book reviews

ELIZABETH ZWICKY,
SAM STOVER, AND RIK FARROW

MANAGE IT! YOUR GUIDE TO MODERN, PRAGMATIC PROJECT MANAGEMENT

Johanna Rothman

The Pragmatic Bookshelf, 2007. 336 pages.

ISBN 978-0-97897392-4-9

This book is about reality-based project management: what you do to try to get a project out the door, with dignity and on a predictable time line. It makes a strong argument that the only way to actually do this is to do some form of incremental scheduling, where you plan only the stuff you actually know about. I'm conflicted about this. On the one hand, I believe it's true. On the other hand, there are still organizations that just don't work this way, and although the book gives advice on coping with and subverting these organizations, I don't think it would have been sufficient for me when I was in one.

Manage It! is full of great advice about the realities of project management. It talks about hallucinating management, the realities of managing physically separated teams, and how to get programmers to provide estimates that mean something. It pushes heavily for down-and-dirty techniques that use low technology to represent what you actually know and that involve gathering real data.

One of my very favorite parts involves actual measurements on how much it costs to fix defects at various stages of a project. Yes, on the projects where she measured, it was more expensive to fix a defect found after release. No, it was not 1000 times as expensive. It was generally more like 32 times as expensive as fixing a defect noticed early on. The numbers were different for different projects, and they didn't go in a straight line. It's a small thing, but it speaks to my need for real data. 32 times is bad enough; you don't need to make up numbers that say that it's 1000 times.

The book is aimed at people who have project management experience. If you're coming at it without experience, particularly if you're at a company with an entrenched culture different from the one the author espouses, you may be bewildered by an alluring but not quite comprehensible description of the promised land of project management. There are appendixes with lifecycle and terminology definitions, but they're probably not going to suffice for somebody who hasn't encountered the terms before. The book is also fairly loosely organized.

I recommend this book for people with some project management experience who want new techniques or to improve their skills. It may also be interesting for new project managers in agile or incremental environments or those who just like jumping into the deep end.

PRACTICAL PACKET ANALYSIS: USING WIRESHARK TO SOLVE REAL-WORLD NETWORK PROBLEMS

Chris Sanders

No Starch Press, 2007. 150 pages.

ISBN 978-1-59327-149-7

It's hard to think of a more powerful tool for network management than a packet sniffer. A good packet sniffer makes all the difference; once you know how to use one, it's like taking a blindfold off. Suddenly you can actually see what's going on! Unfortunately, the other way it's like taking a blindfold off is that when you first do it, you're so overwhelmed with input that you can't understand a thing. Many people never get past that point. There's a crying need for a book that will move such people forward.

If you're the sort of person who used to cheat at games with a hex editor, this book will give you the information you need; it's got a very basic introduction to TCP/IP and higher-level protocols, a good explanation of how to make a machine able to see the packets you need and how to install and use Wireshark (which is the successor to Ethereal), and a quick run-through of some things you can do with it.

The introduction to TCP/IP was written with more practical than theoretical understanding, which is a polite way of saying that, in point of fact, the author doesn't understand what the OSI protocol stack is, although he does a pretty good job of explaining its component parts. He states that it's a recommendation, not a standard. It's not even a

recommendation; it's a theoretical model, providing only a description.

The actual packet analyses vary. Most of them look reasonable to me, and they give you some examples of how the tools work. I'm not sure how well they'll translate to further uses for somebody who hasn't done a lot of work with TCP/IP before. One of them is downright wrong; it discusses a traceroute where a router fails to return an acknowledgment. The traceroute slows down at this point, only to pick up to normal speed when it moves to the next hop. The author suggests that this shows that the performance problems on the network are somehow caused by this router. In fact, it shows that the router has an odd ICMP configuration, but there's no reason to believe that this is more than a cosmetic problem with traceroute. It isn't going to affect anything that doesn't do ICMP with the router. It might be a useful clue (for instance, it might point to a situation where all ICMP was disabled, causing problems with path MTU determination); it might also be a complete red herring. There's no way to tell from the information given.

This book makes a nice starting point for somebody who knows nothing about TCP/IP networking and wants to get started with packet analysis. It needs to be supplemented with a good TCP/IP resource if you're really going to get anywhere in the long term.

LINUX SYSTEM ADMINISTRATION

Tom Adelstein and Bill Lubanovic

O'Reilly, 2007. 272 pages.

ISBN 978-0-596-00952-6

I am getting to the point where I am a grizzled old-timer. As such, I get hostile about statements such as "For example, with almost every UNIX distribution, Sendmail is the only choice of mail transfer agent (MTA)." Fifteen seconds of research suffices to tell me that Postfix, for example, is available prepackaged for FreeBSD, NetBSD, OpenBSD, IRIX, Mac OS X, and Solaris and ships with NetBSD. I was still young and optimistic when we started running other mail transfer agents on UNIX. Without getting into long arguments about what's UNIX and what's not, it's safe to say that Linux and UNIX have a strong family resemblance that extends to running pretty much the same applications in most cases.

This resemblance then makes it puzzling that you can cover Linux system administration in 272 pages, when *Essential System Administration* takes 1176. Admittedly, *Essential System Administration* covers both UNIX and Linux, but given that the *Linux System Administration* authors find Linux more complex and capable than UNIX, it ought to take up about half the space, or at least a third. The reason it doesn't is that the authors concentrate on installation instructions for specific software packages. They do provide some explanation of concepts, but not much, and what they do explain is not always right.

For instance, they start right out by having the reader install a bunch of services on an Internet-connected machine, in their default configuration. To do this, you'd have to be a lot more trusting than I'd care to be; it's a big bad Internet out there. Then, they talk about not logging in as root, but using su—so they have you create an admin account, so you can log in as that instead of root. This completely misses the point of not logging in as root, which is to log in as an identified individual user.

Not to obsess about mail, but their discussion of mail transfer agents confuses a mail transfer agent and a mail server. People don't retrieve mail from mail transfer agents. Mail delivery agents don't retrieve mail from mail transfer agents. The mail transfer agent shoves the mail somewhere and leaves it there for the mail delivery agent or the mail user agent to pick up, with no further involvement. Their discussion of open relaying confuses it with spam in general, showing an open relay as allowing inbound spam; the problem with open relays is that they pass spam, which is neither inbound nor outbound but merely passing through.

And it's not just mail; the backup chapter advocates writing your own backup scripts, without discussing any of the risks involved, and then says that databases "have their quirks" when it comes to backups without clarifying what those quirks are. (Their primary quirk is a violent, often fatal, allergy to having files in an inconsistent state, accompanied by behaviors that frequently result in backed-up files being in such a state. Not knowing this will probably get you into nasty trouble.)

If you have a solid conceptual background in system administration but want a leg up installing a Linux system, this is an interesting walk-through of popular alternatives on Debian. If you don't have the conceptual background, it's not very helpful. At least couple it with a serious system administration book.

VIRTUAL HONEYPOTS: FROM BOTNET TRACKING TO INTRUSION DETECTION

Niels Provos and Thorsten Holz

Addison-Wesley Professional, 2007

ISBN 10: 0-321-33632-1; ISBN 13: 978-0-321-33632-3

REVIEWED BY SAM STOVER

This book is so good that I haven't finished it. I've spent so much time actually *doing* the stuff that I haven't even touched a good third of the book. On the one hand, that limits my ability to give a thorough review, but on the other, I feel confident that the bits I haven't read will live up to the part I have read. OK, enough syrup: let me tell you why I like this book so much.

The book begins with almost 20 pages of honeypot and IP background, which I promptly skipped, then went back and read because I have a responsibility to my readership. It's a good thing I did, because beyond the basic IP review, there's a very succinct and appropriate comparison between high- and low-interaction honeypots. This distinction permeates the entire book: the sections are divided between methods and uses for each type. As the names suggest, the differences deal with the complexity and capability of the honeypot: high-interaction systems require more care and feeding but have the potential to collect different data from low-interaction. It's important to note that both types have their uses: they just allow for different applications of honeypot technology.

After the background, Chapter 2 jumps right into high-interaction honeypots and tools such as Q, Sebek, and Argos. Q is an open source virtual machine application very reminiscent of VMware. In fact, the authors walk you through building a virtual machine, using Q, that can be run from the VMware Player. Sebek is basically a rootkit that you install on your honeypot to monitor and collect malicious activity. Argos, though, was the gem of the chapter, in my humble opinion. Argos is a new tool, developed by researchers from Vrije Universiteit Amsterdam, which monitors the honeypot in a way that detects zero-day attacks. Yes, you read that right, zero-day. Argos is a specific kind of virtual honeypot, built using Q, which marks incoming network traffic data, follows it through system memory, and, if a buffer overflow occurs, creates a report and memory dump. Memory dump analysis has been a longtime hobby of mine, and I think this is an excellent example of how that kind of technique can be used to advance the detection of malicious activity. I was impressed enough with this tool that I'm going to try to work it into a future *;login:* article.

After the high-interaction honeypot chapter, the low-interaction honeypot chapter takes you through LaBrea, Tiny Honeypot, the Google Hack Honeypot, and PHP.HoP, which is a "Web-Based Deception Framework." Interestingly enough, neither Honeyd nor Nepenthes is discussed in this chapter, but, luckily for us, each has its own chapter later in the book. I'm a big fan of Nepenthes as low-interaction honeypots go, so I was really happy to see a whole chapter devoted to deploying and managing it. I must admit that I was so busy setting up my Q/Argos setup that I just skimmed over the low-interaction honeypot chapter, but I do plan to go back and spend some time setting up some low-interaction honeypots to see what I can collect. There are definitely some sweet tools in that arena that I want to learn more about. If your interests lie in that direction, there's more than enough in this book to get you started and keep you busy.

It follows logically that if there are high-interaction and low-interaction honeypots, there have to be hybrid systems. Sure enough, Chapter 7 is devoted to tools such as Collapsar, Potemkin, and RolePlayer. Each of these systems tries to balance scalability and capability in a way that gives options to the prospective honeypotter who has needs outside of the strict high- and low-interaction products.

Chapters 8 and 9 deal with honeypots for client-side attacks and honeypot detection, respectively. I think the client-side honeypot is definitely an area of research that needs attention, and this chapter gives a good intro. After all, you can't just expect all the good stuff to happen your way—sometimes you have to go out there and collect it. Chapter 10 gives five different case studies which walk through several different compromises, all of which explore different vectors and targets. Chapter 11 spends some time talking about tracking botnets, and Chapter 12 deals with using CWSandbox to analyze malware.

In all, this book is well written, proofed, and edited. No glaring spelling errors, and the text is concise and to the point. Some of the material, such as installing Argos, is not for beginners, but some of the techniques, such as using Q to build a virtual honeypot, are very accessible to just about anyone. A truly great find: go buy your copy today.

MINIMAL PERL FOR UNIX AND LINUX PEOPLE

Tim Maher

Manning Publications, 2007. 464 pages.

ISBN 1-932394-50-8

REVIEWED BY RIK FARROW

I had wondered just what minimal Perl could be, ever since I noticed a review about it on Slashdot. So I tracked down the publisher and got a copy of my own. I felt that my Perl skills could certainly use some honing, and I would be motivated to read and use this book, as long as it worked well.

Maher has done a fine job providing an alternative path to learning Perl. The "minimal" in the title has to do with Maher's choice in how to present the material, not in the sense of providing the minimal amount of Perl. Maher starts right out by taking the reader to the fictional land of Perlistan, where there appear to be four different languages, but everyone who lives there can understand each other. What he is referring to are different styles used when writing Perl, and his teaching technique is to stick with a single style that is both efficient to use and easier to read for, say, a shell programmer.

The mention of UNIX and Linux people in the title is not gratuitous, as Maher uses comparisons to the Bourne, Bash, and Korn shells and the grep, egrep, find, and awk commands to illustrate how Perl works and how using Perl provides features that you can't get from using the shell and commands alone. I found myself learning about new features of commands (although keep in mind that I learned how to use grep in 1982), as well as other tidbits that may not be new to people who learned Linux/UNIX in the past ten years.

And that's just a side effect of *Minimal Perl*. Maher does a great job of presenting Perl, from command-line arguments, one-liners, to scripts. The first part of the book focuses on comparing Perl features to those you can have with UNIX commands alone, and it works very well at helping people who already know UNIX learn Perl. The second part of the book works with Perl more as a programming language: for example, Chapter 10 is about looping. You might wonder how someone could spend nine chapters and *not* mention looping, but consider just the intricacies of regular expressions and how regular expressions function differently in sed, grep, and Perl, and you will begin to understand the gentle and thorough path Maher takes.

Maher does discuss using modules (early on, in fact) and CPAN. Object-oriented programming barely gets a mention, although the reader gets introduced to the use of objects along the way.

I can recommend the book to people, like myself, who want a thorough refresher course in Perl. I imagine this book will work great for those who have some grounding in UNIX/Linux but don't yet know Perl. In particular, if you know someone to whom you handed the Camel book (*Learning Perl,* by R.L. Schwartz) and he or she just didn't get it, then try *Minimal Perl*. Its simpler approach may provide just the trick needed.