book reviews

ELIZABETH ZWICKY
WITH
PAUL ARMSTRONG,
SAM STOVER, AND
RIK FARROW

MASTERING REGULAR EXPRESSIONS: UNDERSTAND YOUR DATA AND BE MORE PRODUCTIVE

Jeffrey E. F. Friedl

O'Reilly, 2006. 484 pages.
ISBN 0-596-52812-4

By a strange twist of fate, a significant percentage of all the Java code I have ever written went directly into production on mission-critical systems. For all I know, it's still deployed: all five lines of it. What has this got to do with regular expressions? I ended up writing this delicate, important code that we were going to have to release without proper testing in a language I didn't know because I wasn't scared of regular expressions. Not, you will note, because I was a wizard with them—no, I was merely competent, confident enough about regular expressions, and nervous enough about breaking things that people felt it would be in good hands. And, indeed, it worked better after I fixed it than it did before.

Had I read this book first, I probably would have done a better, faster, more efficient job and felt calmer doing it. I started reading this book with the warm confident glow you get when you're reviewing on good, firm ground—here's a topic I know something about. No, I didn't. I mean, obviously I did have some basic competence, but I still spent a lot of time going "Wow. Now that's cool." And even more time going "Huh? Now if I read that again, I'm going to really understand something complicated and interesting." The author at one point says, "It's not necessarily easy to wrap one's brain around this, but once it 'clicks,' it's a valuable tool." This is certainly true of the *recursive* regular expression he's discussing at that point, but it's more generally true of the book as a whole.

At some point, you are going to need to do regular expressions. You could do it by trial and error and reading manuals, but believe me, I've seen a lot of people do that, and it's not pretty. Instead, you should buy this book. If you don't already know something about regular expressions, it's going to be slow going. That's pretty much the nature of the territory; take heart from the idea that it's even slower when you do it by trial and error. If you've managed to solve a number of interesting problems with regular expressions, but it was harder than you'd hoped, there are some odd failures, and you have a feeling that there was luck involved, you should find this not an easy read, but an enlightening one.

Does it matter what language you're struggling with? Probably not. The book covers Perl, Java, .NET, and PHP in chapters of their own, and it has tables for a number of other languages, plus information on how to figure out what's going on inside your regular expression engine if it's not covered. Besides, the basic concepts are language-independent.

WHY SOFTWARE SUCKS . . . AND WHAT YOU CAN DO ABOUT IT

David S. Platt

Addison-Wesley, 2006. 242 pages.
ISBN 0-321-46675-6

This book is not really meant for us. Instead, it is meant for nontechnical people who are computer users and want to know why technical people are torturing them. It will be enlightening, sometimes in a "self-knowledge hurts" kind of a way and sometimes more in a "Gee, those other techies are just like us, I guess" kind of a way.

I enjoyed it thoroughly—I like a good rant, when I agree with it—and I did learn a few things. I feel warmer and fuzzier about software phoning home when it crashes, for instance. Mostly, I thought it was good fun, and actually surprisingly positive, given the title.

CATASTROPHE DISENTANGLEMENT: GETTING SOFTWARE PROJECTS BACK ON TRACK

E. M. Bennatan

Addison-Wesley, 2006. 270 pages.
ISBN 0-321-33662-3

This is a practical guide to figuring out if you are dealing with a certifiable catastrophe and then getting your project back on track, if at all possible. You can tell it's a practical guide, because sometimes it tells you that, in fact, it is not possible to recover, and you should give up. It is a systematic, process-oriented book, better suited to large companies with official "Projects" than to startups, but even so, it has a refreshing willingness to admit the existence of politics and personal feelings.

This book is best suited for somebody with some management experience. It would be helpful if you think the project is going down the tubes, but everybody else is either more experienced or on too many happy drugs, you're not sure which. It would also be helpful in a situation where everybody agreed there was a disaster at hand, but nobody knew how to get out of it again.

HACK THE STACK: USING SNORT AND ETHEREAL TO MASTER THE 8 LAYERS OF AN INSECURE NETWORK

Michael Gregg, Stephen Watkins (Technical Editor), George Mays, Chris Ries, Ron Bandes, and Brandon Franklin

Syngress, 2006. 442 pages.
ISBN 1-59749-109-8

I began worrying about this book when I got to the subtitle. No, I don't mean I objected to the bit about the 8 layers—I figured (correctly) that they'd intentionally added a layer to the OSI model that involves people. I mean the bit about Snort and Ethereal, which are lovely tools, but minimally helpful at hacking people and downright useless at the hardware level. However, subtitles, like the rest of the cover, tend to be editorial decisions over which the authors have very little control.

The idea of using the OSI networking stack as a way to understand networking as a whole, in a practical context, is a good one. It's made more difficult by the poor match between the OSI model and the way TCP/IP works in practice, but a bit of jiggering makes it come out functional.

Nevertheless, this book tries to cover everything about network security. And I do mean everything—from the breeds of dogs that might be appropriate as guard dogs through the bits in an Ethernet frame through encryption, secure software development, and security policies. The results range from OK to so bad they're funny: "Depending on placement, the windows should be either opaque or translucent." Yes, this is referring to physical windows. It is unclear to me whether the authors meant "Windows in high-security areas should be either opaque or translucent" (although if you're going to make the thing opaque, surely getting rid of it would be a better option) or whether they really meant "translucent or transparent." But in any case, in my world, network security is not incompatible with having at least the occasional transparent window.

The coverage of physical security and encryption is worth avoiding at all costs. About Basic XOR encryption it says, "This type of encryption does not contain any mathematical theory or functions that would introduce diffusion or confusion, which helps prevent cryptanalysis attacks that are based on statistical analysis." Now, my understanding of cryptographic theory is basic, but it's sufficient to tell me that XOR is a mathematical function, that diffusion and confusion are complicated, interesting concepts that you can't mention once without further explanation, and that XOR with a key that's significantly shorter than the encrypted text is in fact vulnerable to statistical analysis.

The interesting part of the book is the demonstrations of practical uses of snort, ethereal (or wireshark, as it is now known) and other tools, some of them intended for good, some of them neutral, and some of them distinctly ill-intentioned. This book would be even more useful if the screen shots were more legible, but there's lots of installation information. On the whole, I'd recommend choosing a different book on hacking tools and system administration.

IPV6 ESSENTIALS, 2ND ED.: INTEGRATING IPV6 INTO YOUR IPV4 NETWORK

Silvia Hagen

O'Reilly, 2006. 418 pages.
ISBN 0-596-10058-2

REVIEWED BY PAUL ARMSTRONG

If you're after a solid technical reference on IPv6, this is certainly a good resource. The writing style makes for easy reading and the book is well laid out, starting with some history and then working its way up the stack as you progress through.

There's also a vast amount of information about not only IPv6 but all the networking technologies that you might employ when using it. Making the book even more enjoyable to read is that everything is referenced by RFC and there are notes when an RFC has been superseded. If you're thinking you've forgotten too much about the ins and outs of networking, an introduction to the required information to understand a chapter is, for the most part, also included.

Although the technical side of the book is excellent, I found a few of the sections on deployment to be a little thinner than I would have liked. Particularly frustrating was the case study of the University of Porto, where the sentence "They see the project as a very interesting experience with some peculiar and proactive measures to overcome various problems that had to be solved by the network administrators" left me with the question, "What problems and how did you solve them?"

Although it's aimed at a technical audience, if you're a manager and your team is considering deploying IPv6, you should consider reading Chapter 1, which covers history, why you might deploy it, and misconceptions, and pages 285–310 of Chapter 10, covering integration, case studies, what's missing in IPv6, security, cost, and vendor support.

REAL DIGITAL FORENSICS: COMPUTER SECURITY AND INCIDENT RESPONSE

Keith J. Jones, Richard Bejtlich, and Curtis W. Rose

Addison-Wesley Professional, 2005. 688 pages. ISBN: 0-321-24069-3

REVIEWED BY SAM STOVER

If you are interested in network or host-based forensics, this book belongs on your shelf. Scratch that: It belongs on your desk. Not in your hands—on your desk. You'll need your hands on the keyboard. I don't know about you, but I learn better when I can do something, instead of just reading about it—and this book is all about doing things. Not only are there a ton of exercises throughout the chapters, there are four "Forensic Analysis" chapters that walk you through different Linux, Windows, and USB device forensic processes.

The book starts out with a chapter each on Windows and *NIX Live Response, then moves into three chapters of network-based forensics. Some of the tools and tips given in the Windows and *NIX chapters might be familiar to veteran sysadmins. The network chapters were obviously written by Mr. Bejtlich, as anyone familiar with his other works will immediately recognize his idiosyncratic hostnames and writing style. There is a small bit of overlap with one of his *The Tao of Network Security Monitoring*, which he actually references (unlike some other authors I could mention).

The first five chapters are pretty basic; however, the host-specific chapters will be a good primer for the network folks, and vice versa for the network chapters and the host folks. At this point, everyone is at a common starting point, and digging into the forensic process begins. Chapter 6 walks you through forensic acquisition, which is the process by which you obtain the data to analyze. There are plenty of tips that relate directly to the legal process—probably the most intimidating aspect of forensic work. A good example is documentation: There is a whole chapter named "Before You Jump In" with a three-page section labeled "Document, Document, Document!"

After the acquisition comes the analysis, and this is where the book really shines. There are seven chapters dedicated to forensic analysis techniques ranging from Windows Registry Reconstruction (Chapter 12) to analyzing Windows and Linux files of unknown origin (Chapters 13 and 15). All seven chapters have numerous pseudo "real-life" case studies—along with actual data on the included DVD, which gives you the feeling that you're truly doing the analysis.

Now that you're hooked on the forensic process, Chapters 16 and 17 help you build your own tool kit. I found this part of the book to be a little lean, but I know any author is at a disadvantage when trying to keep up with all of the latest and greatest tools out there. Since this edition was published over a year ago, there are a number of tools that aren't discussed. Personally, I'd really like to see the next edition have a bit more coverage of open source tools. Not that the authors shy away from open source resources; quite the opposite is true. A number of great tools, such as dcfldd and Pasco, are used extensively throughout. I'd just like to see more. Several of the more common commercial tools are also discussed (e.g., FTK, Paraben's PDA Seizure, and the ubiquitous EnCase).

The remaining chapters deal with two new and rather exciting areas. There are three chapters on mobile device forensics that deal with cell phone and PDA devices. The final section focuses on online forensics, with a chapter each on "Tracing E-mail" and "Domain Name Ownership."

In all, I think this book is a must-have for any budding forensics analyst. The case studies are valid and true to real life. The formatting of the book lends itself to a classroom setting as well. Since the data is provided, it would make for an interesting exercise to see what nuggets students (and teacher!) could unearth. There are a few minor spelling and grammatical errors, but nothing worth complaining about. This is truly a fine work—I can't wait for the second edition.

PRACTICAL CRYPTOGRAPHY

Niels Ferguson and Bruce Schneier

Wiley, 2003. 432 pages. ISBN: 0471223573

REVIEWED BY ERIC SORENSON

To be clear from the outset: I cannot in this review provide an objective assessment of the accuracy of Bruce Schneier and Neil Ferguson's *Practical Cryptography*. I'm not fit to lick Bruce Schneier's cryptographical boots, let alone poke holes in his math, but I can say with certainty that the book provided me with a powerful lens through which to view the crypto systems I know and use the most (IKE, SSL, and WEP). I can heartily recommend it to anyone who wants to learn what differentiates good cryptosystems from bad ones and how to make (more) sure the ones you build are the former, not the latter.

The authors first provide a gentle introduction to first principles ("complexity is the worst enemy of security"), then step deeper into cryptographic primitives such as block ciphers and hash functions and use these building blocks to solve real-world problems, including creating a secure channel over an untrusted medium and negotiating session keys using public-key crypto. The last third of the book emerges from the heavy math to discuss technical and political issues surrounding PKI, the standards process, and technology patenting.

The book is aimed at engineers who are designing or implementing a computer system that uses cryptography, so the section on ciphers and cipher modes, hashes, and MACs has a very pragmatic format: the authors discuss the purpose of each kind of primitive, survey the landscape of options, and offer a recommendation. I found this discussion both fascinating and useful. Previously, when setting up IPSEC VPNs, I had known there were MD5 and SHA options for packet authentiation, but other than selecting the one both endpoints supported I didn't know which to choose. Now I do: The authors recommend SHA because of the higher potential for "birthday" attacks against MD5.

The book's mathematical discussions of randomness, prime numbers, and modulo arithmetic get heavy pretty quickly. The authors realize this and provide both escape hatches ("we won't use this formula [for entropy], so you don't need to remember it") and levity ("most of the math dates back thousands of years, so it can't be too difficult, right?"). They start with basic high school math concepts like the Sieve of Eratosthenes and build from there. I confess I got lost around the section discussing Garner's Formula, but I believe with time and patience one could make it through the whole discussion without needing external references or specialized training.

As I said, the book provides a lens to examine existing cryptosystems. Nowhere is this more evident than in the final chapters, on the dreams versus the reality of PKI and the evils of security protocol design by committee. Bruce's razor-sharp skewering of politial compromises to security problems will be familiar to readers of his blog and Crypto-Gram newsletters; like his nontechnical book *Secrets and Lies*, *Practical Cryptography* gives us the treat of reading his ideas in a more extended format.

Schneier and Ferguson's *Practical Cryptography* is a mine of information for people who want to know more about crypto systems, whether to create new ones or simply to understand better the protocols we use every day to read our email, conduct banking transactions, or browse the Web over wireless while sipping a mocha. The authors don't shy away from technical details, so protocol designers could use this book to avoid pitfalls they might not have known existed, but neither is this book so dense as to be incomprehensible to nonmathematicians. Highly recommended.

LINUX ADMINISTRATION HANDBOOK, SECOND EDITION

Evi Nemeth, Garth Snyder, and Trent R. Hein, with Adam Boggs, Matt Crosby, and Ned McClain

Prentice Hall PTR, 2006. 1,000 pages.
ISBN: 0-13-148004-9

REVIEWED BY RIK FARROW

This latest in the series of administration handbooks shares its friendly yet authoritative tone with its predecessors. I felt I was constantly being provided with excellent advice and guidance as I read sections of this book, advice that I generally agreed with (a nice feeling). I started out in Chapter 7, "Adding a Disk," a thorough tutorial in the various issues involved in adding a new disk to a Linux system. I wanted to see just how much what I already knew from experience matched the chapter and to learn about features I had never used. Along the way, I picked up concepts I hadn't understood, such as the use of hdparm, software RAID, and Linux Logical Volume Management. I liked how the authors have provided accurate cross references to other sections (by page number, not section number, making these easy to find). In the chapter on TCP/IP networking, I learned about mii-tool, something I knew must be there (a way to configure autonegotiation of network interfaces) but was never able to find. This book covers all of the sysadmin topics.

Even a 1,000-page book cannot cover all there is to know about the myriad system administration topics mentioned. The authors' strategy in those cases is to suggest the best references available, be they other books, man pages, or Linux doc files. They go beyond just explaining software to covering network hardware and providing advice about which backup hardware will best fit your needs. It is this soup-to-nuts approach that has helped make this the most popular system administration series of all time.

Occasionally I did identify something that I considered an oversight. For example, although Linux does support the route command, it has largely been replaced by the much more powerful ip command, which gets no mention at all (that I could find). ip appears to be a direct connection to the IP stack in the kernel that gives you more control than route and ifconfig combined, but a section of the book on this would have been great (and is on my wish list for the next edition). Other than this minor oversight, I consider this a great book, one both new Linux sysadmins and other experienced sysadmins will appreciate having.

C IN A NUTSHELL

Peter Prinz and Tony Crawford

O'Reilly, 2005. 600 pages.
ISBN: 0-596-00697-7

REVIEWED BY
RIK FARROW

I can't say I read this book cover to cover. It is a reference book, as its title suggests, and it functions very well at that. I learned C the way I was taught to learn languages in college: with a grammar, some examples, and a compiler to practice with (BDS C). That left lots of holes in my knowledge of C, holes that *C in a Nutshell* performs well to fill in.

The authors describe the use of the auto storage class specifier as "archaic," just like my knowledge of C. They just as clearly explain what the restrict type qualifier does (provides a hint to the compiler that an object will only be referenced via the given pointer). The single largest section in the book covers libc, and it differs from the online man pages in several ways, the most important of which is to provide program examples showing how the function is used. The book starts with the grammar (223 pages), then explains the standard headers and functions (next 270 pages), then finishes up by describing gcc, make, and gdb.

Altogether, this book is a great desktop reference for anyone who needs to understand C. Somehow I have managed to read and write C programs without this book, but I will certainly appreciate having it handy in the future—it is now within easy reach of my favorite chair.