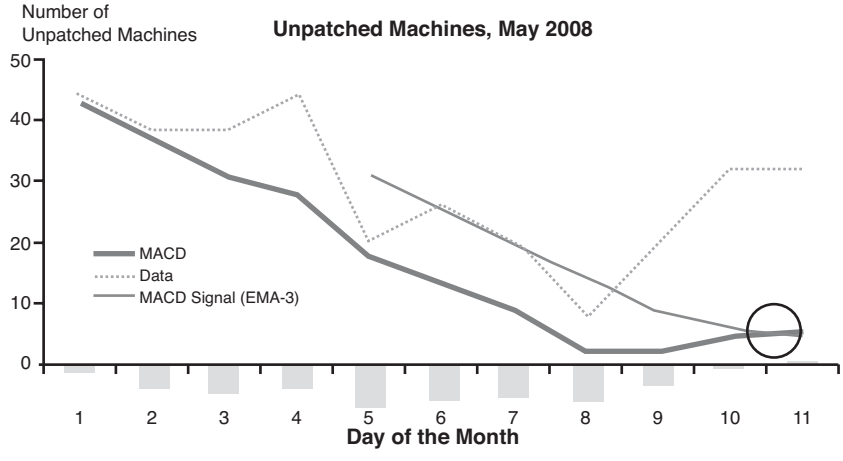


Figure 5-17 MACD chart with a 5- and 10-day EMA. The thick black line is the MACD line, computed by subtracting the 5-day EMA from the 10-day EMA. The 3-day EMA is plotted as the gray line, and the histogram at the bottom shows the difference between the MACD and the 5-day EMA. The histogram is positive when MACD is above its 3-day EMA and negative when MACD is below its 3-day EMA. A signal is generated when MACD signal crosses the MACD.

Applying Moving Averages

Here are some rough guidelines for when to use moving average analysis and when not to:

- Use moving average analysis to get information about trends and changes in trends.
- If you deal with data values that do not show a general trend—the data seems fairly chaotic—moving-average analysis is not very well suited.
- The moving-average time period is a parameter that needs to be carefully chosen. The smaller the time period, the more the moving average becomes reactionary. This means that you are reacting to the data more quickly and you are dealing with the problem of "false positives."
- Do not use this type of analysis for measures that can uniquely be associated with "good" or "bad." For example, a risk of 10 is "bad," and a risk of 1 is "good." You do not need moving averages to tell you when to react to the risk development. You can define a risk of 6 or higher as noteworthy.

  Or positively formulated:

- Use moving-average analysis for measures that do not have set boundaries. For example, the number of packets blocked by your firewall is a measure that has no boundaries. You do not know what a good or a bad value is. However, you want to know about significant changes.

Often, it is useful and necessary to capture the trend behavior at the present point in time. Whereas moving average charts show you the development of your metric over a longer period of time, you can use sector graphs to quickly capture the trend at a certain point in time, generally in the present.

Sector Graphs

An interesting way of analyzing the current state of a time-series dataset is by using a sector graph. The *New York Times* uses this type of graph to show the performance of stocks or markets.[5] The idea of the chart is simple. You take a time-series and fix a point in time that you want to analyze. Calculate the percentage change of the value from the point in time you chose to the value a day ago. Then do the same for the value you chose and its value a week ago. Assume you get a 5 percent change since the day before and a –10 percent change compared to a week ago. These two values now define a point in a coordinate system. Plot the point there. Repeat this for all the time series that you are interested in. By looking at the sector graph, you will get a comparison between all the series.

Instead of choosing a day and a week as the time periods, you can take any other time period that you are interested in. What is of importance is where a point for a time series is going to land in the coordinate system. If the point lands in the upper-right quadrant, for example, you are dealing with a series that has performed well over both periods of time. If the point lies in the bottom-right quadrant, you are dealing with a series that has done well over the short period, but not so good over the long period. It's an improving series. Analogous statements can be made for the lagging and slipping quadrants.

Instead of just drawing simple data points in the quadrants, you can use color and size to encode additional information about the time series. Color, for example, can be used to distinguish between the different series.

An example of how to use a sector chart is given in Chapter 7, "Compliance." There the chart is used to show the development of risk in different departments.

Figure 5-18 shows an example of a sector chart that was generated with Microsoft Excel. The exact steps of generating this graph can be found in Chapter 7. Figure 5-18 shows two data points. The values encode the number of incidents recorded for the Finance and the Engineering departments. The current number of incidents in the Engineering department is 25. The data point is located on the top right, which means that the incidents in the Engineering department have constantly been rising. This is a concern. The finance department shows a current number of 14 incidents. The data point lies in the bottom-left quadrant, which indicates that a constant decrease of incidents has been recorded. This is a good sign.

[5] www.nytimes.com/packages/html/business/20060402_SECTOR_GRAPHIC/index.html

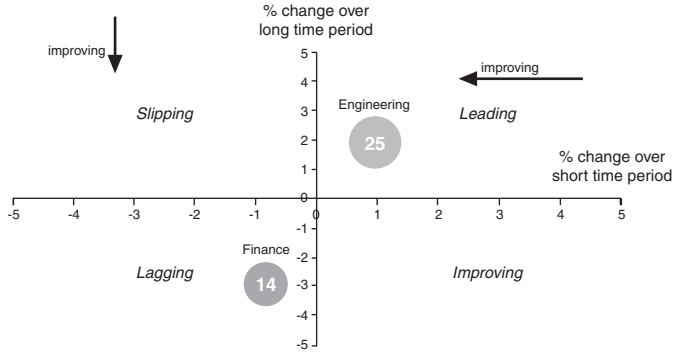


Figure 5-18 Sector chart with explanations of what it means when data points are drawn in the specific quadrants.

CORRELATION GRAPHS

Correlation graphs can be used, like time-series analysis, to analyze data by assessing the extent to which two continuous data dimensions are related. In other words, you want to know for values in one data dimension, do the values of the other data dimension correspond in some orderly fashion? There are two ways to use a correlation graph to analyze log data. Either two data dimensions of the same log file are correlated with each other or the same data dimensions are correlated for different log files. A correlation graph of two data dimensions of the same log entry is used to show how one dimension is related to another. For security log files, this is not very interesting. Different fields from the same log file are not correlated, unless it is already inherently obvious. For example, the event name and the target port are generally correlated. The target port determines the service that is accessed and therefore dictates the set of functionalities it offers. This set of functionalities is then generally expressed in the event name.

Correlation in this context works only with continuous or ordinal data. This already eliminates a lot of data dimensions such as IP addresses and port numbers. Although they could be considered continuous data, in most of the cases they should be treated as values of a nominal variable. There is no inherent ordering that would say that port 1521 is worth more or more important than port 80. It is just a coincidence that Oracle runs on a port that is higher than 80. So what are the data fields that make sense for correlation graphs? Well, they are very limited: Asset criticality is one, for example. This is generally an additional data field that is not contained in the log files, but here you can clearly make a statement about an order. What are some other fields? Traffic volumes, such as

bytes or packets transferred, event severities or priorities, and the file size are all continuous variables. That is pretty much it. Unfortunately. This also means that there is not much reason to actually use correlation graphs in this simple form for log correlation.

If we are expanding our narrow view a little bit and we shift our focus away from only log entries and their data dimensions, there are some interesting places where correlation graphs can be applied. What if we try to take aggregate information—for example, the total number of vulnerabilities found on a system during the past day—and correlate that number with the amount of money invested into each system for vulnerability remediation? These are not individual log entries anymore, but aggregated numbers for vulnerabilities and cost. Suddenly, correlation graphs are an interesting tool. Is the number of vulnerabilities directly correlated with the money we spend on vulnerability management? We hope it is negatively correlated, meaning that the number of vulnerabilities goes down if the money invested in vulnerability management is increased.

A somewhat more complex example is shown in Figure 5-19, where a **correlation matrix** is drawn. I took four data dimensions that were measured in regular intervals. The matrix shows the correlations between each of the four data dimension. The figure shows the individual correlation graphs, where the data points from two dimensions are presented in a scatter plot. Each of the graphs also contains their trend line and the correlation coefficient of the two dimensions. When looking at the trend line, you have to manually inspect the pattern of the data points. Do they run from the bottom-left corner to the top-right corner as in a positive correlation? Do they run from the top-left corner down toward the right-bottom corner as in a negative correlation? How close is each data point to the trend line itself? The closer to the line, the stronger the correlation. If they are randomly dispersed all over the place and do not group around the line, there is no correlation. This is the case in all the graphs in the first column of Figure 5-19. This means that the number of incidents is not related to any of the other data dimensions: Employee Hours, Personnel, or Cost. On the other hand, these three data dimensions are somewhat correlated with Employee Hours and Personnel, showing a strong correlation.

The **correlation coefficients** shown in each of the graphs are mathematical indices expressing the extent to which two data dimensions are linearly related.[6] The closer to 1 (or –1) the correlation coefficient is, the stronger the relationship.

How do we read the correlation matrix in Figure 5-19? You can clearly see a trend in the data points for Employee Hours and Personnel. They group nicely around the trend line. To a lesser extent this is true for Employee Hours and Cost and Personnel and Cost. Second, have a look at the correlation coefficients. They make a statement about whether

[6] You can find more information about the correlation coefficient at http://en.wikipedia.org/wiki/Correlation.

the data dimensions are linearly related. Again, you will find that our two data dimensions of Employee Hours and Personnel are showing a fairly high value, which means that they are strongly linearly related. If one increases, the other will, too. That just makes sense; the more personnel who respond to an incident, the more hours will be burned. It seems interesting that the cost is not more strongly related to the employee hours. There must be some other factor that heavily influences cost. It could be something like there is considerable variability in the payscale of the personnel. It also seems interesting that the number of incidents is not related to any of the other data dimensions. I would have expected that the more incidents, the more expensive it would be to address them—but perhaps once personnel are called to respond to one incident they stay around and address further incidents. It would take some investigating to nail down these other influences on the data.

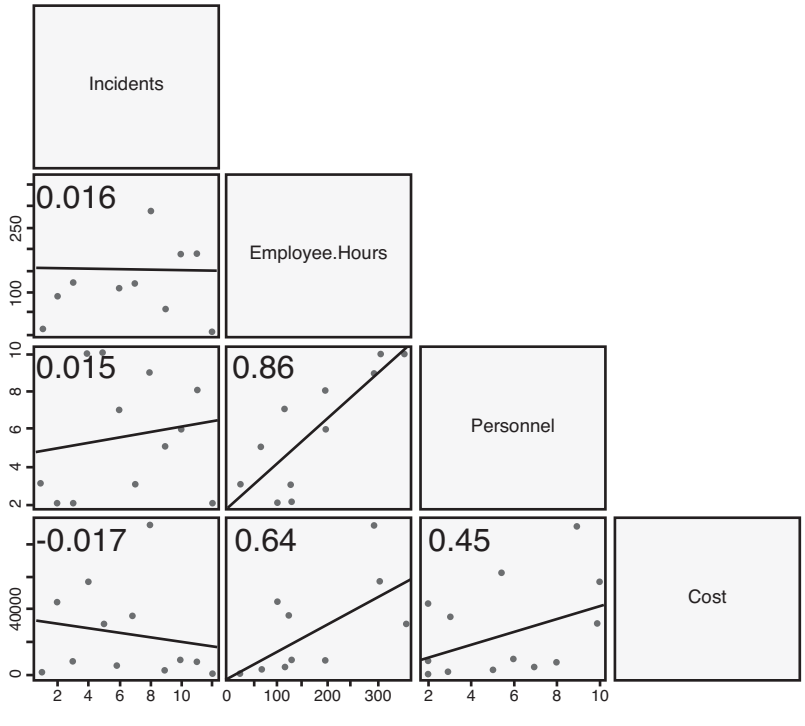


Figure 5-19 Correlation matrix showing the relationships among multiple data dimensions. It shows how the security investment (Cost), hours spent on security-related projects (Employee Hours), and the number of incidents, as well as personnel needed to clean up an incident, are related to each other. The number indicates the strength of the correlation between the two data fields.

GENERATING A CORRELATION GRAPH

To generate the correlation graph in Figure 5-19, you can used R (see Chapter 9). The following is a sample R script that reads data from a CSV file and then generates a correlation graph:

```
1  Dataset <- read.table("data.csv", header=TRUE,
   sep=",", na.strings="NA", dec=".", strip.white=TRUE)
2  panel.cor <- function(x, y, digits=2, prefix="", cex.cor) {
3    usr <- par("usr"); on.exit(par(usr))
4    par(usr = c(0, 1, 0, 1))
5    r <- abs(cor(x, y))
6    txt <- format(c(r, 0.123456789), digits=digits)[1]
7    txt <- paste(prefix, txt, sep="")
8    cex <- 1.5/strwidth(txt)
9    text(0.5, 0.5, txt, cex = cex * 0.4)
10 }
11 panel.myfitline <- function(x, y, digits=2, prefix="", cex.cor, ...) {
12   usr <- par("usr")
13   reg <- coef(lm(y [td] x))
14   abline(coef=reg,untf=F)
15   panel.smooth(x,y,col.smooth=0)
16 }
17 par(cex.axis=2)
18 par(pch=20)
20 pairs(Dataset, lower.panel=panel.myfitline, upper.panel=panel.cor, cex=2,
   cex.labels=2)
```

To run this script, you need a data file (data.csv) that contains a number of data series to be compared. Each column contains the values for a specific variable (for example, the cost). The script first reads the data in line 1. It then defines two functions (`panel.cor` in lines 2 to 10 and `panel.myfitline` in lines 11 to 16). The functions are used to generate the individual squares in the final output. The command in line 20 puts the pieces together and generates the correlation graph.

INTERACTIVE ANALYSIS

So far, we have used static images or graphs to represent data. Once the input data was prepared, we defined the graph properties, such as color, shape, and size, and used it to generate the graph. During the definition process, we generally do not know how the

graph will turn out. Is the color selection really the optimal one for the data at hand? Is there a better data dimension to represent size? Could we focus the graph on a smaller dataset to better represent the interesting parts of our data? What we are missing is a feedback loop that gives us the possibility to interactively change the graphs instead of backtracking to make different choices.

In the Introduction to this book, I mentioned the information seeking mantra: Overview first, zoom and filter, then details on-demand. I am going to extend this mantra to include an additional step:

1. Overview first.
2. Change graph attributes.
3. Zoom and filter.
4. Then details on-demand.

The second and third steps can be repeated in any order. Why the additional step? You could choose the graph properties before generating the first graph. However, this is one of the disadvantages of static graphs. You do not generally know how the graph will look before you have generated a first example. Looking at a first instance of a graph significantly helps to make a choice for the other graph attributes. It is also useful to change the graph attributes, such as color, on demand to highlight different portions of the data. After some of the attributes have been adapted and a better understanding of the data has been developed, a zoom and filter operation becomes much easier and effective.

The second and third steps of the new information seeking mantra are called **dynamic query** in the visualization world. A dynamic query continuously updates the data filtered from the database and visualizes it. It works instantly within a few milliseconds as users adjust sliders or select buttons to form simple queries or to find patterns or exceptions. Dynamic queries have some interesting properties:

- *Show data context:* How do data entries look that are similar to the result, but do not satisfy the query? Conventional queries only show the exact result, whereas dynamic queries can also display data that is similar to the result. This is often a useful thing to know to understand the data better.
- *Dynamic exploration:* Investigations, such as "what if" analysis, are intuitively possible.
- *Interactive exploration:* User-interface support, such as sliders, can be used to change the value of a variable interactively.
- *Attribute exploration:* The data of a single data dimension can be analyzed and explored interactively.

These aspects are all covered by dynamic queries. Keep in mind dynamic queries are a type of user interface. Behind the scenes, systems that support dynamic queries need a way to query the underlying data stores. This is often done through conventional query languages such as SQL.

Dynamic queries are unfortunately not supported by many tools. Most of the ones that exist are in the commercial space. Second, if you have used one of those tools, you know that the amount of data you can explore is fairly limited and is generally a factor of the amount of memory you have available. To support efficient dynamic queries, those tools need to load all the data into memory. Make sure that for large amounts of data you limit the scope of individual queries and work on a sample before you expand your view to the entire dataset.

A second interface concept that supports data exploration is the use of **linked views.** Each type of graph has its strengths when it comes to communicating data properties. You read about these properties in Chapter 3, "Visually Representing Data." To explore data, it is often useful to apply multiple different graphs to see various properties simultaneously. Using a display composed of multiple types of graphs can satisfy this need. To make this view even more useful, it should enable user interaction (i.e., support dynamic queries). The individual graphs need to be linked, such that a selection in one graph propagates to the other ones. This is an incredibly powerful tool for interactive data analysis.

The different types of graphs support different analysis use-cases. Bar charts, for example, are suited for attribute exploration. They are good filtering tools, too. **Attribute exploration** is a method used to analyze a single data dimension. What are the values the dimension assumes? How are the values distributed? Do some values show up more than others? Are there outliers and clusters of values? All these questions can be answered with a simple bar chart showing the frequency of each of the data values. Figure 5-20 shows an example with two linked bar charts, illustrating the concepts of attribute exploration and linked views. The bar chart on the left shows the count of users in the log file. Most activity was executed by the privoxy user. The rightmost side shows the ports used. Only two ports show up, www and https, indicating that we are dealing with Web connections. As the bar chart on the right shows, only about an eighth of connections were secured, meaning that they used HTTPS. On the rightmost bar chart, the secure connections (https) are selected. This selection propagates to the linked bar chart on the left side. We can now see that most secure connections were executed by the privoxy user, followed by ram and debian-tor. Root executed no secure connections at all. Why? Is this a problem? This simple example shows how a bar chart can be used for attribute exploration to show both secure and insecure connections by user.

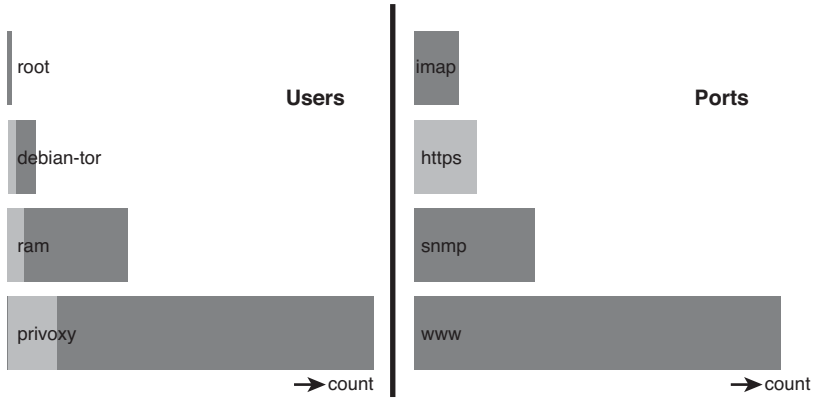


Figure 5-20 Linked bar charts, illustrating the concepts of attribute exploration and linked views. The left side shows the number of log records that contained each of the users. The right side shows the protocol associated with the users' activities. The https protocol bar on the right side is selected, and the selection is propagated to the linked bar chart on the left side, showing that most of the HTTPS connections were executed by privoxy, some by ram, and the rest by debian-tor.

What are some other charts and their role in interactive analysis and linked views? Scatter plots, for example, are a good tool to detect clusters of behavior. We have discussed this in depth already. Using scatter plots simultaneously with other linked views has a few distinct advantages. Interactivity adds the capability to detect clusters interactively and explore the data by selecting the values. This immediately reflects in the other graphs and shows what the clusters consist of. The other chart types, such as line charts, pie charts, parallel coordinates, and so on, can all be used in a similar fashion to represent data. I have already discussed the strengths and applications of all of these charts. All the benefits outlined for scatter plots apply to the other types of graphs, too. Linked views significantly improve the data-exploration process.

Not only selections can be propagated among graphs, but also the choice of color. Each of the graphs can use the same color encoding. This is yet another way that linked views are useful. Instead of using a separate graph to analyze a specific data dimension, that dimension can be used to define the color for the other graphs.

A great tool for interactive data analysis is the freely available **ggobi.** A detailed discussion of the tool appears in Chapter 9. Figure 5-21 shows a screen shot of ggobi, giving you an impression of how an interactive analysis looks. Note how the different graph types are used to highlight specific data properties. In addition to the graphs themselves,

color is used to encode an additional data dimension. It immediately communicates the values of the data dimension and shows how this dimension is related to all the others.

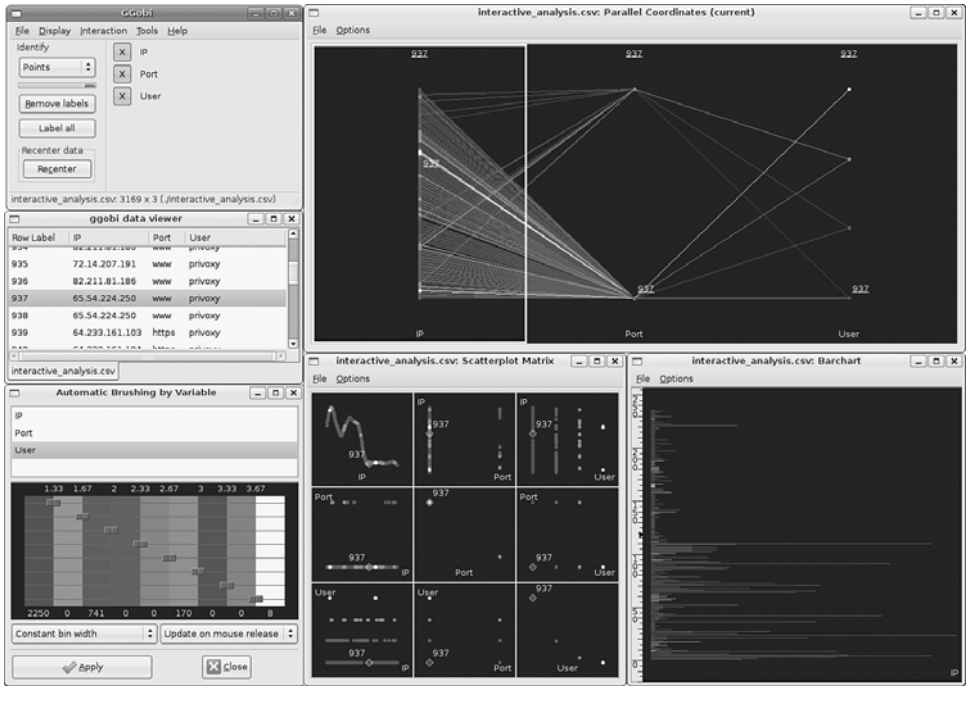


Figure 5-21 An interactive analysis executed with ggobi. Multiple views of the same data simplify the analysis by showing multiple data dimensions at the same time. Color is used to highlight specific entries, and brushing can be used to interact with the data.

The screen in Figure 5-21 is divided into two parts. The left side shows settings for the graphs and a window into the underlying data—the data viewer. The bottom part shows the configuration of the color bins. By moving the sliders, you can interactively change the color assignments for the three data variables that are visualized. The right side of the screen shows three different views into the data. By analyzing all three views, we get a feeling for all the data dimensions. We can see which machines are using which ports and which users are associated with that traffic through the parallel coordinates. We can identify how the data dimensions are correlated, if at all, by using the scatter plot matrix, and the bar chart can be used to see the distribution of the IP address values. Selecting data values in one graph propagates the selection through the other graphs. This supports the interactive analysis of the data.

FORENSIC ANALYSIS

All the concepts discussed in this chapter to this point were methods of analyzing data. How do we put all of these methods and concepts to work to tackle a real problem: the forensic analysis of a dataset unknown to the analyst? Forensic analysis can be split into three use-cases:

- Data exploration to find attacks, without knowing whether attacks are present
- Data exploration to uncover the extent and exact path of an attack
- Documentation of an incident

The second and third use-cases should be integral parts of the incident response (IR) process. Visualization not only helps speed up and facilitate the process of analyzing data, it is also a powerful tool for documenting an incident. I am not going into detail about how your IR process can be enhanced to use visual tools because IR processes differ slightly from company to company. However, given an understanding of the last two use-cases, it is a natural extension to include visualization in your own IR process.

Finding Attacks

Trying to uncover attacks through log analysis is not an easy task. This is especially true if there are no hints or you have no particular reason to be suspicious. It is much like trying to find the proverbial needle in a haystack. Visualization should play a key role in this detection process. We can learn from the visualization world about how to approach the problem. We have come across the **information seeking mantra** a couple of times already. We will see that it has its place in forensic log analysis, too. The first analysis step according to the information seeking mantra is to gain an overview. Before we can do anything with a log file, we have to understand the big picture. If the log is from a network that we already know, it is much easier, and we can probably skip the overview step of the analysis process. However, the more information we have about a log and its data, the better. We should try to find information about the contents of the log file from wherever we can. Information such as that which can be gathered from people who operate the networks we are about to analyze or system administrators responsible for the machines whose logs we have can help provide needed context. They can all help us interpret the logs much more easily and help us understand some of the oddities that we will run into during the analysis process.

All the principles discussed earlier around interactive analysis are useful to achieving an efficiently conducted forensic log analysis. Many questions about the log files can be easily answered with dynamic queries. For example, what services is machine A using? Nothing easier than that. Generate a linked view with two bar charts. The first bar chart

shows the source addresses, and the second one the target ports. Select machine A in the first chart and have a look at the linked selection in the target port bar chart. Find something interesting? Follow it and explore the data, one click after the other.

Unfortunately, there is no simple recipe for forensic log analysis that is independent of the type of log file to analyze. Each type of log file requires specific analysis steps. For certain types of logs, however, there are commonalities in the analysis process. I will introduce an analysis process that tries to exploit these commonalities.

The complete process for forensic log analysis is shown in Figure 5-22.

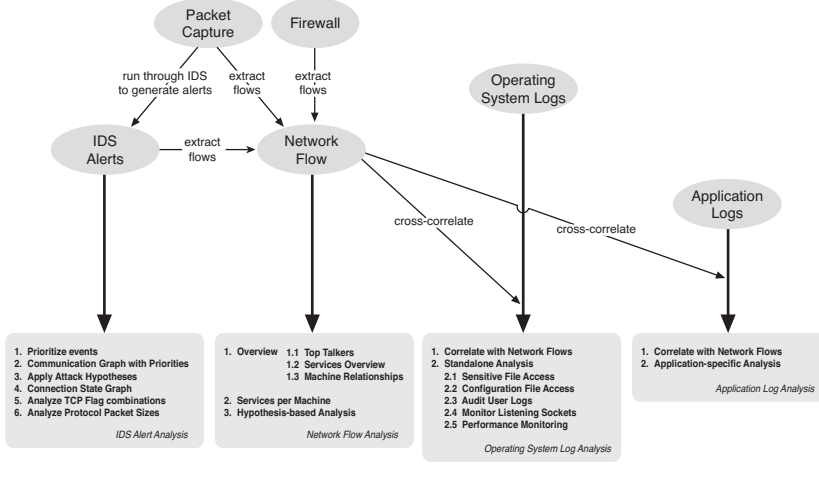


Figure 5-22 Forensic analysis process summary diagram. Ovals represent data sources, and the boxes contain the individual analysis processes.

The analysis process differs significantly depending on the type of log file that is analyzed. If it is a network-based log—anything from packet captures to network-based IDS logs—certain analysis steps apply. If a host-based log has to be analyzed, either on the operating system level or even on the application level, different analysis steps are necessary.

For the discussion of the forensic analysis process, I break the process up into different phases based on the diagram in Figure 5-22, and will therefore start with network flow data.

Network Flow Data

Network flow records are a great tool for gaining an overview of a forensic situation. Keep in mind that network flow data can be derived from other log types, such as packet captures, and in some cases firewall log files, and sometimes even from NIDS logs. I discussed this in depth in Chapter 2, "Data Sources." We first need to gain an initial understanding of the network traffic that we are analyzing. Here are the graphs we will generate:

1. Top Talkers (source and destination)
2. Top Services
3. Communication Graph

We start by generating some overview graphs to see the hosts and their roles in the network. Use a bar chart to show the frequency of connections seen for both source and destination addresses. Sort the charts by the connection frequencies to see the top "talkers" on the network. If you have data about machine's roles, use it as color in the chart. Make sure you are focusing on the most important roles so as to not overload the charts with color. Do the same for the destination ports. Use the machine's role as the colors again.

So far, the graphs do not reveal relationships between machines. You could use an interactive tool to explore the relationships by selecting different machines in the bar charts and simultaneously monitoring the change in the other charts, but there is a better solution. Use a link graph that displays the source and destination addresses in a communication graph.

You could use the amount of traffic to encode the thickness of the edges between the machines. You can measure the amount of traffic in either bytes, packets, or as number of passed packets, and so on. There are no limits to your creativity. If you use firewall logs to begin with, color the edges based on whether traffic was blocked or passed. If packets are passed and blocked, use yet another color.

Figure 5-23 shows all four graphs for a sample firewall log. I decided to show only the top 15 sources and destinations. Otherwise, the bar charts would be illegible. The same was done for the services. To match the data in the bar charts, the link graph shows only traffic between the top 15 addresses, too. The link graph colors machines that are managed by us in light gray. All other machines are dark gray. Also note that the edges (i.e., arrows) are colored based on whether the traffic was blocked. Dark edges indicate blocked traffic.

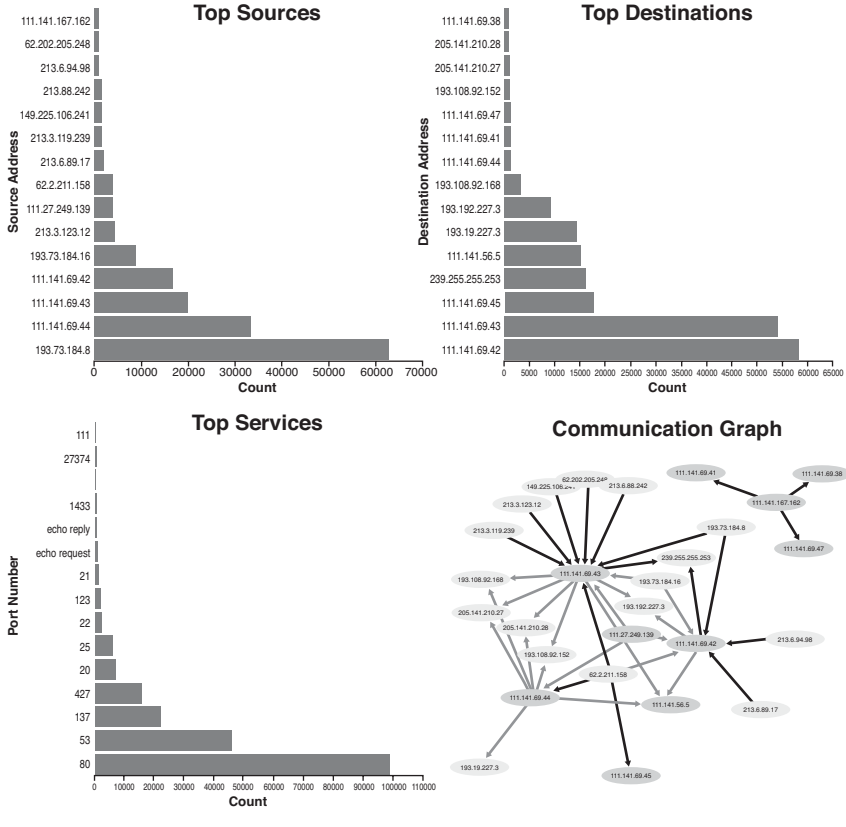


Figure 5-23 Gain an overview of the traffic in the log file. Who are the top talkers, what are the services accessed, and what is the relationship between these machines?

This gives us a first overview of what the log file is about. To explore in further detail how the graphs were generated, check the sidebar.

GENERATING A TRAFFIC OVERVIEW GRAPH

To generate the graphs in Figure 5-23, I used a pf log from my OpenBSD firewall. The first step in analyzing the pf log file is to convert it from pcap into some textual format. This can be done with tcpdump, which comes with OpenBSD.[7] Reading the log is done with

```
tcpdump -i pflog0 -nn > pf.log
```

Now you need AfterGlow (see Chapter 9), more specifically the pf parser shipped with AfterGlow. The script is located in `src/perl/parsers/pf2csv.pl`. Further, you need ChartDirector (again, see Chapter 9) and the `bar.pl` tool shipped with AfterGlow. It is located in `src/perl/charts/bar.pl`. That's it. When you have all these components, run the following command to generate a bar chart of the source addresses:

```
cat pf.log | pf2csv.pl "sip" | bar.pl -f source.png -n 15 -t "Source Addresses"
-p > sources.list
```

The command first extracts the source addresses (`sip`) from the log file. Then the bar chart is generated for the top 15 entries and saved as `source.png`. The `-p` switch is used to print the top 15 entries to the console. I am saving them for later use in a file called `sources.list`. You will see in a minute why. You repeat the same command for the destination addresses (`dip`). Don't forget to save the top 15 entries as `destinations.list`. Then do the same for services (`dport`). Almost done; we just need the link graph now. This gets interesting. The challenge is to graph only the top 15 hosts. How do we filter all the others out? AfterGlow does not provide an explicit capability to do this. However, by using a combination of the following command and properties file, we can achieve this:

```
cat pf.log | pf2csv.pl "sip dip action" | afterglow.pl -t -c graph.properties
```

Note the action in the list of fields to extract from the pf log. Why do we need this field if we graph only the sources and destinations? Remember, I colored the edges

[7] Do not try to read those files with other versions of tcpdump; you will not be able to read them. OpenBSD uses a proprietary, extended pcap format to store the pf logs.

between the hosts based on the action field—hence the field in the data. To show only the first two columns (sip and dip) in the link graph, we use the `-t` switch when running AfterGlow. By feeding it three columns, the third column can be used in the property file to control the color for the edges!

To show only the top 15 hosts, you have to create a property file like this:

```
variable=open(SRC,"sources.list"); @src=<SRC>
variable=open(DST,"destinations.list"); @dst=<DST>

color="darkgray" if (grep(/$fields[0]/,@src) &&
  (grep(/$fields[1]/,@dst)) && (field() = ~ /^111/))
color="gray" if (grep(/$fields[0]/,@src) &&
  (grep(/$fields[1]/,@dst)))
color="invisible"

color.edge="green" if ($fields[2] eq "pass")
color.edge="red"
```

I am using two commands to start with. First, I read the `sources.list` file into the array `@src`. The same is done for the destination addresses. The `variable` property is used to specify arbitrary Perl code that is executed when AfterGlow is started. After that, I color the machines on the list of top 15 sources (i.e., addresses on the `sources.list`), as well as the destinations on the `destination.list` in gray. The source addresses are accessible by using the `$fields[0]` field. I am defining two different shades of gray to distinguish between internal nodes and external nodes. The internal nodes I identify by looking at IP addresses that start with 111. All nodes that did not get a color assigned are going to be filtered out by giving them an `invisible` color. Finally, the edge color is defined by the third column, which is not plotted as a node, but we can still access its value. If the field's value is `pass`, we use a gray edge; otherwise, the edge is black.

For more information about AfterGlow, have a look at Chapter 9.

Instead of using pf logs, the same graphs could have been generated with traffic flows, for example, from Argus (see Chapter 2). Assuming your source is a pcap file, the commands are as follows. First you need to convert the pcap into an Argus file, which essentially extracts the relevant information that represents the network flows:

```
argus -r file.pcap -w file.argus
```

Now you can run the ra tool suite on the Argus file to extract the information you need. This is the command to generate the bar charts from the network flows:

```
ra -n -s saddr -r file.argus | bar.pl -f source.png -n 15 -t
"Source Addresses" -p > sources.list
```

You cannot use `rahosts` to do this because it does not output the number of times the host showed up. You might want to consider using `ragator` rather than `ra` to aggregate the flows and not just show raw flows. See Chapter 2 for more information about this.

To generate the link graph of communicating hosts, you use the following:

```
ragator -n -s saddr daddr -r file.argus --ip
| awk '{printf("%s,%s\n",$1,$2)}'
| afterglow.pl -t -c graph.properties
```

The use of `awk` converts the tab-delimited output into a CSV-delimited output that AfterGlow can understand. In addition, I filtered for only IP traffic, which gets rid of MAC addresses in the output. If you want them back, just remove that part in the command. In graph properties, you can take out the edge color assignment because there is no information about passed and blocked packets as in the firewall case. The rest of the property file can stay the same. Voilà, you have the same graphs, generated based on Argus data.

Using the graphs we generated, we should now try to figure out whether we have found any visible anomalies. Questions we can try to answer include the following:

- Is there a source or destination host that sticks out? Is there a host that generates the majority of traffic? Why is that? Is it a gateway that possibly even does Network Address Translation (NAT)? This would explain why it shows up so much.
- Is a certain service used a lot? Is that expected? If TFTP is the service used the most, for example, something is probably wrong.
- Are there services that were not expected? Is Telnet running on some systems instead of SSH?
- Are some machines communicating with each other that should not be?

Fairly quickly, after answering all these questions, you will want to know which services each machine is offering. To best visualize this information, use a treemap, as shown in Figure 5-24. Treemaps are well suited to encode a lot of information in a small area. They allow us to easily analyze the distribution of protocols in the network traffic.

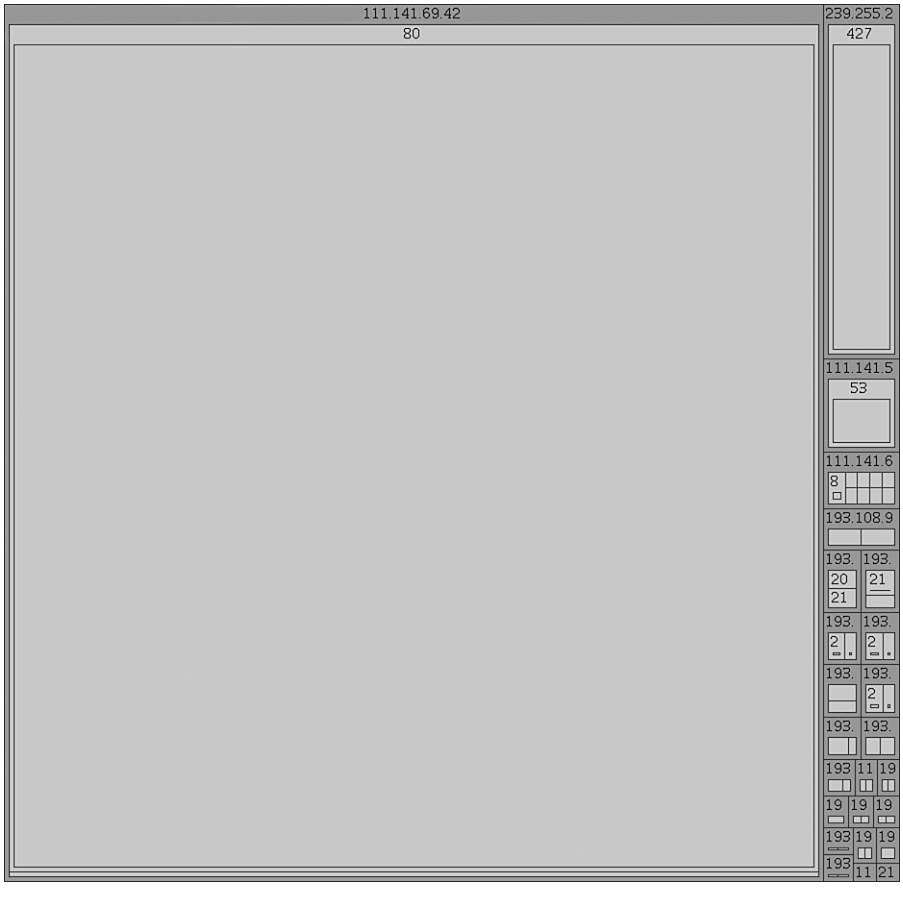


Figure 5-24 Treemap showing the machines on the network along with the services they are being accessed on.

We can easily see that there is one machine, .42, that gets most of the traffic on port 80. The other machines are so small that they seem to be hidden. Therefore, I generated a second graph of the same traffic, this time filtering out machine .42. Figure 5-25 shows the result. Now we can see all the machines and the ports they were accessed on. In the

case of visualizing a firewall log, we could use different colors to reflect whether the traffic was blocked or passed.

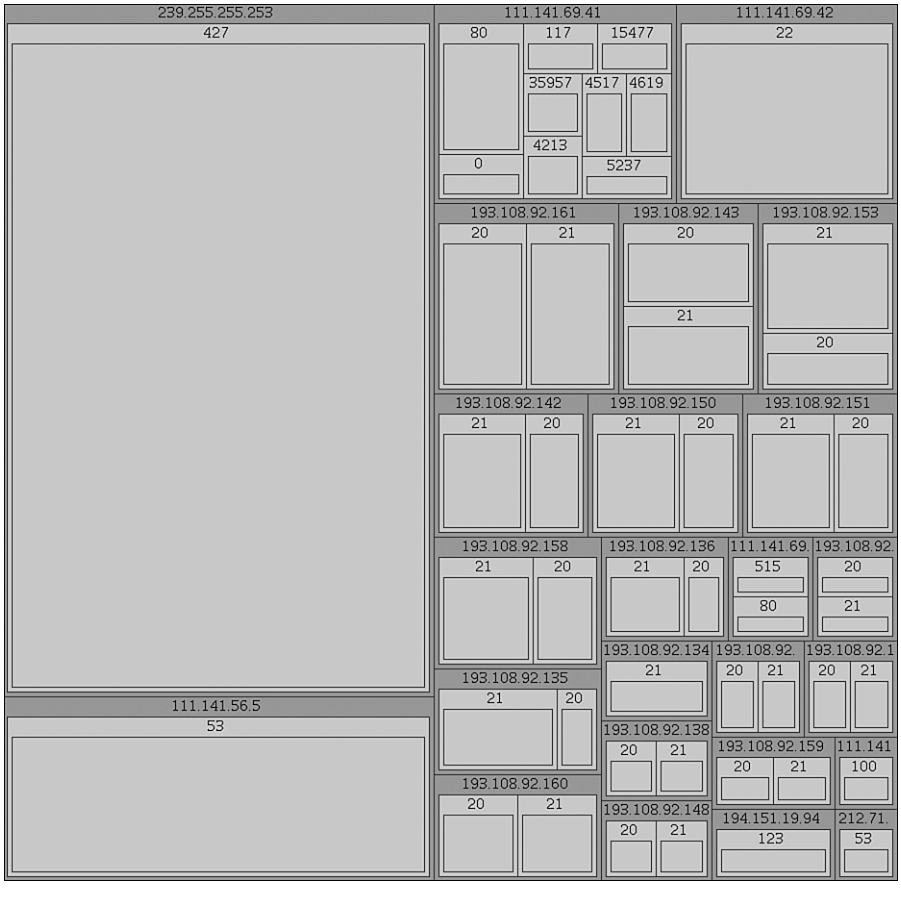


Figure 5-25 Treemap showing machines and their services. This time the dominant machine is filtered to show all other machines.

Figure 5-25 shows that a lot of machines were targeted with FTP. The next step is to verify whether these connections were indeed successful, and if so, whether those machines were meant to have FTP running? You can verify this by either looking at the raw packet captures or executing a port scan of those machines.

VISUALIZING NETWORK TRAFFIC IN A TREEMAP

Traffic logs can be used to generate a treemap like the one in Figure 5-25 that shows the machines and the services targeted. Here is how you take a log from a UNIX iptables firewall and prepare the data to be visualized in Treemap (see Chapter 9). We have to first extract the destination machine and the destination port from the log records and then format the output as a TM3 file. First, enter the header lines for the file and save this as `services.tm3`:

```
COUNT     Machine    Service
INTEGER   STRING     INTEGER
```

Then, extract the necessary fields from the iptables log and convert the output into tab-separated records. Then append the records to the previously generated file:

```
cat iptables.log | perl -pe 's/.*DST=([^ ]*).*
DPT=(\d+).*/\1,\2/g' | sort | uniq -c
| perl -pe 's/^\s*//, s/[, ]/    /g' >> services.tm3
```

Open this data in Treemap and visualize it with a hierarchy of Machine > Service, which generates a treemap similar to the one in Figure 5-25.

From here, the next step in analyzing network flow data is to define some **hypothesis** about the data and then verify the data against them. By thinking about your environment and what types of activities you could encounter, you can come up with a set of assumptions about what traffic might be interesting to look at. The hypothesis does not necessarily have to be true. By applying the hypothesis to the log file, it will help confirm whether the hypothesis was right or wrong. Consider this sample hypothesis: You are trying to uncover worm attacks by saying that a worm-infected machine will contact a lot of other machines and generate extensive communication patterns that will be readily visible in communication graphs.

Various other graphs can be used to analyze network flow data in more detail and find possible attacks. The generic analysis steps did not necessarily uncover attacks. Most of the times it is the application of a hypothesis that will help uncover anomalies and, possibly, attacks. In Chapter 6, "Perimeter Threat," I present some hypotheses that help, for example, uncover DoS attacks or worms. To be honest, detecting these two cases is not

rocket science because the volume of traffic involved in both of these attacks is quite significant. Again, other use-cases depend specifically on the data and use-cases that are of interest. Here I gave an introduction and a starting point for your quest. You can find more use-cases involving network flows in Chapter 6, where I show an example of how to monitor network usage policies to uncover unwanted behavior.

Table 5-1 shows the individual analysis steps that you can use to analyze network flow data. If you collected packet captures and not just network flows, you can use an additional step to analyze your data. Run your packet captures through an intrusion detection system to see whether it finds any attacks. To do so with Snort, run the following command:

```
snort -l /var/log/snort -c /etc/snort.conf -U -A full -r <pcap_file>
```

This command writes the Snort log file into `/var/log/snort/alert`. The additional IDS events generally uncover a significant amount of additional data. The next section shows how to deal with all this information.

Table 5-1 Summary of Network Flow Data Analysis Steps

Step	Details
1. Gain an overview.	Analyze: • Top talkers • Services overview • Machine relationships
2. Analyze overview graphs.	Can you find any anomalies in the previous graphs? Verify the top talkers, services, relationships, and so on.
3. What services are target machines offering?	Generate a treemap that shows the services per machine.
4. Verify services running on machines.	Are there any machines that should not be offering certain services? Analyze the previous graph, keeping your network configuration in mind. A DNS server should probably not expose a Web server, either.
5. Hypothesis-based analysis.	Come up with various hypotheses for analyzing your network flows.

Intrusion Detection Data

What can we do with network-based intrusion detection data? The difference from the data we have discussed before is that NIDS data shows only a subset of all connections,

namely those that violated some policy or triggered a signature. The only thing we can do to figure out which machines are present on the network and what their roles are is to treat the limited information in the IDS logs as network flow data. This is definitely not a complete picture, but it at least shows the machines that triggered IDS alerts in relationship to each other. However, we can do more interesting and important things with IDS logs.

IDS logs can, to a certain degree, be leveraged to prioritize and assess machines and connections. How hard has a target machine been hit? How "bad" is a source machine? How malicious is a connection? To do so, we need to define a **prioritization schema.** The higher the priority, the worse the event. I am assuming a scale of 0 to 10, where 10 is a highly critical event. As a starting point for calculating the priority of an event, we are using the priority assigned by the IDS, sometimes called the severity. We might have to normalize the numbers to be in the range from 0 to 10, which in some cases requires the conversion from categorical values, such as High, Medium, and Low to numeric values. Unfortunately there is no standard among IDSs to use the same ranges for assigning a priority to events. Some use scales from 0 to 100, others use categorical values. Based on this initial score, four external factors are applied to skew the score:

- *The **criticality** of the target machine:* This requires that every target machine is classified based on its criticality to the business. Machines that contain company confidential information should be rated higher than test machines.
- ***History** of the sources:* Keep a list of machines that were seen attacking or scanning your network. Machines that have scanned your network before will get a higher score. Ones that have attacked machines on your network will also get a higher score.
- ***Chance of success**:* Is the port indicated in the event open on the target machine? Often, IDSs report attacks that did not have a chance to succeed (for example, because the target port was not open). If the target port was not open, lower the priority score.
- ***Vulnerability** status:* Is the vulnerability that the attack was trying to exploit exposed and present on the target machine? If it was not present, lower the priority score.

To validate the last two points, you might need a vulnerability scan of the target machines. If those two conditions are false, you can drastically decrease the priority of your event. The attempted attack does not have the potential to significantly harm you. This type of event prioritization is something that security information management (SIM) solutions are using to calculate priorities for events and help get everyone focused on the important ones. You can also use other factors to rate an event. Depending on your environment, you might want to consider doing so.

Now that you have a priority for each event, you can first assign this to each event and then plot them atop your already existing communication graphs. The figure shows only priority 9 and 10 events so as to not overload the graph. In addition, for the priority 9 events, all the external nodes, machines that are not situated on our network, are aggregated to a single "External" node. For the priority 10 events, I show the exact address for the node. These configuration decisions are simply measures taken to prevent overloading of the graph and to keep it legible.

Figure 5-26 A network communication graph generated from network flow data. In addition to the network flows, IDS data is used to color the edges with the corresponding priorities.

The edges (i.e., arrows) are colored based on the priority. The darker the edge, the higher the priority. If a link was seen multiple times, the maximum of the individual priorities was used to choose a color for the edge.

If you are fortunate enough to have NIDS logs and network flows, you can execute numerous interesting analysis tasks. The following discussion assumes that you have both types of logs available. Some of the analysis also works if you have only NIDS logs.

What are the next steps to take after we have a graph that helps prioritize individual connections, such as the one in Figure 5-26? We should look at the graph and analyze it. What are the things we can identify that might hint at a problem or a potential attack? There are many things to look for, and some of them might require us to generate some additional graphs. For now, let's concentrate on the graph we have (Figure 5-26). We should start by defining some hypotheses about attacks that we can then look for in the graph. What are some of the things we would expect to see, and what are the questions we need to ask if there was indeed an attack hidden in the logs?

1. Based on the scoring of nodes and the exclusion of all low-priority connections, what are serious and important events? This is an important question to keep in mind. At this point, we should think hard before we dismiss a specific communication.
2. Do clusters of nodes behave in similar ways? Why is that? Do the targets in that cluster have any properties in common? Are there outliers in the cluster? Why?
3. If a machine gets successfully compromised, it might start to initiate sessions back to the attacker. Are there any such instances? This is where traffic flows can help provide the information about new sessions.
4. Do machines try to initiate sessions that never get established? This is common for scanning activity where ports are probed to see whether a service will answer back.
5. Do any connections show a strange combination of TCP flags? Any connection that does not comply with the RFCs, thus violating protocol specifications, is a candidate.
6. Do any connections have an anomalous byte count? This analysis should be done on a per protocol level. DNS over UDP, for example, should always have a certain packet size. HTTP requests are normally smaller than the replies. And so on.

This list is by no means complete. Many other hypotheses could be established and checked for. For now, the link graph in Figure 5-27 shows the same graph as in Figure 5-26, but this time it is annotated with all hypotheses that were true for this graph. The only hypotheses from the list that we can apply to this graph are the one identifying clusters of target machines, which are called out with a rectangle, and the one where we are looking for outgoing connections, possibly identifying infected or compromised machines. All those cases are called out with a small circle.

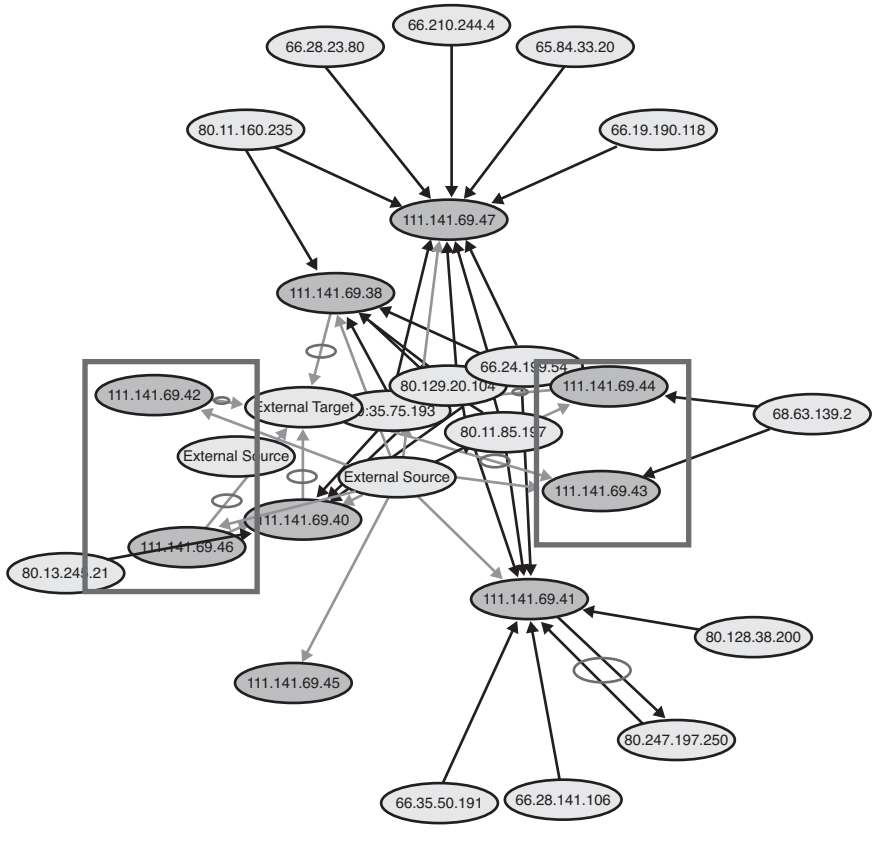


Figure 5-27 Attack analysis graph with callouts that mark the attack hypotheses.

There seem to be two clusters of two machines each that have similar behavior. These machines very likely have a similar role in the network. Based on the fact that the clusters are small, it will not be spectacularly interesting to analyze the clusters anymore. If they were bigger, we should figure out what the exact IDS events were that targeted these machines. It seems more interesting to investigate the connections going to the outside. There is one particular instance that is called out with a big oval that seems interesting. The IDS observed an attack of highest priority targeting an internal machine. In addition, the IDS picked up another high-priority event that went back to the attacking machine. This is strange. Why would the internal machine trigger another high-priority event backout? This is definitely an instance that should be investigated!

Some of the hypotheses in the preceding list call for a new graph that needs to be generated to answer those questions. We need to get some additional data about the sessions, which we can gain from extended network flows. Figure 5-28 shows a graph that includes connection status information. The color of the nodes represents the connection status based on Argus output. In addition, the size of the target nodes encodes the number of times a connection was seen between the two machines. The input used to generate the graph is to extract the source, the destination, and the connection status from the Argus logs. The following is the AfterGlow configuration used to generate the graph:

```
color="gray" if ($fields[2] eq "RST")    # reset
color="gray20" if ($fields[2] eq "TIM")  # timeout
color="gray30" if ($fields[2] eq "ACC")  # accepted
color="gray50" if ($fields[2] eq "REQ")  # requested
# connected, finished, initial, closed
color="white" if ($fields[2] =~ /(CON|FIN|INT|CLO)/)
color="gray50"
size.target=$targetCount{$targetName}
size=0.5
maxnodesize=1
```

With the graph in Figure 5-28, we can try to answer the questions 4 and 5 from the preceding list. Does the graph show any scanning activity? It seems like there are at least two clusters that look like scanners. One is situated in the middle, and one is on the left side of the graph They are both annotated as thick circles in the graph. Unfortunately, if we try to further analyze this, the graph is of limited use. I would like to know whether the connections that look like scanning activity were actually successful. Because multiple connections are overlaid in one node, however, the source node's color encodes the connection state of just one connection. It does not communicate the status for each of the connections, and the target nodes are too small to actually see the color on the nodes. To address this, we need to generate another graph and filter the original data to only show those nodes. The result is shown in Figure 5-29.

To confirm the hypothesis that the two nodes in question are really scanners, we have to look at the connection status. The upper-left machine shows a lot of connections in the INT or REQ state, meaning that only a connection request was seen and never an established connection. Very likely, this is some machine that is scanning. On the other hand, the machine on the lower right seems to have only established connections. Most likely this is not a case of a scanner.

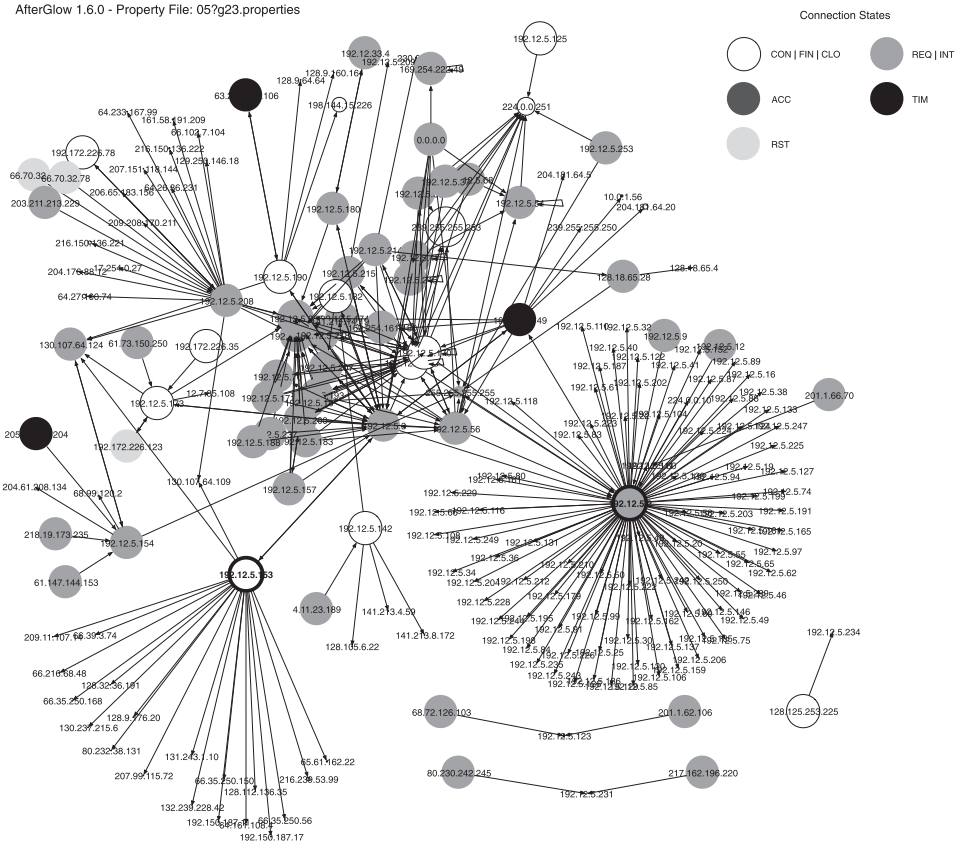


Figure 5-28 Attack analysis graph encoding the connection states and the number of times connections were seen between the machines.

The good news is that all the connections the scanner (upper-left machine) attempted were unsuccessful. The part that causes concern is that there are some machines that are attempting to connect to the scanner machine. We should verify whether those connections were successful to assess the impact. It is not quite clear, without knowing the role of this machine, why other machines are trying to contact it.

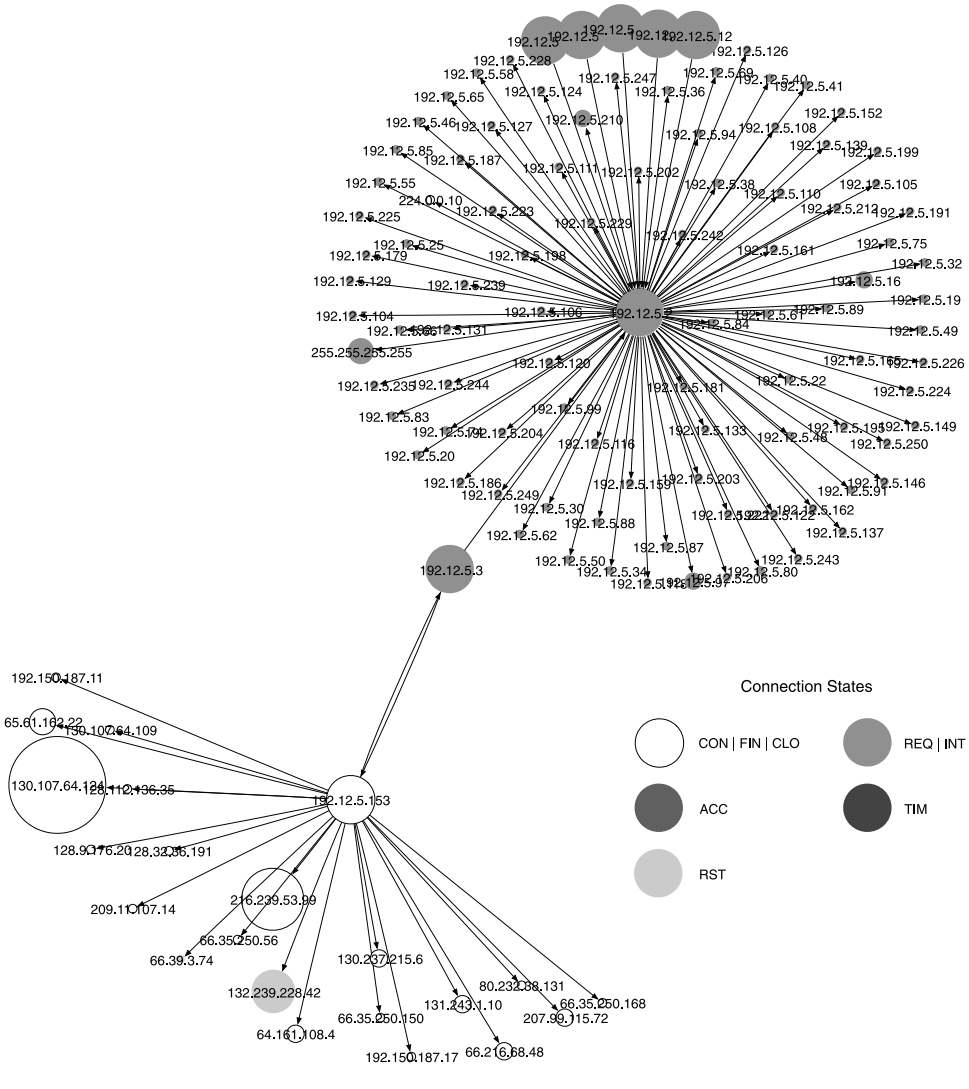


Figure 5-29 Zoom of the attack analysis graph, which only shows potential scanners.

What are some other things that we can identify in the original graph (refer to Figure 5-28)? The fifth hypothesis asked whether there were strange combinations of

TCP flags in a session. This requires generating a graph that shows all the TCP flags for a session. With Argus, this is fairly simple. The following command will accomplish this:

```
ragator -nn -s saddr daddr dport bytes status -Z b -A -r log.argus - ip
```

The `ragator` command merges matching flow records in the Argus log (log.argus) and outputs the result on the console. The `-Z b` switch instructs Argus to output the individual flags of a session. Here is a sample output listing:

```
192.12.5.173    130.107.64.124.53       31          189       CON
192.12.5.173    192.172.226.123.443     652         3100      FSRPA_FSPA
```

The first record indicates that a connection was established, indicated by the `CON` flag. The termination of the session was not yet seen in the time frame observed. The second entry shows the summary of multiple sessions of which some were terminated with regular FINs, while some of the connections were terminated with a RST, hence the R in the output. Both outputs are absolutely normal and will commonly show up. A way to visualize this type of information is to use the packet counts as sizes for the source and target nodes and use the status flags to drive the color. This is similar to what we have been doing in the previous graphs. I leave it to the reader as an exercise to generate this graph and analyze it.

The sixth hypothesis is about analyzing packet sizes for different protocols. Each protocol has characteristic packet sizes. A DNS request, for example, should always be fairly small. If you see large packets as DNS requests, something is wrong. To analyze packet sizes, I am going to use a box plot. Figure 5-30 shows how the sizes of both requests and responses are distributed for each destination port shown in the log file. Note that the x-axis is displayed using a logarithmic scale. This helps to display the packet sizes, because a lot of the packets are fairly small and a few of them are really large. The side-by-side display of request and response sizes enables the comparison of individual services. You will need quite a bit of experience to interpret this graph. What are the packet sizes for all the protocols used on your network? How do requests and responses compare? To analyze the graph, apply heuristics. For example, HTTP (port 80) should have fairly small request sizes. If you see a lot of large requests, it probably means that unwanted data is being transferred in HTTP requests. On the other hand, HTTP responses are probably fairly large. As shown in Figure 5-30, this is exactly the case.

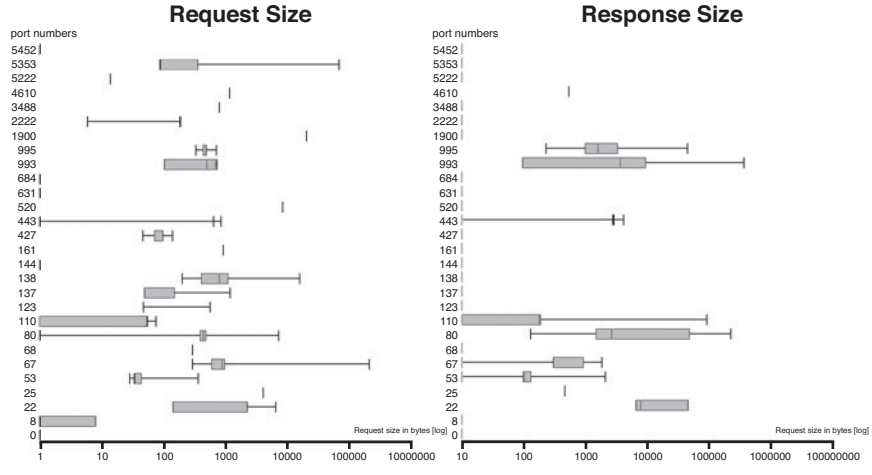


Figure 5-30 Box plot showing protocol size distribution for requests and responses of different protocols.

GENERATING A PACKET-SIZE BOX PLOT

The box plot in Figure 5-30 was generated by using ChartDirector and the `boxplot` script from AfterGlow.[8] The data to drive the plots stems from an Argus traffic capture. To graph that data, we first extract the correct fields from the data. To do so, use the following command:

```
ragator -nn -s dport bytes status -z -A -r cap.argus
- ip | awk -v OFS=, '{print $1,$2}' > out.csv
```

We need the service (`dport`), the bytes transferred (`bytes`), and the status of the connection (`status`) from the capture. We just need IP traffic, because all the other traffic does not offer services. To generate a box plot of this information, use the following command:

```
cat out.csv | boxplot.pl -f service.png -n 6000 -l
```

This command generates a box plot of all the port numbers that are smaller than 6000. It also makes use of a log scale for on the x-axis to accommodate for protocols that have large packets sizes. This way we can see the smaller packet sizes more clearly.

[8] http://afterglow.sf.net

Operating System Log

The analysis of operating system log files yields some interesting use-cases and possibilities to further provide the insights necessary to find attacks. Operating system logs can be used in two cases: either to correlate them with other log sources or on their own. The first case is to use them to provide additional intelligence for network-based logs. For example, if an IDS reports a DoS attack against a service on a machine, the operating system logs can be used to verify whether that service indeed terminated and the DoS attack was successful. The other use-case for operating system logs is to use them on their own. By looking for interesting entries in the OS logs, you can often identify attacks, too. However, finding attacks with only OS logs is not always easy. As discussed in Chapter 2, not that many types of events are recorded in OS logs, and attacks are not specifically identified in OS logs.

How can an OS log be correlated with a network-based log, such as network flows? There are many ways to do so. The general idea is always that you either confirm or deny some activity that was discovered in the network-based data (especially when dealing with IDS events) or the OS logs are used to complete the picture and give more context. Often, this reveals interesting new information that the network-based logs alone would not reveal.

Correlating OS with network-based logs raises the challenge of "gluing" the two logs together. This is done through the IP addresses in the network logs. We need to combine the OS logs from a machine with the corresponding network-based log entries mentioning that specific machine. Let's look at an example to illustrate this process. The case I discuss shows network-flow data with SSH connections targeting one of our machines. A sample entry looks like this:

```
05-25-04 11:27:34.854651  *   tcp    192.168.90.100.58841   ?>
192.4.181.64.ssh    149    172     14702      54360     CON
```

The flow shows that there was an SSH connection from 192.168.90.100 to 192.4.181.64. In the OS logs of the target machine (192.4.181.64), we can find a log entry generated by SSH, indicating a successful login:

```
May 25 11:27:35 ram-laptop sshd[16746]: Accepted password for root from
192.168.90.100 port 58841 ssh2
```

In addition, we have to make sure that the times of the log entries match. This is where it is important to have synchronized clocks! Figure 5-31 shows a graph where all network traces from SSH connections are shown. In addition, the SSH entries from the OS log are

included in the graph. The benefit of looking at both traces is that we get a more complete picture. The network traces show all the SSH activity from all the hosts. In addition to that information, the OS log provides information from an operating system perspective. The OS logs contain the user that logged in and not just from which machine the login originated. This information can be useful. The example in Figure 5-31 utilizes the OS logs to show which users accessed our server (192.4.181.64). The graph helps identify one machine, 192.168.90.100, which was used to log in to our server. The network traces help complete the picture to show all the SSH connections the originator machine attempted.

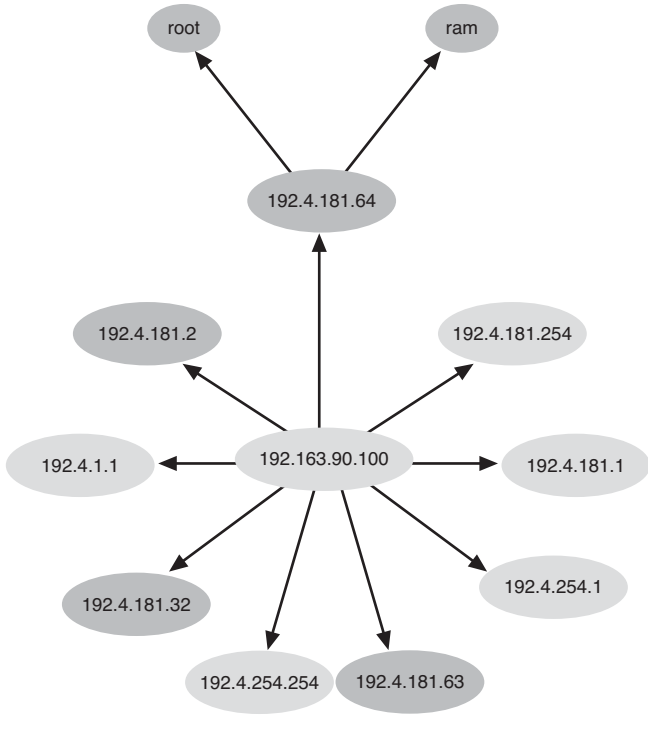


Figure 5-31 SSH activity, shown from both the network and the operating system perspective. This picture reveals that our critical server (192.4.181.64) is not the only machine this user accessed.

Why did the source machine in Figure 5-31 attempt to connect to all these other machines? And worse, the login to the critical server was successful. It seems that the user account was compromised and the person behind all this activity, the one controlling the originator, should be investigated.

OS logs do not necessarily have to be correlated with network traces. Even by themselves, they provide a lot of value. In some cases, these use-cases need special configurations of the operating system to enable the necessary level of logging to record these instances. The following are some use-cases that OS log files can be used with:

- *Monitor file access* of sensitive documents: To enable file auditing, have a look at Chapter 2. This can reveal users poking around in files that they have no reason to look at. Also, graph the number of files an individual user accesses, possibly even across multiple machines. This often reveals users who are snooping around.[9]
- *Monitor file access* of configuration files: Chapter 2 shows how to set up file auditing. Configure all configuration files to be audited. Make sure there is a justification for every configuration change on your machines. Ideally, there is a capability to correlate configuration changes with trouble tickets that document the change and show the proper authorization.
- *Audit user logins*: Look for logins per machine. Who are the users accessing machines? Graph the total number of logins per user. Do some users show suspicious numbers of logins?
- *Monitor listening sockets*: Every new open port showing up on a machine needs to be investigated. There needs to be a good reason why a server suddenly starts listening on a new port. Again, you will hope that a trouble ticket justifies the new service. If not, why does a machine suddenly offer a new service?
- *Monitor performance-related parameters*: By looking at CPU load, memory utilization, free disk space, and so on, you can not only detect performance degradations or machines that are starting to run at their limit, but performance related measures sometimes reveal interesting security-related issues. If a server is generally dormant during the night and suddenly shows a lot of activity, this might be a sign of an intrusion.

OS logs can prove useful in many more use-cases. This list serves merely as an inspiration. Visualization is powerful; through it, changes in behavior are easy to detect.

Application Logs

Going up the network stack, after the operating system we arrive at the application layer. There are many interesting use-cases where visualization is of great help in this space.

[9] The monitoring of file access would have revealed that Gary Min of Dupont was accessing thousands of documents, and an embarrassing insider crime case could have been prevented. See Chapter 8 for a discussion of this case.

The big difference from the previous discussion is that there are no generic use-cases. Instead, every application, depending on its logic, has its own unique analysis approaches. The following are some sample classes of applications that I briefly address to outline visualization use-cases:

- Network infrastructure services, such as DNS and DHCP
- Network services, such as proxy servers, Web servers, and email servers
- Applications, such as databases, financial applications, and customer relationship management software

It would be instructive to consider many more classes of applications, but I picked these classes of applications because they can be correlated with network flows. I cover more application-based use-cases later in this book. Fraud is an interesting topic in the realm of application log analysis. I discuss the topic of fraud in Chapter 8, "Insider Threat."

The classes of applications listed here can all be used to visualize and detect problems in the application itself. DNS, for example, can be used to find unauthorized zone transfers. Again, I do not discuss these application-specific use-cases here, but pick them up later, for example in Chapter 6. What I am interested in, for the moment, is how application logs can help with some of the analyses we have done earlier. For example, can DHCP logs be used to facilitate or improve the analysis of network-based logs?

We can use many network infrastructure services such as DNS and DHCP to improve our network data visualizations. How can we use DNS logs? Well, DNS is about mapping host names to IP addresses and vice versa. Why would we use that information for visualization? One could argue that to resolve IP addresses into host names, you could just do a DNS lookup at the time the graph is generated. That would certainly work. However, we have to be clear about what exactly we are doing. We are resolving an IP address at a different point in time, and more important, we are probably resolving it with a different DNS server than the one that was used in the original network where the logs were captured. Imagine a network where a private address space is used. The DNS server for that network will be able to resolve the private IP addresses to a host name. If you try to resolve those addresses with a different DNS server, the IP will not resolve to the same host name. This is the reason we should use DNS logs to improve our analysis.

DHCP logs represent a similar source of data. They give us a way to map IP addresses to MAC addresses, or actual physical machines. MAC addresses are globally unique and therefore identify a machine uniquely. This can be useful in an environment where DHCP is being used. At the time of visualization, the IP address showing in the graph

would most likely not be the same machine anymore as the one that was generating the activity. Ideally, you would also have an asset inventory at hand, which maps MAC addresses of machines to their owners. That way you have the capability to not just identify a specific machine responsible for a specific activity, but you could also identify a person responsible for the given machine. Applying all this data to a problem is fairly straightforward. The data can be used as a lookup table and is utilized to replace the IP addresses in the original logs.

How can we use network services, such as Web or mail servers, to aid in our analyses? Again, for this discussion, I am not interested in specific use-cases for these types of logs. Those are discussed in the Chapter 6. For the purposes of this analysis, I am interested in what additional information these logs can provide. Let's start with proxy logs. There are multiple types of proxies. Some are merely relays. The ones that I am interested in are proxies that require users to authenticate themselves. The logs from those proxies enable us to map IP addresses to users! This is incredibly interesting. Instead of identifying a machine that was causing mischief, we can now identify user names that are responsible for some activity, and we hope this will translate to actual humans. Note that this is not necessarily an easy task!

Do other network service logs result in similar benefits? The answer, as usual, is "it depends." All services that require a user login are potential candidates. We need to look for log entries that tie the user login to his or her IP address. ipop3d is a POP daemon that logs every session with a user name and client address from where the session was initiated:

```
Jun 12 09:32:03 linux2 ipop3d[31496]: Login user=bhatt host=PPP-
192.65.200.249.dialup.fake.net.in [192.65.200.249] nmsgs=0/0
```

If we extract the user and the user's IP address, we have an association of user to machine again. Other services provide similar log entries that help associate users with machines. With some logs, it is possible to associate a machine not just with a login but also with an email address. Fairly obvious, mail servers are candidates for this. One of them is Sendmail. Be careful with mail server logs. They are among the worst logs I have ever seen. Instead of logging on a session level, mail servers often log on an application logic level. For example, Sendmail logs a message as soon as the server gets an email that has to be delivered. At that point, Sendmail logs that it got a message from a certain email address. It does not yet log to whom the messages was addressed. Only after the email is ready to be delivered will it log that information. This makes it incredibly hard

for us. We have to manually stitch those messages together. Mail processors generate even worse logs, logging every individual step during the mail processing and always logging just a piece of the complete information that we would need for visualization.

The only class of information that we have not looked at yet is desktop applications. The challenge with desktop applications is that they do not allow logins over the network. Therefore, the log files do not contain IP addresses that we could use to correlate the information with network traffic. One possible use of application logs is to use them to gain more information about users and their roles. We will run into various problems trying to do that, however. Assuming that the user names are the same among applications and network services, we can try to look for events related to the same user's activities and glean information from the application log entries. Most likely, however, the flow is the other way around. You will have to use application logs as the base and augment the information with network layer information. This will enable you to correlate activity in applications with the origin of that activity. This can help to verify what other activities a certain user is involved in. Is someone executing a certain transaction on an application while at the same time using a network service? An example application is the supervision of financial traders. If they are placing a trade shortly after receiving a call via Skype or an instant message from a machine outside of the corporate network, it is possible that they got a tip from a third party.

The graph in Figure 5-32 shows what a simple, fake scenario could look like. The left part of the graph shows all instant messenger traffic. You can clearly see that the traders are communicating with each other. However, one machine seems to be an outlier. It looks like an instant message originated from the gateway address. This could indicate that an external message was received. This might be important to know. This by itself might indicate a policy violation. To see whether this is really something to investigate, we need the IP address to do an owner association. In addition, we want to see the trades themselves. Some example trades are shown in the figure. We see the user who posted the trade, the accounts involved, and the amount transferred. The rightmost graph in Figure 5-32 shows a graph where all this information is merged together. The IM traffic is plotted but with the nodes changed to the machine's owner rather than the IP addresses. In addition, the nodes are colored based on the transaction volume that each of the users traded. The darker the node, the more money was traded. We can see that the user who received an instant message from an external address was not the one posting the biggest trade. In addition to this graph, it would be interesting to see a timetable that shows the timing sequence of how the trades relate to the instant messages. I leave it to you to imagine what such a graph would look like.

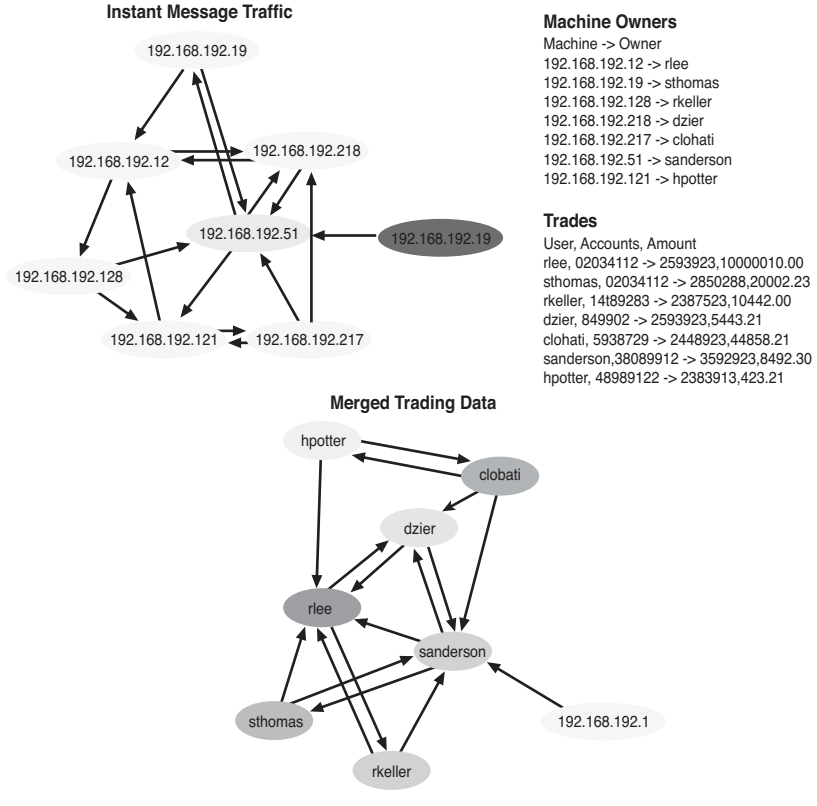


Figure 5-32 An example where an application log was correlated with network behavior based on the user names in the log files.

Additional Data Sources

A number of data sources are not covered in the process I just outlined. This does not mean that they are less useful. On the contrary, they could provide some significant insight into the behavior the logs recorded. Additional data comes not only from devices that generate real-time log files, but also from completely different sources, such as statically updated spreadsheets. Information such as the role of machines on the network is often managed in spreadsheets. Only a few networks that I have seen are actually documented in a configuration management database (CMDB) or in an asset management tool. This is unfortunate because it would make our analyses much easier if we had

access to up-to-date data from CMBDs. Other information is important, too, such as policies. Which machines should have access to which other machines? Which user roles have access to which machines? One common use-case is that you want to allow only users in the DBA (database administrator) role to use the DBA accounts to work on the database. You want to prevent every other user account from using this account. To do so, you need role information for the users. This information can often be found in a directory, such as LDAP or Active Directory.

How do you use this information in the analysis process? Ideally, the data sources are used as overlays to the existing graphs. In some cases, it will enhance the accuracy and the ease of analyzing log files by providing more context. In other cases, these additional data sources spawn an entire set of new applications and detection use-cases. We saw one application of additional data in the previous section on application logs, where I mapped IP addresses to their respective owners. Other use-cases are similar to this. What are some additional data sources that we should be looking at? Here is a short list:

- *Vulnerability scanners:* They can help with filtering out false positives from IDSs and factor into the priority calculation for the individual events. Make sure you are not just getting the vulnerabilities for machines but also the open ports and possibly some asset classification that has been used in the vulnerability management tool.
- *Asset criticality:* This information is often captured in spreadsheets. Sometimes the vulnerability management tool or an asset management database can provide this information. It is not always necessary to have a criticality for each machine on the network. Knowing which machines are the highly critical ones is typically sufficient.
- *User roles and usage policies:* User roles can be collected from identity management stores or possibly from logs that mention role changes. This information can be useful, especially in conjunction with policy modeling. If you can define usage policies, this will enable the monitoring of user activity with regard to respective roles. Things such as engineers accessing an HR server or salespeople accessing the source code repository become fairly easy to express and monitor. Policies are not restricted to IP addresses and who can access machines. They can extend to the application layer where the definition of acceptable behavior inside of applications becomes possible.
- *Asset owners:* Generally, this type of information is found in spreadsheets. Sometimes an asset management or a CMDB is available that stores this type of information. The information is useful for mapping IP addresses to machines and perhaps even to the owners responsible for those machines.

You can use this information in multiple ways for visualization. It could be used to replace values with ones that are looked up in these sources, as we have done in Figure 5-32, where we replaced the IP addresses with the respective owner of the

machines. Another way is to use them for color assignments or you could even use the owner and explicitly use it as an additional data dimension in the graph.

This concludes the discussion of the attack detection process. We have seen how we can forensically analyze log files and apply graphs to simplify the process. Unfortunately, the process does not guarantee the detection of attacks that have happened. It is merely a tool to help analyze the logs files for any suspicious signs and to uncover potential problems, or in some cases, attacks. The next section examines how to assess an attack. This is slightly different from what we have done so far. Instead of looking for the attack, the premise is that you know that there was an attack, possibly even knowing what some of the affected machines are or how the attack was executed.

Assessing an Attack

A fairly different use-case compared to detecting attacks in log files is the assessment of a successful attack. An attack can be detected in a number of ways, be that through the attack detection process or, for example, a customer who called in and reported an issue that could be as benign looking as a performance problem or a service doesn't work anymore. The assessment of the attack impact and extent is important to know what was affected and how big the loss is. It is also necessary to understand how the attacker was able to penetrate the systems and in turn how to prevent similar attacks in the future.

The following discussion applies mainly to cases where more complex attacks are executed. Typically, those attacks involve a network component. If the attack affects only one host and is executed locally on the machine, visualization of log files cannot help much in this scenario. However, if an attack is executed over the network and potentially involves multiple machines, there is a chance that visualization can help shed some light on the details of the attack, how pieces are related, and so on. I call this analysis **attack path analysis.** I would like to understand how an attacker was able to enter the network, what he touched, and so forth.

To start the process, we start with data gathering. As soon as the attack assessment process is started, we need to begin collecting pertinent log files. I hope that a variety of logs are available: network flows, intrusion detection data, firewall logs, host logs, and so on. We should extract a period of time preceding the attack. It might be enough to go back an hour. Depending on the type of attack, it is possible that not even a day is enough; instead, an entire year might have to be analyzed. When you start analyzing the attack, you will fairly quickly understand the time frame that you are interested in. Let's start with just an hour. Most likely, you will know what machine was attacked. Extract records just for this machine (or machines, if multiple ones were affected). Most likely, your problem at this point will be that you don't know what the source of the attack is. Therefore, finding the source of the attack is going to be the first analysis objective. How

do you do that? It might be close to impossible if the attacker executed the attack carefully enough. However, most likely the attacker made a mistake somewhere along the line. One of those mistakes could be that the attacker tried to access services on the target machine that are not available. Another possibility of detecting an attacker is the number of interactions he had with the target machine. Use a bar chart to show how many connections each of the sources opened to the target machine. Anything abnormal there? Use some other techniques of the attack analysis process to see whether you can find anything interesting that might reveal the attacker.

After you have identified potential attackers, use this candidate set to analyze all the activity seen by these machines. Most likely, a link graph will prove useful to show the relationships between all the attack candidates and the target machine. At this point, it might prove useful to extend the analysis window and take more data into account to see what the source machines have touched over a longer period of time. This will give you a good understanding of the extent of the attack. Which machines were affected and what services were used?

Especially with the information about the services the attackers used, you can verify the host logs to see whether you find any clues about how exactly the attackers entered the systems. The goal should be to gain a clear understanding of how the attack worked, who the attackers were, and which machines were involved. And more important than that, you should have a clear understanding of what data was affected!

In the next step, you can try to design a response to this attack. There are many possibilities:

- Blocking this type of traffic on a firewall between the attackers and the target machine
- Patching the vulnerabilities on the end system that the attackers exploited
- Introducing additional levels of authentication
- Deploying automatic response capabilities to block such attacks in real time

Visualizing the attack path is probably one of the most useful tools. It can help you not just analyze and understand the attack, but it also helps communicate the attack to other teams and eventually to management. Let me summarize the individual steps again that I went through to assess the attack:

1. Get records for the affected machine.
2. Find the source of the attack by looking for strange access patterns (e.g., connections to ports that are not open, excessive number of connections, strange behavioral patterns).

3. For the sources identified to be the potential attackers, analyze all the data referencing them. Find which machines they touched.
4. Deploy countermeasures to prevent similar attacks in the future.
5. Document the attack in detail (see the next section).

It would be interesting if you had the capability to execute all these steps in near real time. Doing so, you could prevent the attacks from happening. The part about responding to the attack and putting mitigation capabilities into place especially benefits from quick turnaround times. Commercial systems are available to help with these steps.

Documenting an Incident

The last part of forensic log visualization is the documentation of an incident. The attack detection process helped us find an attack and identify it. The second part was about assessing the impact and extent of an attack. When that information is known, we generally have to document the incident and provide this information to management, possibly law enforcement; and in a lot of cases, we can use the information gathered to educate people in our organization. Only part of incident documentation can be done with log files. Often, forensic images will be taken from machines that will serve as evidence and documentation of an incident. However, the part that can be done through log files is often useful to help communicate how an attacker entered the systems and the extent of the problem.

I have already discussed a lot of the elements of incident documentation when we talked about reporting. The important things to keep in mind are two things:

- Who is the audience for the incident documentation?
- What is the best way to represent the information?

These two questions will help you make sure that the documentation meets its goals. If you are writing the documentation for your management, make sure that you show on a higher level how the attack happened. Why was the attacker able to get in? Don't go into all the gory details of the vulnerabilities that were present and the exploit code that was used. Show concepts and where more controls could have prevented the attack. If you are communicating the information to the owner of the servers that were penetrated, mention all those gory details. Help them understand how the attacker penetrated the system, but don't forget to mention how the attack could have been prevented. The system administrators will not be too interested in the network components of the attacks but instead will want to know what happened on the server itself.

It is not always the best idea to use graphs and visualization for this type of documentation. Some of the documentation is better communicated in textual form. However, if you are trying to show an attack path, how the attacker actually entered the system, a picture is still worth a lot. Show the network topology, and on top of it, how the attacker was able to come in. Link graphs are generally a great tool for this type of documentation. I could spend a lot more time on this topic, but it is not one that benefits tremendously from visualization. In sum, if you can summarize information in a graph to communicate the pertinent information with other people, do so!

REAL-TIME MONITORING AND ANALYSIS

Thus far, our focus was on forensic or historical analysis of security data. Let's shift our focus slightly and take a look at how data on systems, and applications, can be monitored in real time (or near real time, to be accurate). The focus for real-time monitoring is to understand the current state of systems and applications, and presently ongoing tasks or events of interest. These events will obviously directly impact the current state and change it consequently. To communicate these properties, the current state, as well as current events, we use **dashboards.**

DASHBOARDS

You have definitely seen and possibly used a dashboard before. Stephen Few in his book on information dashboard design[10] defines a dashboard as follows:

> A dashboard is a visual display of information needed to achieve one or more objectives which fits entirely on a single computer screen so it can be monitored at a glance.

This is a lot crammed into one sentence. However, all the individual components are important. I show you a little later how to put all the components in the definition to work. One of the key points that is not explicitly mentioned in the definition, but is implicitly covered under the topic of an objective, is the focus on building a dashboard for a specific audience. In other words, a dashboard should always have the exact use-cases and people looking at it in mind. In computer security, three main groups or types of dashboards are needed:

[10] Stephen Few. Information Dashboard Design (O'Reilly, 2006) p. 34.

- *Operational:* A dashboard used to track core processes, metrics, and status. It communicates low-level information at a glance and is used for real-time monitoring. The audience is security analysts in need of accurate real-time information.
- *Tactical:* Used for tracking processes of departments, networks, states of machines, and so forth. It helps analyze exception conditions and summarizes data to analyze the root case of a problem. Managers of security operations or supervisors are often the audience for this dashboard.
- *Strategic:* A dashboard that helps monitor the execution of strategic objectives. Trend visualizations are commonly found in these dashboards. The need for real-time information is relaxed. These dashboards are used to improve coordination and collaboration, and the audience generally includes the CISO, CSO, and other executives.

No matter what type of dashboard, one of the most common and important concepts is the concept of **comparison.** It is the capability to see trends and changes over time that should attract the attention of the viewer. This does not mean that only changes should be shown in dashboards. Often, it is the fact that some measure has not changed that makes it noteworthy. Table 5-2 shows some examples of comparative measures. Also shown in the table are examples of how these measures can be used and what type of dashboard would generally employ this measure.

Table 5-2 Dashboard Comparative Measures

Measure	Example	Dashboard Type
Comparison against the same measure in the past	Number of attacks last year compared to today	Tactical/strategic
The current state of a measure	Number of exposed vulnerabilities at present	Operational/tactical/strategic
The relationship to a future target that the measure should meet	Percentage of unpatched machines	Tactical
A prediction of the measure established sometime in the past	Forecast of the cost to keep antivirus signatures current	Tactical/strategic
The relationship to a future prediction of the measure	Percent of machines updated to a new operating system this quarter	Tactical/strategic

continues

Table 5-2 Dashboard Comparative Measures *(continued)*

Measure	Example	Dashboard Type
A value reflecting the norm for this measure	Average number of failed logins per person, or the normal time ranges when people log in to their desktops, or the number of hours it takes on average to patch critical systems	Operational/tactical
Future prediction for a measure	Number of machines that need replacement in a year or the size of the user population in a month	Strategic
A different version of the same measure	How the risk posture compares to other companies in the same industry	Strategic
A related measure to compare with	Cost of running in-house security monitoring compared to the security and risk posture that would result if security monitoring were outsourced	Strategic

As can be seen from Table 5-2, most comparative measures are useful for strategic dashboards. Tactical and, especially, operational dashboards more commonly use fixed measures to convey a state of a measure. Communicating states of a measure, as opposed to a comparative expression, can be done in many different ways. However, it is not only measurement or comparison that a dashboard is used for. A dashboard can help address a number of other objectives and goals. I mentioned *aggregation* and *summarization* of information in previous chapters. This means that various heterogeneous data sources are possibly combined into one single measure. The indicators from these devices are generally summarized to represent one single value or measure that presents the big picture. A more complicated process than simple summarization is the process of *correlating* information. We have discussed correlation at length in different places throughout this book. It is one of the main methods for helping to reduce the base information into manageable pieces that help **trace** and **monitor** the state of security. These are both passive means, but dashboards are also used for active tasks such as **alerting.** The dashboard should have the capability to quickly communicate measures that are out of their norm. It should alert and help focus the attention on the most important areas.

The consumer or viewer of a dashboard is often using it to **predict** future states. As always with predictive analysis, however, you have to really understand all the factors that can change a measure and weigh them against known or possible future changes of those. How are they going to change, and how in turn are they going to change the measure?

What is of utmost importance when designing a dashboard is that it is designed for the target audience. A CISO is not interested in the same measures as a security analyst. The analyst is likely interested in much more technical and operational detail, which the CISO generally doesn't have much interest in. He or she is going to be interested in aggregate measures and higher-level metrics. As a case study, let's identify what a CISO dashboard could look like.

The CISO Dashboard

I talked to a few CISOs about what they want to see on their dashboard. Interestingly enough, I got as many different answers as I interviewed people. On the other hand, there was definitely a common theme: "Show me when my crown jewels are at risk." Every company has a different set of crown jewels. For example, for the private bank in Geneva, Switzerland, it is the list of numbered accounts along with the information as to who their owners are. The other common theme is risk management. Ideally, they would like to see how much risk their information is exposed to. However, the way to measure and display the risk differs among all the CISOs. I got the many different answers mainly because every CISO has a slightly different role and corresponding responsibilities. Some of them are fairly technical and have no problem in engaging in detailed discussions about network devices and such. Those people usually tend to want more technical details on their dashboards. In contrast, some CISOs are not technical at all. They come more from the business side. For them, a dashboard that shows TCP port numbers and such is useless. You will find all possible profiles of CISOs between the two extremes I have outlined.

A CISO's job, no matter how technical he is or what his exact job description looks like, is to keep an organization's information secure. He is generally situated more on the business side than the technical side, although most people working for him are fairly technical. This has the effect that he will need a dashboard that aggregates and translates the technical details of information security into business metrics. Terms such as *mean time to repair* (MTR), *threat,* and *exposure* will definitely play an important role. So what does the CISO's dashboard look like? What information does it convey? Before we can answer these questions, we have to think about the goal, the purpose of such a dashboard. As mentioned previously, the CISO needs a knowledge management dashboard. He needs a way to see the current state of information security in his organization. The dashboard needs to aggregate and report on some key performance indicators (KPIs) that enable him to make informed business decisions based on facts and hard numbers, not on feelings and unfounded indicators. A dashboard needs to enable the CISO to take action so that security is managed to business expectations. More and more, the security organization is not viewed as a cost center but instead as a service that has to show

business value. This clearly means that the technical inputs, the technical data collected by systems and devices, need to be translated into business, policy, and compliance requirements.

The information flow should not be unidirectional. The technical indicators are not only aggregated, summarized, and reported up to the CISO's dashboard, but the flow also works the other way around. The indicators collected will (or should) result in policy adjustments, process improvements, and new IT controls. The information flow should be a learning process that helps track exceptions to then improve security architectures and policies and enable management processes to be tuned.

These requirements merely give us a high-level idea of what the CISO will be interested in. It should be clear at this point that we should not list metrics such as the number of vulnerabilities on a host or the amount of free disk space on our critical servers. These are measures that are important for operational and technical dashboards. What are some measures that a CISO would find interesting? Compliance is one of the areas. This includes compliance with regulations, such as Sarbanes-Oxley, HIPAA, PCI, and so on, but it also means compliance with company policies. Another area is strategic risk management. What is the risk landscape surrounding critical company information? How efficient is the security department? Where is potential for improvement to simplify processes and make them more effective?

An important factor to always keep in mind is that a CISO has reporting responsibilities, too. He needs to justify his budget with the finance department and the board of directors. The board in turn would like to see how well the company complies with regulations and how risk is distributed across the organization. Although IT or informational risk is only a part of the total risk, it is an important factor in the overall risk equation.

So what does a CISO's dashboard look like? Figure 5-33 shows you!
Note that for the dashboard in Figure 5-33, I am not going to discuss how to calculate the individual indicators. The risk calculation, for example, is not the topic of this chapter. It is the way the data is represented that is important. On the top left, you find an indicator of the overall current status of information security. If this traffic light is red, a critical incident is currently happening. This could be that a router absolutely critical for financial trading is down. Whether this is indeed a security problem is not important at this point. Every minute such a device is down costs the company a lot of money in lost productivity. It is important to investigate the possibility of a security breach at this point. The second traffic light indicates the risk associated with the "crown jewels." This light should always be green. If it starts turning orange or even red, the crown jewels are being exposed to a significant risk. In some cases, this means that the information is being accessed anomalously. This is important if you have, for example, a recipe for a beverage that is highly confidential. It could be that the integrity is at risk for information such as

chemical compositions of materials or it could be that the availability of the core customer service is not guaranteed anymore, resulting in significant monetary losses for the company. However you map the most significant risk, this traffic light should represent the state of that risk.

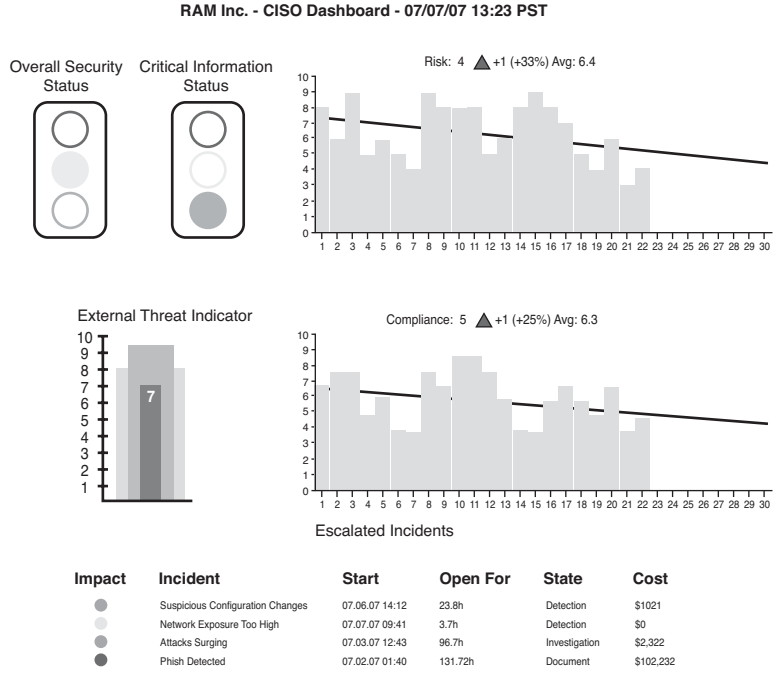


Figure 5-33 A sample dashboard for a chief information security officer.

On the top right of the dashboard, the dashboard shows how risk has developed over time. The same trend information is shown for the state of compliance. Compliance means how well the policies are implemented, assuming that the company actually has meaningful policies that also map to the regulatory mandates to which the company has to adhere, such as Sarbanes-Oxley for companies that are traded on the U.S. stock market. Again, I will not discuss how to map data into this chart. I am assuming that there is a way to map the compliance state into a number between 0 and 10, where 10 indicates perfect compliance.

An interesting way to use these two trends is to analyze the impact of external factors on the measures. For example, suppose you are running a security-awareness program. What effect does that program have on the risk exposure and the compliance behavior of

your network? You will hope that the program has the effect that the risk decreases and the compliance measure increases. The nice thing about having a way to measure these indicators is that it suddenly is possible to document an impact on or an improvement in specific measures. This is incredibly useful when trying to justify investments or show how investments manifested themselves in the network.

The last graph in the CISO dashboard indicates the external threat. The three differently colored bars indicate the average threat year to date (light gray), month to date (dark gray) and today (black). We can see that this instance shows that generally there is a high threat level. This month was higher than the average year to date, but fortunately today seems to look slightly better than the rest of the month. Finally, the bottom part of the dashboard shows the top four critical problems that need companywide attention. Not all of them are pure security problems, but they do potentially have a security aspect to them. The color indicates the assessed impact of these incidents. All the other data about the incidents helps prioritize the mitigation efforts.

One other indicator that would be interesting for CISOs is a comparison of certain measures with other companies, either in the same industry or even across industries. For example, if a financial services company could compare its compliance state with other players in the industry, the outcome could be used to either justify the security budget or, if all the other companies are doing better, could be a means to get a bigger budget to catch up. The comparison is not at all an easy task because the measures need to be normalized. Merely comparing the number of critical incidents does not work. It starts with defining what a critical incident is. Each company would probably define this in a slightly different way. After defining critical incidents, the measure must be normalized based on the size of a company. Protecting 10,000 machines is a completely different endeavor than protecting 100. The number of incidents will likely differ significantly. However, having a capability to compare measures across companies would be interesting and useful.

Figure 5-33 is an example of a generic strategic dashboard. Independent of the company and its exact business, the dashboard can look the same. The definition of a tactical or operational dashboard, on the other hand, is dependent on the user of the dashboard, technical controls deployed, regulatory drivers, and so forth. I therefore refrain from defining such dashboards. If you are going to define your own dashboard, or you have a chance to influence how your own dashboard looks, here are some dashboard design principles that you might find interesting and helpful.

Dashboard Design Principles

The design of the dashboard in Figure 5-33 adheres to some simple design principles for dashboards. These design principles are universal, and all dashboards should follow them. The principles are as follows:

- *Use a single screen:* As soon as the viewer has to scroll around to see all the information, he needs to interact with the dashboard, which is not desirable and is not always possible. Just think of the case where the dashboard is projected on a wall. There is no way of interacting with the display. Furthermore, scrolling makes the user lose focus and get lost. Also, make sure that there are no more than five elements on a dashboard. Users generally find more than five elements hard to digest.
- *Show the right amount of detail:* Often, rounded numbers and values are easier to read and remember than exact values. Exact values might clutter and obfuscate important information.
- *Use simple graphs and designs:* Make the dashboard visually attractive, but without unnecessary decorations, such as backgrounds, gauges, and so on. Use visual design principles so as to not overuse color and other visual properties (see Chapter 3).
- *Use the right graphs and charts for the information at hand:* Use simple graphs and keep the variety to a minimum. The more of the same types of graphs that can be used, the simpler it is to read the dashboard. It is not just the choice of graphs but also how data is put in context and mapped to the graphs that is important. It is, for example, deceiving to start a bar chart at a value other than 0.
- *Provide enough context for the data:* Showing a number that indicates 100 attacks year to date does not communicate much. What is it compared to? Is this a lot? What is the trend? All sorts of contextual information could be added such as the data for the previous year or industry numbers.
- *Highlight important data effectively:* The most important data should be on the upper left of the dashboard. Use visualization theory to design the graphs. For example, the color red draws attention. You can use color to highlight exceptions.

These principles will ensure that a dashboard is not cluttered with unnecessary information and does not deteriorate to an expensive tool that nobody uses (because of difficulty interpreting it).

Unfortunately, I am not aware of a free tool that enables you to easily implement such a dashboard. One option is to use a library such as ChartDirector and build a dashboard from scratch. Some commercial solutions can help in this area, such as RTView from SL.[11] The problem with most of these solutions is, however, that they need quite some tuning and they do not necessarily implement all the previously listed principles.

We have touched on a few interesting topics in this section. I alluded to security metrics a lot. How do you measure security or aspects of it? To keep the focus on visual

[11] www.sl.com

analysis, I did not go into detail about how to measure individual components and have refrained from defining too many metrics. If interested, take a look at Andrew Jacquith's book *Security Metrics* (Addison-Wesley, 2007). It is a great book for those interested in the topic of security metrics. For in-depth discussions of dashboards and the visual design aspects of them, I can recommend Stephen Few's book *Information Dashboard Design* (O'Reilly, 2006).

SITUATIONAL AWARENESS

One topic often mentioned in the context of dashboards is **situational awareness** or **situation awareness.** The term has its origin in the military and intelligence world. It often has a connotation of real time and geographical views. However, in the end, a situational awareness view is nothing other than a dashboard. An example is shown in Figure 5-34. Here, the geographic location of intrusion detection sensors is encoded on a map. Each intrusion detection event is painted as a little square block, stacked on top of each other. Different colors are used to encode the severities of the alerts.

The graphs in a situational awareness display are near real time to provide a current situational picture. This is useful when decisions need to be made based on the actual state at the very moment. The display in Figure 5-34 helps manage a set of IDS sensors, deployed all across the United States. An analyst can quickly make decisions based on which site is currently under attack and shift his focus and attention accordingly.

When do you use these types of dashboards, and why are they useful? Have you ever been in a position to make a decision and you wished you had a view into your data to know how all your individual business areas are doing? Does your boss keep asking you about the compliance status of all the machines in Europe? Or did you have to decide where in your network you wanted to introduce a new Web service first? All these scenarios and many more are in support of a situational picture of certain aspects of your data. Situational awareness is about being able to make better decisions based on available data. If you know that currently a lot of security incidents are being investigated in your New York location and the Munich office shows absolutely no activity, it is probably smart to introduce a new Web service in Munich first.

Situational awareness screens are frequently used in command centers and projected on the wall where everyone can see them. These screens are also useful for communicating certain properties to upper management. Imagine, as another example, a traffic light type of view that indicates whether all the vital services of your business are running correctly. This can be broken down into lights that represent each individual service. Those can be further broken down to show the supporting infrastructure that enables the service to work. Again, this is all familiar turf for us. It follows the paradigm of generic dashboards.

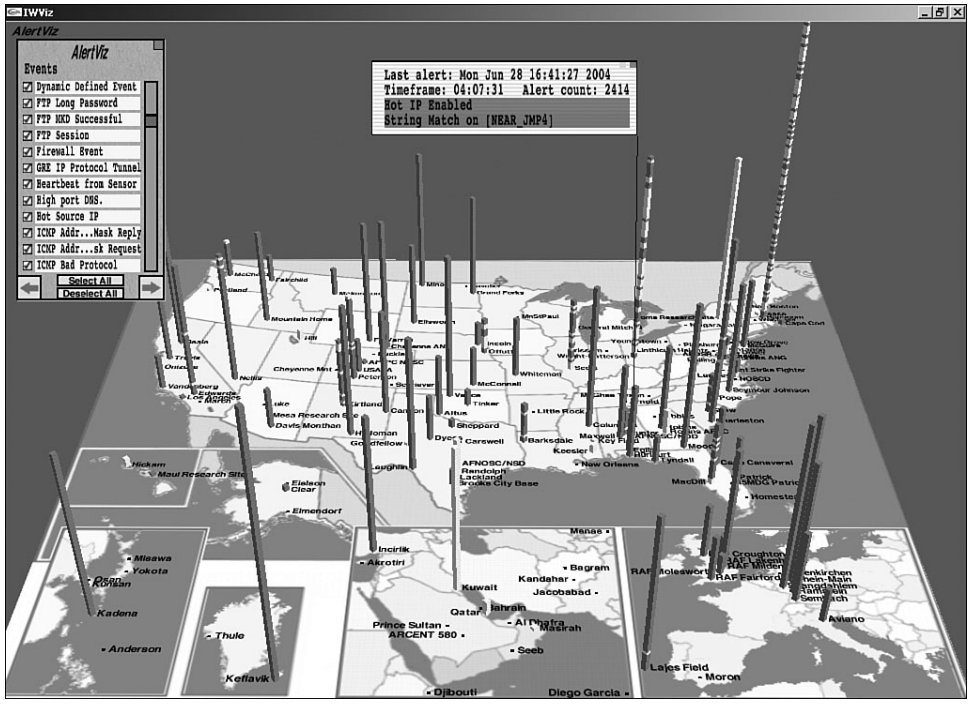


Figure 5-34 A sample situational awareness dashboard showing the distribution of intrusion detection events across sensors deployed all over the United States.

SUMMARY

This chapter showed the process of using visual methods to analyze security data. It focused on the analysis process rather than discussing the data feeds and how they can be collected or processed. We looked at two main topics of visual analysis: historical and real-time analysis. I discussed four areas of historical analysis, covering reporting, time-series visualization, interactive analysis, and forensic analysis. The discussion of these various methods has shown that there are multiple approaches to understanding log data. One powerful tool is interactive analysis of logs. The discussion introduced an extension of the information seeking mantra to include an iterative step of refining graph attributes and filters. I then discussed the forensic analysis of log data for three different use-cases: discovering attacks, attack assessment, and incident reporting. Attack

detection was done using a process that should be followed to investigate log files. It helped us not just detecting attacks, but also correlating different log sources.

After the aspects of historical log analysis were covered, I shifted to the real-time use-case. The main portion of this section was covered by discussing dashboards to communicate real-time information to help consumer of the dashboards make more accurate and timely decisions. The chapter ended with a short discussion about situational awareness, which is a special case of dashboards that links the real-time data to, for the most part, geographical locations.

Index

Page numbers followed by *n* denote footnotes.

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q-R

S

T

U

V

W-X-Y-Z