June07login_press.qxd:login June 06 Volume 31 5/27/07 10:23 AM Page 64

book reviews

ELIZABETH ZWICKY, CHAOS GOLUBITSKY, SAM STOVER, AND RIK FARROW

WHAT TO DO WHEN YOU HAVE NO BOOK REVIEW

ELIZABETH ZWICKY

Periodically, I find myself in need of a technical book on a subject I know almost nothing about (I wouldn't need a book if I knew what I was doing) when reviews are not available to me for one reason or another. There I am, standing in a bookstore, in front of a row of books of unknown quality. What do I do then?

Well, reviewing means I read a lot of books, and not all of them are books I'd pick out myself. I wish I could say they were all winners, but even the books that I am unenthused about in print are the winning tip of an iceberg. I often think with great sympathy about the editors who have to read the manuscripts out of which these are winnowed. The books I hate were selected out of thousands of even worse books.

This experience with the good, the bad, and the ugly has led me to some rules I can use to make an educated guess about which is the best of the available books on a subject. These rules work best with physical books, where you can actually read bits that you select, and of course they don't guarantee excellence. But I find they're pretty reliable.

1. Start with the cover. It is a good thing if it is a second edition or later; enough people bought the first edition to make it worth doing again. It is a bad thing if the number of authors (or editors, in the case of an anthology) is greater than the edition + 1 (so a first edition can have two authors, a second edition three, and so on). Books work best if there's a strong unifying force, and you can't achieve that kind of unity well with more than two people. However, extra authors added for later editions usually mean that one or more of the original authors dropped out. If a first edition has enough authors to make up a sports team, I'm usually ready to move on right there.
2. While you're still judging the book by its cover, look at who published it. How do you feel about that publisher? The fact is that there are good books out there by every publisher, even the ones that periodically make me wonder whether they do actually reject books, but if you're making guesses, it's fair to predict based on your previous experience. If you've never heard of a publisher before, that's not necessarily a bad thing; I'd prefer an unknown publisher to one I've tried and hated. Good new presses do show up.
3. Open the book up. Stay superficial for another moment, and flip through it. Look at the print; is it a reasonable size? If you care about fonts, now would be a moment to rise above your prejudices. Some lovely books come in terrible, terrible fonts. But a book that's not marked "large type" and is printed in big fonts and/or with big margins is a flashing warning sign saying "We wanted this book to look big and important and it isn't."
4. OK, let's actually think about the content now. Start by looking for a section in the preface that tells you who the audience for the book is. There should be one. It should not say the moral equivalent of "Everyone on earth, or at least everyone who ever touches a computer, should read this book." You do not always need to be in the target audience, although it's a very nice sign if you are. But there needs to be a target audience, or the book itself can't be very coherent.
5. Look at the illustrations. Once again, ignore any graphic design tendencies you might have. Some otherwise sane presses let authors do their own illustrations, which tends to leave them one step up from being drawn in crayon on paper napkins. Regardless of what they look like, and whether they are easily legible, think about whether they have interesting, comprehensible content. If most of the illustrations are graphs with no X and Y axis labels, or screen shots of menus, this is a very bad sign.
6. Temporarily unleash any copy-editor instincts you have, and look at 10–20 pages. If you can find grammatical errors, spelling errors, or typos on more than one-quarter of the pages you look at (and I'm not talking forgivable ending-sentences-with-a-preposition–type errors, I'm talking "Excuse me, but that's three sen-

tences held together with randomly placed commas"–type errors), you will go insane trying to read the book. Please note that all books go to press with some errors; the first one you find might be your own good luck. But if there's a regular pattern, the editors were slack.

7. Find a section, any section, that you know something about. Read that section, and see if it gets the bits you already understand right. This can occasionally be misleading if the section you know is one that is truly very tangential to the main content to the book but, in general, coverage is pretty even. If it's incoherent or wrong in deep fundamental ways about the stuff you know, it's probably not right about the stuff you don't.

These rules are of course no substitute for reviews and recommendations from people you know and trust, or even from published reviewers (which I've got to say come a poor second to personalized advice from sensible friends). But they'll usually find you the least horrible available option.

CODE QUALITY: THE OPEN SOURCE PERSPECTIVE

Diomidis Spinellis

Addison Wesley, 2006. 521 pages. ISBN 0-321-16607-8

I've enjoyed this book a lot, but can I blame it for the shortage of reviews? It's a very dense book, with something to think about in every sentence. If you carefully absorb everything it has to say and manage to implement it, you will be a programming wizard. If you try to skim it you will have a very bad headache. (At all costs avoid the "Advice to Take Home" sections, which at worst run to more than five pages of bullet points, more than the human brain can possibly take home.) Fortunately, the chapters are relatively independent, so you could just gnaw on whichever subject you're interested in at the moment and be assured of getting as much out of it as you put in.

You will, however, need to be putting energy into it. The author is willing to put all the dots down, but it's up to you to connect them. Sometimes I suspect that this is because he doesn't realize the connections aren't intuitively obvious to every reader; other times it seems intentional. Frankly, with this many pages, there isn't the space to spell everything out. It's amazing that he manages to do a good job on such a wide range of topics as it is.

This is, unfortunately, a good example of a book you should not try to judge by the usability of its illustrations, which are heavy on content and light on pixels. Some of them ought to come with a magnifying glass.

BACKUP & RECOVERY: INEXPENSIVE BACKUP SOLUTIONS FOR OPEN SYSTEMS

W. Curtis Preston

O'Reilly, 2007. 729 pages. ISBN 0-596-10246-1

REVIEWED BY CHAOS GOLUBITSKY

The problem with backups is that you can't afford to ignore them, but your knowledge about how to do them is probably out of date. Backups are a subproblem of data storage, and storage options (as your users will tell you) change rapidly. This book is designed to make sure you have the information you need to perform your corporate backups effectively and without risk of terrifying data loss. Curtis Preston's new edition is a rewrite and significant expansion of the 1999 *Unix Backup and Recovery* (but this version covers Windows as well). The book's secondary goal is to focus on free and inexpensive backup solutions.

Preston is extremely knowledgeable about the world of backups, and the book shines most when he is showing off his experience. The overview chapters are full of information on backup strategies and considerations, on features to think about when shopping for commercial backup products, and on backup hardware. Highlights of these sections range from neat tricks such as how to use the Towers of Hanoi problem to improve incremental backup scheduling, to great descriptions of how several types of tape and optical media work.

About a third of the book is dedicated to general and specific commentary on database backup and recovery, including blueprints for designing and debugging instances of database servers from Oracle to PostgreSQL. These chapters may contain more about database implementation than you wanted to know—a stated goal is to help sysadmins and DBAs communicate with each other around the subject of backups—but the information you need to get your server running again after a disaster is almost certainly in there.

A number of chapters and sections are contributed by other authors or specialists, including most of the chapters on Open Source backup systems. The best of these chapters, such as Leon Towns-von Stauber's fast and usable blueprint for performing bare-metal recovery of a Mac OS X machine, have "invaluable reference during some future catastrophe" written all over them. Unfortunately, the quality of the contributed chapters is not consistent. I found it particularly frustrating that the chapters on Amanda and Bacula focused on different design features, preventing a head-to-head comparison of the two products. Bacula's job scheduling and volume management capabilities got only a paragraph mention, whereas Amanda's had multiple pages. Having used Bacula, I know that it can do some job scheduling, and that its volume expiration facilities are sometimes confusing. Those features would have benefited from fuller treatment, and more careful editing would have improved the book as a whole.

A related word of warning is in order: This is not your grandmother's backup book. I have friends and relatives who don't do home backups (or whose backup schedule is "once every three years whether it needs it or not"), and I hoped to augment my advice for them with some of the free or cheap backup tools discussed in this book. Some programs that could be used on home networks, including rsync and BackupPC, are discussed, but most configuration examples pertain to managed networks. I would need to do other research to decide whether these tools would be feasible for non-savvy home users.

Unless your only data protection requirements fit neatly into one of the scenarios for which blueprints are provided, *Backup & Recovery* is not the last backup book you'll ever have to read. However, by reading this book, you will learn something you didn't know about the environment you are backing up and the tools you already use and gain perspective on what to think about when designing or improving your backup process.

BOTNETS: THE KILLER WEB APP

Craig Schiller, Jim Binkley, Gadi Evron, Carsten Willems, Tony Bradley, David Harley, and Michael Cross

Syngress, 2007. 464 pages. ISBN: 978-1-59749-135-8

REVIEWED BY SAM STOVER

I wasn't sure how this book was going to play out. I thought it might end up not being technical enough, but with the two names I did know from the author list (Evron and Willems) I had high hopes. I was not disappointed: I think this book is an *excellent* read for just about anyone interested in bots and botnets. I differentiate between bots and botnets because there is ample focus on both aspects of this new threat.

The book starts out with a fairly comprehensive overview of bot evolution, which I found to be the right mix of theory, history, and technical detail. This sets the stage for Chapter Two, which lays out the life cycle of a botnet. Since individual bots comprise a botnet, it's interesting to see how the technologies between the two differ. Creating a bot requires mobility and exploit code, while controlling a botnet requires a Command and Control (C&C) mechanism that's intuitive and easy to use (those poor, overworked bot-herders).

The next two chapters deal with C&C mechanisms and how different bots function in different botnets before we get to Chapter 5, which deals with detecting botnets. The first four chapters deal with understanding bots and botnets; the remaining eight deal with detection and mitigation. One of the things that I really like about this book is that the authors spend a lot of attention on two very different response/detection mechanisms. The first centers around the ourmon tool, which monitors network traffic and produces graphs to give you insight into how bots and botnets are impacting your network. There's just enough installation instruction to get you started; I'm running ourmon now, and it's pretty painless if you are running FreeBSD or Ubuntu—they have tips for installing on both. Think of ourmon as MRTG with a focus on detecting characteristics inherent to bots and botnets. Even with my little honeynet, I've found this tool to be very interesting and can easily see how beneficial it would be in an enterprise environment.

The second mechanism is called sandboxing, which focuses on interacting with the bot directly and performing analysis on the actions and capabilities. This is the main reason I was interested in this book. I've used Nepenthes (which is mentioned briefly several times), but I wanted to learn more about how sandboxing works—specifically, CWSandbox. Since Willems wrote CWSandbox, I figured he'd be the person to write the chapter on sandboxing, and I was right. Chapter 10 deals exclusively with sandboxing, with a heavy emphasis on CWSandbox. This is probably the most technical chapter in the book, and it's quite a read.

The remaining two chapters deal with resources available to learn more and communicate with other concerned citizens. There is a good sampling of various types of organizations and tools to help the bot-stricken administrator.

There were a couple of spelling and grammatical errors, as seems to be the trend in Syngress books, but nothing to get spun up over. A fair bit of overlap between chapters makes it pretty evident that multiple authors wrote the different chapters, but this was only mildly annoying. The chapters on ourmon and CWSandbox are worth the price of admission alone, and while I would have liked to see a bit more emphasis on Nepenthes, I can always hope that will follow in the next edition.

Overall, this was a great read. If you want to get a jump on the botnet learning curve, I'd recommend starting with this book.

BUILD REAL-WORLD, END-TO-END, NETWORK MONITORING SOLUTIONS WITH NAGIOS

David Josephsen

Prentice Hall, 2007. 230 pages. ISBN 978-0-13-223693-5

REVIEWED BY RIK FARROW

I came away from the 2006 LISA conference determined to learn more about monitoring, and Josephsen's book seemed like a good way to get my feet wet. I knew that Josephsen has a clear and easy writing style from his articles in *;login:*, and I had hoped that would help me make the plunge into learning about Nagios. I was right.

Nagios is a complex topic. Like any popular example of Open Source Software, there is a lot of information available online. What Josephsen adds to the online documentation is a thorough approach that starts by getting readers to consider their goals in monitoring, not just what should be monitored but how, as well as interrelationships that affect monitoring. Josephsen applies his personal experience in monitoring a large, distributed infrastructure to how he introduces the concepts, a touch that he carries throughout his book.

His introduction to the software itself is gentle, so that I am quickly convinced I can write my own Nagios plug-ins if need be. He explains that Nagios is really a scheduling and notification scaffold, and what it can do is largely up to you. After a short but mandatory chapter on installation, he gets into the nitty-gritty of configuration. This is, I expect, the place where most people who try using Nagios soon give up, as there are many files that must be edited. Although Josephsen does focus on the editing of files, he also covers the GUIs that can be used for configuring the Nagios Web interface. Along the way, he provides practical hints for making the management of the Nagios configuration itself easier.

Josephsen concludes with chapters on visualization and the new Nagios Event Broker interface. We humans are very good at extracting data quickly from visual information, and Josephsen spends a lot of his page budget explaining how to add useful graphs to Nagios, something that you don't get from just installing Nagio alone. Again, using clear explanations and an appropriate dose of humor, he explains how to start using RRDTool to produce graphs that best utilize your mental capabilities and impress managers everywhere.

The final chapter describes the Event Broker, a method that allows you to extend Nagios by adding in your own callbacks. As Josephsen explained this, I quickly understood that the Event Broker allows programmers to add to existing event handling by exposing internal global variables and data structures. Although you can't create logical service groups, for example, a collection of the hosts and services that provide a Web farm with a database backend, you can customize how Nagios deals with its events.

Three appendices round out the book: buildtime options, the many configuration options in two keys files (nagios.cfg and cgi.cfg), and command-line and plug-in arguments.

I did have some problems with the typesetting of this book. I find it amazing that Prentice Hall still can't typeset single back-quotes after all these years (having screwed this up in my first book 18 years ago), so what looks like a single-quoted string (and makes no sense) is really a command executed within a subshell, for example. That aside, I can recommend this book to you if you want to get started with Nagios or learn more about how to best set up a monitoring system.