Computer Security and the Internet: Tools and Jewels

Paul C. van Oorschot

The official version of this book is available at
https://www.springer.com/gp/book/9783030336486

ISBN: 978-3-030-33648-6 (hardcopy), 978-3-030-33649-3 (eBook)

Pages ii, iii, iv to be provided by Springer

To Rita and Jack

Contents in Brief

Table of Contents

‡This symbol denotes sections that readers or instructors may elect to omit on first reading, or if time-constrained. A few of these are background sections, which students may read on their own as review. The end notes completing each section suggest both further elementary readings, and entry points to the research literature.

Foreword

There's an old adage that many people espouse: "Keep It Simple, Stupid". Unfortunately, when it comes to the nitty-gritty of securing computer systems, networks, and the Internet: Everything is Complicated. A highly relevant quote comes from Albert Einstein: "Everything should be made as simple as possible, *but no simpler*." It is often making things too simple that leads to missing requirements, design flaws, implementation errors, and system failures.

At the end of Chapter 1, Paul Van Oorschot lists two dozen fundamental principles that should underlie the design and implementation of systems, and particularly systems with stringent requirements for trustworthiness (e.g., security, reliability, robustness, resilience, and human safety). These principles all can contribute in many ways to better systems, and indeed they are highlighted throughout each following chapter as they apply to particular situations.

One particularly desirable principled approach toward dealing with complexity involves the pervasive use of design abstraction with encapsulation, which requires carefully defined modular interfaces. If applied properly, this approach can give the appearance of simplicity, while at the same time actually hiding internal state information and other functional complexities behind the interface.

Paul's book identifies ten relatively self-contained structural areas of widespread concern, each of which is probed with detail sufficient to establish relatively accessible groundwork for the primary concepts. The book makes considerable headway into underlying realities that otherwise tend to make things difficult to understand, to design, and to implement correctly. It provides collected wisdom on how to overcome complexity in many critical areas, and forms a sound basis for many of the necessary fundamental components and concepts. Also, much of what is described here is well chosen, because it has survived the test of time.

Paul has a remarkably diverse background, which is reflected in the content of this book. He is one of the special people who has made major contributions to the literature in multiple areas: computer/network security (including certification-based system architectures, authentication, alternatives to passwords, Internet infrastructure, protocols, and misuse detection), applied cryptography (e.g., public-key infrastructure, enhanced authentication), software, and system usability. Much of his work crosses over all four of these areas, which are mirrored in the topics in this book. Paul has extensive background in both academia and the software industry. His earlier *Handbook of Applied Cryptography*

is highly regarded and widely used, and also reflects the cross-disciplinary thinking that went into writing *Tools and Jewels*.

The balance of Paul's academic and practical experiences is demonstrated by the somewhat unusual organization of this book. It takes a fresh view of each chapter's content, and focuses primarily on precisely what he believes should be taught in a first course on the subject.

To avoid readers expecting something else, I summarize what the book intentionally does *not* attempt to do. It does *not* seek to be a cookbook on how to build or integrate systems and networks that are significantly more secure than what is common today, partly because there is no one-size-fits-all knowledge; there are just too many alternative design and implementation choices. It also does *not* deal with computer architecture in the large—beginning with total-system requirements for hardware, software, and applications. However, it *is* oriented toward practical application rather than specific research results; nevertheless, it cites many important items in the literature. Understanding everything in this book is an essential precursor to achieving meaningfully trustworthy systems.

Altogether, Paul's realistic approach and structural organization in this book are likely to provide a very useful early step—particularly for students and emerging practitioners (e.g., toward being a knowledgeable system designer/developer/administrator), but also for computer users interested in a better understanding of what attaining security might entail. The book may also become an excellent starting point for anyone who subsequently wishes to understand the next stages of dealing with complexity, including layered and compositional architectures for achieving total-system trustworthiness in hardware, software, networks, and applications—as well as coping with the pitfalls inherent in system development, or in more wisely using the Internet.

Designing, developing, and using systems with serious requirements for trustworthiness has inherent complexities. Multics is a historical example of a clean-slate system—with new hardware, a new operating system, and a compiler that facilitated taking advantage of the new hardware features—that is worth studying as a major step forward into secure multi-access computing. Paul begins to invoke some of its architecture (e.g., in Chapter 5). But the real lessons from Multics may be its highly principled design and development process, which carefully exposed new problems and then resolved them with a long view of the future (e.g., completely avoiding stack buffer overflows and the Y2K problem in 1965!). Paul noted to me that if we don't teach anything about strong features of our old systems, how can we make progress?

Chapter 6 provides an example of the book's pragmatic focus, discussing software security and related vulnerabilities—headlined still, sadly, by buffer overruns (mentioned above). It provides background on a selection of well-known exploits currently in use, exposing the importance of software security. A main focus there, and throughout the book, is on helping readers understand what goes wrong, and how. This puts them in a position to appreciate the need for solutions, and motivates pursuit of further details in additional sources, e.g., using references in the chapter End Notes and occasional inline Exercises (and perhaps follow-up courses). This style allows instructors and readers to pick and choose elements to drill deeper, according to personal interests and time constraints.

Such an approach, and the book's pervasive focus on principles, match my own interests quite well, e.g., including my ongoing involvement in developing the CHERI clean-slate capability-based hardware-software system, a highly principled effort taking advantage of the past history of computer systems. Among other foci, the CHERI hardware, software, and extended hardware-aware LLVM (low-level virtual machine) compiler specifically help remediate most of the software security issues that are the main focus of Chapter 6—including spatial and temporal memory safety issues, and protection against buffer overflows in C. Paul cites a CHERI reference in his far-reaching epilogue. (CHERI is a joint effort of SRI and the University of Cambridge.)

Each of the other chapters that I have not mentioned specifically here is a valuable contribution by itself. All in all, I believe Paul's timely book will be extremely useful to a wide audience.

Peter G. Neumann
Chief Scientist, SRI International Computer Science Lab,
and moderator of the ACM Risks Forum
July 2019

Preface

Do not write so that you can be understood, write so that you cannot be misunderstood.
–Epictetus, 55-135 AD

Why this book, approach and target audience

This book provides a concise yet comprehensive overview of computer and Internet security, suitable for a one-term introductory course for junior/senior undergrad or first-year graduate students. It is also suitable for self-study by anyone seeking a solid footing in security—including software developers and computing professionals, technical managers and government staff. An overriding focus is on brevity, without sacrificing breadth of core topics or technical detail within them. The aim is to enable a broad understanding in roughly 300 pages. Further prioritization is supported by designating as optional selected content within this. Fundamental academic concepts are reinforced by specifics and examples, and related to applied problems and real-world incidents.

The first chapter provides a gentle overview and 20 design principles for security. The ten chapters that follow aim to provide a framework for understanding computer and Internet security. They regularly refer back to the principles, with supporting examples. These principles are the conceptual counterparts of security-related error patterns that have been recurring in software and system designs for over 50 years.

The book is "elementary" in that it assumes no background in security, but unlike "soft" high-level texts, does not avoid low-level details; instead it selectively dives into fine points for exemplary topics, to concretely illustrate concepts and principles. The book is rigorous in the sense of being technically sound, but avoids both mathematical proofs and lengthy source-code examples that typically make books inaccessible to general audiences. Knowledge of elementary operating system and networking concepts is helpful, but review sections summarize the essential background. For graduate students, inline exercises and supplemental references provided in per-chapter end notes provide a bridge to further topics and a springboard to the research literature; for those in industry and government, pointers are provided to helpful surveys and relevant standards, e.g., documents from the Internet Engineering Task Force (IETF), and the U.S. National Institute of Standards and Technology.

Selection of topics

For a one-term course in computer and network security, what topics should you cover, in what order—and should breadth or technical depth be favored? We provide a roadmap.

A common complaint is the lack of a concise introductory book that provides a broad overview without being superficial. While no one book will meet the needs of all readers, existing books fall short in several ways. Detailed treatments on the latest advances in specialty areas will not be introductory-level. Books that dwell on recent trends rapidly become dated. Others are too long to be useful introductions—instructors who are not subject-area experts, and readers new to the subject, require guidance on what core material to cover, and what to leave for follow-up or special-topic courses. Some books fail at the presentation level (lacking the technical elements required for engineering and computer science students to develop an understanding), while others that provide detailed code-level examples often lack context and background.

Our aim is to address these deficiencies in a balanced way. Our choices of what to include and exclude, and when to provide low-level details vs. high-level overviews, are informed by guidance from peers, personal experience in industrial and academic research, and from teaching computer security and cryptography for 30 years. The presentation style—which some readers may find atypical—reflects the way in which I organize my own thoughts—metaphorically, putting things into boxes with labels. Thus the material is delivered in "modular" paragraphs, most given short titles indicating their main focus. Those familiar with the *Handbook of Applied Cryptography* (1996) will notice similarities. The topics selected also reflect a personal preference of the core content that I would expect of software developers starting out in industry, or junior security researchers. The material herein also corresponds to what I wish had been available in a book to learn from when I myself began in computer and network security many years ago. The soundness of these choices will be revealed in the course of time.

Framework and systematization

My hope is that this book may serve as a framework from which others may build their own tailored courses. Instructors who prefer to teach from their own notes and slide decks may find it helpful to point students to this text as a coherent baseline reference, augmenting its material with their own specialized content. Indeed, individual instructors with special expertise often wish to teach certain topics in greater detail or add extra topics and examples, while other experts will have analogous desires—but expanding different topics. While a single book clearly cannot capture all such material, the present book may serve as a common foundation and framework for such courses—providing instructors a unified overview and basis for further study in security, rather than leave students without a designated textbook.

Why use a book at all, if essentially all of the information can be found online, in pieces? Piecemeal sources leave students with inconsistent terminology, material of widely varying clarity and correctness, and a lack of supporting background (or redundancy thereof) due to sources targeting readers with different backgrounds. This makes learning inefficient for students. In contrast, services provided by a solid introductory text include: context with well-organized background, consistent terminology, content selection and prioritization, appropriate level of detail, and clarity.

A goal consistent with providing a framework is to help systematize knowledge by

carefully chosen and arranged content. Unlike "tidy" subareas such as cryptography, computer and Internet security as a broad discipline is not particularly orderly, and is less well structured—it often more closely resembles an ad hoc collection of vaguely related items and lessons. We collect these over time, and try to organize them into what we call knowledge. Books like this aim to arrive at a more unified understanding of a broad area, and how its subareas are related. Organization of material may help us recognize common techniques, methods and defensive approaches, as well as recurring attack approaches (e.g., middle-person attacks, social engineering). Note that where security lacks absolute rules (such as laws of physics), we fall back on principles. This acknowledges that we do not always have precise, well-defined solutions that can be universally applied.

Length, prioritization and optional sections

A major idea underlying a shorter book, consciously limited in total page count, is to avoid overwhelming novices. Many introductory security textbooks span 600 to 1000 pages or more. These have different objectives, and are typically a poor fit for a one-term course. While offering a wealth of possible topics to choose from, they are more useful as handbooks or encyclopaedias than introductory texts. They leave readers at a loss as to what to skip over without losing continuity, and the delivery of core topics is often split across several chapters.

In contrast, our approach is to organize discussion of major topics in single locations—thereby also avoiding repetition—and to make informed (hard) choices about which topics to cover. We believe that careful organization and informed selection of core material allows us to maintain equal breadth at half the length of comparable books. To accommodate the fact that some instructors will not be able to cover even our page-reduced content, our material is further prioritized by marking (with a double-dagger prefix, "‡") the headings of sections that can be omitted without loss of continuity. Within undaggered sections, this same symbol denotes paragraphs and exercises that we suggest may be omitted on first reading or by time-constrained readers.

Counterintuitively, a longer book is easier to write in that it requires fewer choices by the author on what to omit. We recall the apology of Blaise Pascal (1623–1662) for the length of a letter, indicating that he did not have time to write a shorter one. ("*Je n'ai fait celle-ci plus longue que parce que je n'ai pas eu le loisir de la faire plus courte*.")

Order of chapters, and relationships between them

The order of chapters was finalized after trying several alternatives. While each individual chapter has been written to be largely independent of others (including its own set of references), subsequences of chapters that work well are: (5, 6, 7), (8, 9) and (10, 11). The introduction (Chapter 1) is followed by cryptography (Chapter 2). User authentication (Chapter 3) then provides an easy entrypoint via widely familiar passwords. Chapter 4 is the most challenging for students afraid of math and cryptography; if this chapter is omitted on first pass, Diffie-Hellman key agreement is useful to pick up for later chapters.

Our prioritization of topics is partially reflected by the order of chapters, and sections within them. For example, Chapter 11 includes intrusion detection, and network-based attacks such as session hijacking. Both of these, while important, might be left out by

time-constrained instructors who may instead give priority to, e.g., web security (Chapter 9). These same instructors might choose to include, from Chapter 11, if not denial of service (DoS) in general, perhaps distributed DoS (DDoS) and pharming. This explains in part why we located this material in a later chapter; another reason is that it builds on many earlier concepts, as does Chapter 10. More fundamental concepts appear in earlier chapters, e.g., user authentication (Chapter 3), operating system security (Chapter 5), and malware (Chapter 7). Readers interested in end-user aspects may find the discussion of email and HTTPS applications in Chapter 8 appealing, as well as discussion of usable security and the web (Section 9.8); Information Technology staff may find, e.g., the details of IPsec (Chapter 10) more appealing. Software developers may be especially interested in Chapters 4 (using passwords as authentication keys), 6 (software security including buffer overruns and integer vulnerabilities), and 8 (public-key infrastructure).

Cryptography vs. security course

Our book is intended as a text for a course in computer security, rather than for one split between cryptography and computer security. Chapter 2 provides cryptographic background and details as needed for an introduction to security, assuming that readers do not have prior familiarity with cryptography. Some readers may choose to initially skip Chapter 2, referring back to parts of it selectively as needed, in a "just-in-time" strategy. A few cryptographic concepts are deferred to later chapters to allow in-context introduction (e.g., Lamport hash chains are co-located with one-time passwords in Chapter 3; Diffie-Hellman key agreement is co-located with middle-person attacks and key management in Chapter 4, along with small-subgroup attacks). Chapter 4 gives additional background on crypto protocols, and Chapter 8 also covers trusts models and public-key certificates. The goal of this crypto background is to make the book self-contained, including important concepts for developers who may, e.g., need to use crypto toolkits or make use of crypto-based mechanisms in web page design. In our own institution, we use this book for an undergrad course in computer and network security, while applied cryptography is taught in a separate undergrad course (using a different book).

Helpful background

Security is a tricky subject area to learn and to teach, as the necessary background spans a wide variety of areas from operating systems, networks, web architecture and programming languages to cryptography, human factors, and hardware architecture. The present book addresses this by providing mini-reviews of essential background where needed, supplemented by occasional extra sections often placed towards the end of the relevant chapter (so as not to disrupt continuity). As suggested earlier, it is helpful if readers have background comparable to a student mid-way through a computer science or computer engineering program, ideally with exposure to basic programming and standard topics in operating systems and network protocols (data communications). For example, readers will benefit from prior exposure to Unix (Chapter 5), memory layout for a run-time stack and the C programming language (Chapter 6), web technologies such as HTML, HTTP and JavaScript (Chapter 9), and basic familiarity with TCP/IP protocols (for Chapters 10 and 11). A summary of relevant prerequisite material is nonetheless given herein.

Trendy topics vs. foundational concepts

For a course that intends to convey the latest information on fast-moving areas, a web site is better suited than content delivery by a textbook. We have designed this book to avoid quickly becoming obsolete, by providing a solid foundation with focus on long-standing concepts, principles, and recurring error patterns. Perhaps surprisingly, a great many security-related errors are invariant over time, independent of evolving platforms. For this same reason, we find it instructive to teach about old flaws that are now fixed (temporarily, in specific products and protocols), as lessons to help us avoid repeating these errors on each new platform. (Mistakes are inevitably repeated, but this way we at least have names for them and can look up the best repair ideas from earlier.)

When teaching, we also rely on a complementary set of hands-on assignments and software tools; these may change as operating systems and technologies evolve. Many excellent, independent online resources are available for programming and tool-based security assignments; we do not attempt to tie hands-on assignments directly to this book itself. The "Exercises" interspersed throughout our chapters herein are not programming-based, but rather are simple review-type thought experiments or more frequently, pointers into the literature for additional material suggested as interesting next-step topics (albeit non-essential to a basic introduction). Along with references provided in the chapter end notes, these are suitable for strong undergrads or more commonly, beginning graduate students who seek exposure one level deeper or broader.

Acknowledgements

This book has benefited immensely from the thoughtful feedback of many colleagues and students. Among many others too numerous to mention, I wish to thank in particular:

AbdelRahman Abdou	Aleks Essex	M. Mannan	Reza Samanfar
Nicholas Akinyokun	Hemant Gupta	Ashraf Matrawy	R. Sekar
Furkan Alaca	Urs Hengartner	Vaclav (Vashek) Matyas	Anil Somayaji
David Barrera	Cormac Herley	Lee McCallum	Elizabeth Stobert
Chris Bellman	Jason Hinek	Fabian Monrose	Tao Wan
Sonia Chiasson	James Huang	Peter G. Neumann	Dave L. Whyte
Peter Choynowski	Trent Jaeger	Bryan Parno	Glenn Wurster
Wayne Du	Poe Kgengwenyane	Joel Reardon	Tatu Ylönen
Will Enck	Hassan Khan	Vladimir Robotka	Lianying (Viau) Zhao

I would also like to thank Ronan Nugent at Springer for his utmost professionalism, and for making the publication process a true pleasure. Errors that remain are of course my own. I would appreciate any feedback (with details to allow corrections), and welcome all suggestions for improvement.

Paul C. van Oorschot
Ottawa, Canada
September 2019

Typesetting Conventions

Conventions used in this book for text fonts, coloring, and styles include the following.

	Examples
paragraph labels	INLINE HOOKING.
headings for examples	**Example** *(WannaCrypt 2017).*
headings for exercises	**Exercise** (Free-lunch attack tree).
emphasis (often regular words or phrases)	*This computer is secure*
technical terms	*passcode generators*
security principle (label for easy cross-reference)	LEAST-PRIVILEGE (P6)
software systems, tools or programs	Linux, Firefox
security incidents or malware name	*Morris worm*, *Code Red II*
filesystem pathnames and filenames	*/usr/bin/passwd*, *chmod.exe*
system function or library calls	execve(), malloc
command-line utilities (as user commands)	**passwd**
OS data structures, flag bits, account names	`inode`, `setuid`, `root`
computer input, output, code or URL	`ls -al`, `http://domain.com`