Book Reviews

MARK LAMOURINE, BRANDON CHING, PETER H. SALUS, JEFF BERG, EVAN TERAN, AND RIK FARROW

Jenkins: The Definitive Guide
John Ferguson Smart
O'Reilly Media, 2011. 406 pp.
ISBN 978-1-449-30535-2

Anyone who's worked on the development side of system administration for any length of time has probably put together some form of automated build system. Those of us who are older probably did it from scratch.

A corollary of the Agile development process is the need for a strong automated build and test service. The popularity of Agile development has led to the improvement of services to manage the build and test process and to provide visibility at each step. To oversimplify, the idea is to rebuild and retest with every code check-in. Ideally, little bugs are discovered early and fixed before they grow into big problems. Of course, people have given this a name: continuous integration. Jenkins is a fairly recent addition to the tool set.

It's important to understand at the outset that Jenkins doesn't really do anything by itself. Jenkins manages and coordinates a series of activities that would otherwise be done manually or not at all.

Jenkins: The Definitive Guide is light on philosophy. The introduction and justification for using continuous integration in general and Jenkins in particular takes just eight pages. That includes a three-page seven-step timeline to get from no automation to complete continuous integration. The rest of the book actually does a fine job of filling in the outline but doesn't waste any effort trying to proselytize.

The book is both sparse and dense. Jenkins integrates with dozens of other pieces of software, and the author doesn't try to hand-hold the reader through any of them. For example, Jenkins is a Java application. Java installation and verification take a single paragraph with a URL reference to the Oracle Web site for download and installation instructions. Likewise, Git installation and creation of a GitHub account to download the sample jobs for the next chapter take just over a page.

Each chapter covers a high-evel task for the user. These correspond to the configuration of different Jenkins activities. They also match the timeline presented in Chapter 1 for gradually introducing continuous integration.

The first four chapters cover the introduction, preparation of the environment, and installation and configuration of Jenkins. The example project for the book is hosted on GitHub and the book instructs the reader to create an account and fork the sample repository. I'm not sure how I feel about reference books which depend on live commercial services. Realistically, the book is likely to be obsolete before GitHub goes belly up, but it bothers me somehow.

The remaining chapters cover creating build jobs, automated testing and notification, authentication and access control mechanisms, automatic code quality scanning and reporting, and automated deployment. The final chapter talks about updates, configuration backups, and storage management and server loading. I was glad to see that because I didn't see anything about capacity planning at the beginning.

On top of the standard features there is a long and growing list of plugins for open source and commercial applications available, each of which will require some knowledge of the related software. I think this is a good thing, but it's clear that this book (and this software) is not for the novice developer or sysadmin.

Jenkins was written in Java and to manage Java projects. This shows in the focus of the examples in the book. The plugin list, though, shows that Jenkins is being used to manage projects in a dizzying array of environments. All of the current major scripting languages and Web frameworks are represented, as are a number of more obscure (to me) tools. The book doesn't cover these, but if a plugin exists for your language or environment it doesn't look like it will be difficult to install or configure.

Jenkins is managed almost entirely through the Web user interface. The book is full of screen shots illustrating the text. One concern is that any changes to the visual layout of the user interface will make portions of the book obsolete.

User interface evolution isn't a problem limited to books and software with graphical interfaces, but I think it could be a greater problem for them.

"Guide" is an appropriate term to use for this book if you take it in the sense of a tour rather than a reference. Each chapter shows you something important and concrete but then leaves you at the entrance to some new place for you to explore on your own, since each subject is covered well elsewhere. Take the tour and then decide what parts you want to explore next and in greater depth.

This book is licensed by the author under the Creative Commons license. The content is available on the Internet at DocBook. This is a case of Tim O'Reilly putting his money where his mouth is. He's spoken out publicly against draconian government-enforced content monopolies. He's willing to publish CC licensed books. He knows how to decide what will make money, and he's convinced that he doesn't need an exclusive copyright to publish a book worth buying. And he's right.

—Mark Lamourine

Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/cURL, 2nd Edition

Michael Schrenk

No Starch Press, 2012. 362 pp.
ISBN 978-1-59327-397-2

In the second edition of *Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/cURL*, Michael Schrenk introduces you to the world of automated webbots and scripts that can filter, parse, store, and process Web-based information that suites your needs. Using the massive information available online and through the methods described in this book, you can wield that data almost any way you want. Need to collect metrics or parse and store content for your academic research? What about wanting to have an email sent to you when your bank account gets low? Perhaps you are watching the price on an eBay auction and you want to be notified—or even have your script automatically bid for you—when it hits a certain amount. Webbots can do all of these things and more.

The book is broken up into four parts comprising 31 chapters. While this sounds like a lot, most chapters are short and to the point, covering a specific topic and guiding you to external sources where appropriate. The first section covers foundational techniques and technologies and introduces you to the PHP language and the cURL library. The second section delves into some simple projects of common Web automation tasks and demonstrates the general script flow and how specific libraries function to get results. Some example projects here are price monitoring, form submission, aggregation, and email-reading/sending webbots. Section three covers advanced topics such as dealing with cookies, encryption, authentication, and macros. Finally, section four covers general webbot considerations such as fault tolerance, stealth, and webbot-friendly Web design.

As someone who has been writing PHP/cURL webbots and spiders myself for about seven years now, I was particularly excited to read this book. I think Michael does a good job of covering most of the basic techniques and challenges having to do with webbot scripting. Although there are a number of good chapters that will get you up and running quickly, I do feel that most of the chapters were a bit cursory, lacking some very important coverage of alternate tools and techniques. Also, Michael doesn't delve deeply into topics such as AJAX and complex JavaScript-driven sites and forms. In my experience, a large number of sites use these kinds of techniques, and without the right tools they can be very difficult to parse.

Overall, *Webbots, Spiders, and Screen Scrapers* is well-written, easy to follow, and will get you started quickly. Having said that, its lack of depth in certain areas definitely makes it most appropriate for beginning developers/scripters.

—Brandon Ching

A Culture of Innovation: Insider Accounts of Computing and Life at BBN

David Walden and Raymond Nickerson, eds.

Waterside Publishing, 2011. 559 pp.
ISBN 978-0-9789737-0-4

We owe a great deal to the concept of the industrial lab, the first of which was that of Thomas Edison in Menlo Park, NJ (now renamed Edison). World War II gave rise to IBM's Research Division in Manhattan and, since 1970, in Yorktown Heights, NY. It also saw the start of Bolt Beranek and Newman in Cambridge, MA, in 1948.

This volume incorporates the narratives of 19 of those who were involved in BBN over a period of 60 years. It is not 100% new material. Some of the contents originally appeared in two special issues of *IEEE Annals of the History of Computing* (v. 27.2 and v. 28.1 [2005, 2006]), but many of those articles reappearing in this volume are expanded versions, and some of the chapters are completely original to this work.

Dick Bolt and Leo Beranek founded their partnership in 1948. Their first job was the acoustical engineering of the not-yet-built UN headquarters in New York. Bob Newman became

a partner and the name was changed in 1950. I am not going to discuss acoustics, but the Harvard Electro-Acoustics Lab and the MIT Psycho-Acoustic Lab underlie all the work of the following decades. BBN grew rapidly. By 1960, there were offices in Los Angeles and Chicago; there were 128 employees in Cambridge, 22 in LA, and three in Chicago.

One of the "bright young men" at MIT was J.C.R. Licklider. In 1962, Lick went to Washington, to ARPA, where he became a prime mover in the expansion of computing and in what would become the ARPANET. In 1968 BBN's response to the RFP for the ARPANET was complete. In October 1969 the first two IMPs (Interface Message Processors), one at UCLA, one at the SRI in Menlo Park, CA, communicated with each other: the 21-year-old corporation would make key contributions to a technology that was to change the world.

But the ARPANET/Internet was far from the end of BBN's innovations.

BBN had been the "experimental" site for DEC's PDP-1; thanks to Licklider, Dick Pew, and their cohort, the field of human-computer interaction came into being; networked email was born here (thanks to Ray Tomlinson; pay no attention to the Shiva Ayyadurai silliness); time-sharing was demonstrated here, etc.

I don't want to turn what should be a review into a menu, but if you are reading *;login:*, you owe a major debt to BBN: acoustic signal processing, control systems, torpedo data analysis, several medical applications, educational technology, speech processing, and natural language understanding are merely a few of the topics discussed in this volume.

Not everything is an easy read; some chapters are better than others. But the work is quite significant in that you can hear the voices of a number of remarkable individuals. I learned and profited from every chapter.

One of the most fascinating is Stephen Levy's "History of Technology Transfer at BBN," covering 1948–1997 (the last decade is covered in Walden's "Epilog"). I had not realized the number of business relations of various kinds BBN (and successors) had entered into, nor what a large percentage of the exchange of agreements was profitable.

Although currently a branch of Raytheon, BBN still functions as a research and development facility, unlike Bell Labs or XeroxPARC. This volume is a fitting monument. If you are at all interested in technological history, *A Culture of Innovation* is more than merely a worthwhile investment. All the contributors deserve my thanks; Walden and Nickerson, my gratitude.

—Peter H. Salus

The Tangled Web: A Guide to Securing Modern Web Applications

Michal Zalewski

No Starch Press, 2012. 268 pp.
ISBN 978-1-59327-388-0

Michal Zalewski succeeds in condensing into a single comprehensive volume topics that could easily fill several books, and he provides the right reader with exactly what he or she needs to know regarding the Web, the browsers we use to navigate it, and the considerations we need to be aware of to secure it. Depending on your experience with these topics, this is not a breeze-through read; this book requires time and attention to grasp and hold the topics covered.

The book is split into three parts: "Anatomy of the Web," "Browser Security Features," and "A Glimpse of Things to Come." The first third of the book really lays the groundwork for the remaining portions, giving the reader the necessary background to understand what makes the browsing experience work. Zalewski covers the basics, devoting entire chapters to the makeup of a URL, the Hypertext Transfer Protocol, HTML, Cascading Style Sheets, browser-side scripts, non-HTML document types, and browser plugins. As I'll expand on later in this review, Zalewski crams enormous amounts of information into the 15–20 pages (on average) per chapter. Whether it is technical detail or history behind the design of a browser or standard, he leaves no stone unturned. The second third delves specifically into security features of browsers, languages, and plugins, with chapters on topics such as content isolation logic, origin inheritance, content recognition mechanisms, and handling rogue scripts. The last third of the book looks at security features and browser mechanisms that are expected to emerge. A chapter is devoted to common Web vulnerabilities.

The level of detail Zalewski goes into is excellent. Topics such as browser history, design considerations and behavior, HTML markup, security features in plugins, and secure coding are covered with minimal filler and with examples that illustrate discussions throughout. Only rarely is the reader left with questions, and the text provides detailed references pointing the reader toward additional information if necessary. As an example of the depth that Zalewski provides for a given topic: the HTML chapter covers everything from a discussion of the RFC and subsequent HTML version evolution to browser parsing behavior in handling different code segments, including the role UTF-8 characters can play in manipulating a browser's parsing behavior. Zalewski acknowledges that the book is not all-encompassing in certain areas; however, I would argue that it's as close to comprehensive as is necessary. For those who get spun around in

the details, there are "Engineering Cheat Sheets" at the end of each chapter summarizing major points made throughout.

I would recommend this book without question for any Web application developer. The information within is essential knowledge to be applied in everyday efforts. I'd also pose it as essential reading for security professionals—researchers, analysts, penetration testers, etc.—who will touch the Web application space. As a member of this community, I can say the information presented is just another useful tool in the old shed that will be applicable at some point. Given the detail, you may want to keep it around to thumb through on the fly.

—Jeff Berg

A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security

Tobias Klein

No Starch Press, 2011. 208 pp.
ISBN 978-1593273859

A Bug Hunter's Diary is a unique book. Its approach to discussing the topic of computer security is completely different from any other I've read, and that's a good thing. Instead of the usual "this is what could be done," this book says "this is what I did and why."

What makes this book so different really boils down to two things:

- The level of detail given when discussing the bugs is extremely high. You will need a working knowledge of C or C++, and assembly (usually x86) wouldn't hurt either.
- The format of the book is literally that of a diary, which makes it more of a unique read.

There are eight chapters—an introduction followed by in-depth analysis of seven major bugs that the author found and developed successful exploits for.

The introduction is a good overview of the different approaches that are applicable to this type of work, ranging from static analysis to runtime analysis with a debugger to fuzzing. The author very much prefers static analysis but is quick to point out that each approach has its pros and cons and that everyone will have their preference.

Each "diary entry" is broken down into the steps that the author took to develop the exploit and closes each one with two very useful things: "vulnerability remediation," which discusses what the vendors did to patch the problem and how long it took, and "lessons learned," usually a short list basically describing some rules of thumb by which the problem could have been avoided entirely.

So what types of bugs are we talking about here? Pretty much exclusively ones that fall into the category of memory corruption. While things like XSS and other higher-level bugs are very popular now (and are just as serious), they aren't addressed in this book. These are "hard core," low-level bugs which require an intimate knowledge of the languages, operating systems, and architectures being exploited. For those not very familiar with these types of memory corruption bugs, I highly recommend reading the article "Smashing the Stack for Fun and Profit." It's an older *Phrack* article, but really spelled out the basics of how memory corruption bugs can be leveraged by an attacker.

Of course the book isn't perfect. The level of detail, while extremely impressive (and useful), can be a bit overwhelming. It's not that there is too much information to take in; it's more about presentation. Just about every page is half filled with code listings of some kind, which is great but also can make the flow of reading a little difficult. You will probably find yourself jumping back and forth between the code listings and the descriptions several times before you have that "ah-ha" moment and see what the bug is. Fortunately, the author puts the most relevant lines of code in bold so at least you know what you are supposed to be looking at.

All in all, this is a great book, especially for those who have a strong background in C or C++ programming and want to learn how to think like a security engineer.

—Evan Teran

D is for Digital: What a Well-Informed Person Should Know About Computers and Communications

Brian Kernighan

DisforDigital.net, 2011. 223 pp.
ISBN 978-1463733896

Sporting a white cover with blue lettering, *D is for Digital* mimics the look of classic Kernighan books. But the target audience for this book is not programmers, but, rather, educated people who are not CS majors.

Brian writes in his foreword that he has been teaching a class at Princeton called "Computers in Our World" since 1999, and his experiences teaching what people need to understand about computers for over a decade really shows in his book. *D* is not a textbook, but a gentle and clear journey that covers hardware, software, and communications in 12 chapters. I kept picking the book back up and reading more, partially for

the history embedded in it and partially because I enjoyed learning just how Brian approaches difficult topics. I had toyed with writing a book about computers a long time ago, but got bogged down in my explanation of binary. Brian has no problem with covering binary, assembler, file systems, JavaScript, Web bugs, and traceroute, while keeping the tone light and readable.

D is split into three sections: hardware, software, and communications. Communications covers the Internet, but also some communication hardware, cryptography, security, and privacy issues. If this seems like a lot to cover in just over 200 pages, the goal is not to overwhelm the reader, but to provide a solid background. The chapters on the Web and privacy are worth the price all by themselves.

There are no footnotes or references, in keeping with the style of the book. There is a list of resources at the back and a glossary. There is also an index, so if the reader knows what she is looking for, ADSL or "bit rot," she can find a good explanation for it in this book.

D makes a great gift for the person who is always asking questions, or perhaps for someone who really needs to know what he or she is talking about. I wish that this book were required reading for anyone attempting to write legislation related to computers, the Internet, and online privacy. I did find myself wondering what level of education should be expected of the target audience, and settled for anyone who has at least two years of college. Also, the book can be used as a reference, in that any part of the book can be read in isolation.

—Rik Farrow