Alex Groce (agroce@gmail.com), Northern Arizona University

Ross Anderson's *Security Engineering* (now in its third edition, but the second edition is reviewed here; I have only glanced at the third edition, but am certain it is also excellent, and more up-to-date) is something this column seldom considers: a fairly exhaustive, very long, and quite technical book on an actual computer science topic. It is also a book any serious "lay reader" can enjoy, unlike some of the other more technical software engineering classics we've looked at previously.

What is security engineering? Many software engineers now have to worry about security, of course, and have a general idea of "security." But what exactly security engineering encompasses is far from obvious. Anderson answers the question in the first chapter:

> "As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves.
>
> Security engineering requires cross-disciplinary expertise, ranging from cryptography and computer security through hardware tamper-resistance and formal methods to a knowledge of economics, applied psychology, organizations and the law. System engineering skills, from business process analysis through software engineering to evaluation and testing, are also important; but they are not sufficient, as they deal only with error and mischance rather than malice."

A few things are clear here: first, there is a lot of overlap with the concerns of software engineering. The first paragraph above could be referring to software engineering instead of security engineering. Second, the difference that animates security engineering is the emphasis on failure not as the consequence of a probabilistic environment or user error, but of *malice*.

In a software context, security engineering is "software engineering with a bad guy." This is one reason Anderson's book is so enjoyable to read. From the success of the Donald Westlake/Richard Stark Parker novels to the box office power of the numerous OCEANS 11, 12, 42, etc. films, it can be guessed that many of us find it fun to think like a criminal. Ok, we probably don't consider it fun to think like a carjacker or a mugger, but do consider it fun to think like a suave jewel thief or a high-stakes con artist. Or, for that matter, a black-hat hacker.

Of course, Anderson's perspective is that of someone aiming to frustrate a hacker. Or, for that matter, a jewel thief. One important point is that this is *not* a book on software security engineering, only. Anderson considers physical systems and alarms, secure printing (that is, how to frustrate counterfeiters), and other robustly "real world" security topics. This actually improves the very extensive coverage of software and computer security. Anderson's topic is security writ large, and such an all-encompassing approach means that, for example, human and psychological factors in computer and software security don't take a second-fiddle to purely

technical details of cryptographic protocols, though those are provided as well. A man-in-the-middle attack is easier to understand when it's not just presented in terms of a trace of a protocol run (does A→I:{NA, A}KI…I→B:{NB}KB appeal to anyone who isn't already in on the gag?), but as an anecdote about a MIG-in-the-middle from Cold War days (the story turns out to be too good to be true, but probably representative of real actions in other wars). The comprehensiveness means that the reader comes to understand a huge variety of perspectives, narrowly technical, broadly economic, and fundamentally social. This book has a section on computer games, but also a chapter on nuclear command and control, a section on taxi meters, and enough detail on SWIFT to help you through the recent discussions of that banking society and sanctions on Russia.

Unsurprisingly, therefore, this is a huge book; it's more than one third the size of Proust's entire *In Search of Lost Time*. It isn't going to help you avoid writing dodgy C or C++ code (for that, read Seacord's *Secure Coding in C and C++* or Howard and Le Blanc's *Writing Secure Code*, but don't recommend either to your sister-in-law, the dentist who wants to understand more about security topics). But it will give you an endlessly entertaining picture of how to make things secure when other people want to make them insecure. To understand why this book should be read by every software practitioner or researcher who is serious about security, I can think of no better advice than the following: read Section 2.4 of the Second Edition, which is available for free online: https://www.cl.cam.ac.uk/~rja14/Papers/SEv2-c02.pdf. If you don't like that, you won't like the book. If you find it interesting, concise, and full of new ways to think about a topic as ostensibly boring as passwords, then you should seek out Anderson's book if you want to better understand the big picture of security, and, incidentally, the place of software in that picture.