Preface

HORTENSIO: Madam, before you touch the instrument
To learn the order of my fingering,
I must begin with rudiments of art
To teach you gamouth in a briefer sort,
More pleasant, pithy and effectual,
Than hath been taught by any of my trade;
And there it is in writing, fairly drawn.
—*The Taming of the Shrew,* III, i, 62–68.

On September 11, 2001, terrorists seized control of four airplanes. Three were flown into buildings, and a fourth crashed, with catastrophic loss of life. In the aftermath, the security and reliability of many aspects of society drew renewed scrutiny. One of these aspects was the widespread use of computers and their interconnecting networks.

The issue is not new. In 1988, approximately 5,000 computers throughout the Internet were rendered unusable within 4 hours by a program called a *worm* [432].[1] While the spread, and the effects, of this program alarmed computer scientists, most people were not worried because the worm did not affect their lives or their ability to do their jobs. In 1993, more users of computer systems were alerted to such dangers when a set of programs called *sniffers* were placed on many computers run by network service providers and recorded login names and passwords [374].

After an attack on Tsutomu Shimomura's computer system, and the fascinating way Shimomura followed the attacker's trail, which led to his arrest [914], the public's interest and apprehension were finally aroused. Computers were now vulnerable. Their once reassuring protections were now viewed as flimsy.

Several films explored these concerns. Movies such as *War Games* and *Hackers* provided images of people who can, at will, wander throughout computers and networks, maliciously or frivolously corrupting or destroying information it may have taken millions of dollars to amass. (Reality intruded on *Hackers* when the World Wide Web page set up by MGM/United Artists was quickly altered to present an irreverent commentary on the movie and to suggest that viewers see *The Net*

[1] Section 22.4 discusses computer worms.

instead. Paramount Pictures denied doing this [448].) Another film, *Sneakers*, presented a picture of those who test the security of computer (and other) systems for their owners and for the government.

Goals

This book has three goals. The first is to show the importance of theory to practice and of practice to theory. All too often, practitioners regard theory as irrelevant and theoreticians think of practice as trivial. In reality, theory and practice are symbiotic. For example, the theory of covert channels, in which the goal is to limit the ability of processes to communicate through shared resources, provides a mechanism for evaluating the effectiveness of mechanisms that confine processes, such as sandboxes and firewalls. Similarly, business practices in the commercial world led to the development of several security policy models such as the Clark-Wilson model and the Chinese Wall model. These models in turn help the designers of security policies better understand and evaluate the mechanisms and procedures needed to secure their sites.

The second goal is to emphasize that computer security and cryptography are different. Although cryptography is an essential component of computer security, it is by no means the only component. Cryptography provides a mechanism for performing specific functions, such as preventing unauthorized people from reading and altering messages on a network. However, unless developers understand the context in which they are using cryptography, and unless the assumptions underlying the protocol and the cryptographic mechanisms apply to the context, the cryptography may not add to the security of the system. The canonical example is the use of cryptography to secure communications between two low-security systems. If only trusted users can access the two systems, cryptography protects messages in transit. But if untrusted users can access either system (through authorized accounts or, more likely, by breaking in), the cryptography is not sufficient to protect the messages. The attackers can read the messages at either endpoint.

The third goal is to demonstrate that computer security is not just a science but also an art. It is an art because no system can be considered secure without an examination of how it is to be used. The definition of a "secure computer" necessitates a statement of requirements and an expression of those requirements in the form of authorized actions and authorized users. (A computer engaged in work at a university may be considered "secure" for the purposes of the work done at the university. When moved to a military installation, that same system may not provide sufficient control to be deemed "secure" for the purposes of the work done at that installation.) How will people, as well as other computers, interact with the computer system? How clear and restrictive an interface can a designer create without rendering the system unusable while trying to prevent unauthorized use or access to the data or resources on the system?

Just as an artist paints his view of the world onto canvas, so does a designer of security features articulate his view of the world of human/machine interaction in the security policy and mechanisms of the system. Two designers may use entirely different designs to achieve the same creation, just as two artists may use different subjects to achieve the same concept.

Computer security is also a science. Its theory is based on mathematical constructions, analyses, and proofs. Its systems are built in accordance with the accepted practices of engineering. It uses inductive and deductive reasoning to examine the security of systems from key axioms and to discover underlying principles. These scientific principles can then be applied to untraditional situations and new theories, policies, and mechanisms.

Philosophy

Key to understanding the problems that exist in computer security is a recognition that the problems are not new. They are old problems, dating from the beginning of computer security (and, in fact, arising from parallel problems in the noncomputer world). But the locus has changed as the field of computing has changed. Before the mid-1980s, mainframe and mid-level computers dominated the market, and computer security problems and solutions were phrased in terms of securing files or processes on a single system. With the rise of networking and the Internet, the arena has changed. Workstations and servers, and the networking infrastructure that connects them, now dominate the market. Computer security problems and solutions now focus on a networked environment. However, if the workstations and servers, and the supporting network infrastructure, are viewed as a single system, the models, theories, and problem statements developed for systems before the mid-1980s apply equally well to current systems.

As an example, consider the issue of assurance. In the early period, assurance arose in several ways: formal methods and proofs of correctness, validation of policy to requirements, and acquisition of data and programs from trusted sources, to name a few. Those providing assurance analyzed a single system, the code on it, and the sources (vendors and users) from which the code could be acquired to ensure that either the sources could be trusted or the programs could be confined adequately to do minimal damage. In the later period, the same basic principles and techniques apply, except that the scope of some has been greatly expanded (from a single system and a small set of vendors to the world-wide Internet). The work on proof-carrying code, an exciting development in which the proof that a downloadable program module satisfies a stated policy is incorporated into the program itself,[2] is an example of this expansion. It extends the notion of a proof of consistency with a stated policy. It advances the technology of the earlier period into the later period. But in order to

[2] Section 22.7.5.1 discusses proof-carrying code.

understand it properly, one must understand the ideas underlying the concept of proof-carrying code, and these ideas lie in the earlier period.

As another example, consider Saltzer and Schroeder's principles of secure design.[3] Enunciated in 1975, they promote simplicity, confinement, and understanding. When security mechanisms grow too complex, attackers can evade or bypass them. Many programmers and vendors are learning this when attackers break into their systems and servers. The argument that the principles are old, and somehow outdated, rings hollow when the result of their violation is a nonsecure system.

The work from the earlier period is sometimes cast in terms of systems that no longer exist and that differ in many ways from modern systems. This does not vitiate the ideas and concepts, which also underlie the work done today. Once these ideas and concepts are properly understood, applying them in a multiplicity of environments becomes possible. Furthermore, the current mechanisms and technologies will become obsolete and of historical interest themselves as new forms of computing arise, but the underlying principles will live on, to underlie the next generation—indeed the next era—of computing.

The philosophy of this book is that certain key concepts underlie all of computer security, and that the study of all parts of computer security enriches the understanding of all parts. Moreover, critical to an understanding of the applications of security-related technologies and methodologies is an understanding of the theory underlying those applications.

Advances in the theory of computer protection have illuminated the foundations of security systems. Issues of abstract modeling, and modeling to meet specific environments, lead to systems designed to achieve a specific and rewarding goal. Theorems about composability of policies[4] and the undecidability of the general security question[5] have indicated the limits of what can be done. Much work and effort are continuing to extend the borders of those limits.

Application of these results has improved the quality of the security of the systems being protected. However, the issue is how compatibly the assumptions of the model (and theory) conform to the environment to which the theory is applied. Although our knowledge of how to apply these abstractions is continually increasing, we still have difficulty correctly transposing the relevant information from a realistic setting to one in which analyses can then proceed. Such abstraction often eliminates vital information. The omitted data may pertain to security in nonobvious ways. Without this information, the analysis is flawed.

The practitioner needs to know both the theoretical and practical aspects of the art and science of computer security. The theory demonstrates what is possible. The practical makes known what is feasible. The theoretician needs to understand the constraints under which these theories are used, how their results are translated into practical tools and methods, and how realistic are the assumptions underlying the theories. *Computer Security: Art and Science* tries to meet these needs.

---

[3] Chapter 13 discusses these principles.

[4] See Chapter 8, "Noninterference and Policy Composition."

[5] See Section 3.2, "Basic Results."

Unfortunately, no single work can cover all aspects of computer security, so this book focuses on those parts that are, in the author's opinion, most fundamental and most pervasive. The mechanisms exemplify the applications of these principles.

Organization

The organization of this book reflects its philosophy. It begins with mathematical fundamentals and principles that provide boundaries within which security can be modeled and analyzed effectively. The mathematics provides a framework for expressing and analyzing the requirements of the security of a system. These policies constrain what is allowed and what is not allowed. Mechanisms provide the ability to implement these policies. The degree to which the mechanisms correctly implement the policies, and indeed the degree to which the policies themselves meet the requirements of the organizations using the system, are questions of assurance. Exploiting failures in policy, in implementation, and in assurance comes next, as well as mechanisms for providing information on the attack. The book concludes with the applications of both theory and policy focused on realistic situations. This natural progression emphasizes the development and application of the principles existent in computer security.

Part 1, "Introduction," describes what computer security is all about and explores the problems and challenges to be faced. It sets the context for the remainder of the book.

Part 2, "Foundations," deals with basic questions such as how "security" can be clearly and functionally defined, whether or not it is realistic, and whether or not it is decidable. If it is decidable, under what conditions is it decidable, and if not, how must the definition be bounded in order to make it decidable?

Part 3, "Policy," probes the relationship between policy and security. The definition of "security" depends on policy. In Part 3 we examine several types of policies, including the ever-present fundamental questions of trust, analysis of policies, and the use of policies to constrain operations and transitions.

Part 4, "Implementation I: Cryptography," discusses cryptography and its role in security. It focuses on applications and discusses issues such as key management and escrow, key distribution, and how cryptosystems are used in networks. A quick study of authentication completes Part 4.

Part 5, "Implementation II: Systems," considers how to implement the requirements imposed by policies using system-oriented techniques. Certain design principles are fundamental to effective security mechanisms. Policies define who can act and how they can act, and so identity is a critical aspect of implementation. Mechanisms implementing access control and flow control enforce various aspects of policies.

Part 6, "Assurance," presents methodologies and technologies for ascertaining how well a system, or a product, meets its goals. After setting the background, to explain exactly what "assurance" is, the art of building systems to meet varying levels

of assurance is discussed. Formal verification methods play a role. Part 6 shows how the progression of standards has enhanced our understanding of assurance techniques.

Part 7, "Special Topics," discusses some miscellaneous aspects of computer security. Malicious logic thwarts many mechanisms. Despite our best efforts at high assurance, systems today are replete with vulnerabilities. Why? How can a system be analyzed to detect vulnerabilities? What models might help us improve the state of the art? Given these security holes, how can we detect attackers who exploit them? A discussion of auditing flows naturally into a discussion of intrusion detection—a detection method for such attacks.

Part 8, "Practicum," presents examples of how to apply the principles discussed throughout the book. It begins with networks and proceeds to systems, users, and programs. Each chapter states a desired policy and shows how to translate that policy into a set of mechanisms and procedures that support the policy. Part 8 tries to demonstrate that the material covered elsewhere can be, and should be, used in practice.

Each chapter in this book ends with a summary, descriptions of some research issues, and some suggestions for further reading. The summary highlights the important ideas in the chapter. The research issues are current "hot topics" or are topics that may prove to be fertile ground for advancing the state of the art and science of computer security. Interested readers who wish to pursue the topics in any chapter in more depth can go to some of the suggested readings. They expand on the material in the chapter or present other interesting avenues.

Roadmap

This book is both a reference book and a textbook. Its audience is undergraduate and graduate students as well as practitioners. This section offers some suggestions on approaching the book.

Dependencies

Chapter 1 is fundamental to the rest of the book and should be read first. After that, however, the reader need not follow the chapters in order. Some of the dependencies among chapters are as follows.

Chapter 3 depends on Chapter 2 and requires a fair degree of mathematical maturity. Chapter 2, on the other hand, does not. The material in Chapter 3 is for the most part not used elsewhere (although the existence of the first section's key result, the undecidability theorem, is mentioned repeatedly). It can be safely skipped if the interests of the reader lie elsewhere.

The chapters in Part 3 build on one another. The formalisms in Chapter 5 are called on in Chapters 19 and 20, but nowhere else. Unless the reader intends to delve into the sections on theorem proving and formal mappings, the formalisms may be

skipped. The material in Chapter 8 requires a degree of mathematical maturity, and this material is used sparingly elsewhere. Like Chapter 3, Chapter 8 can be skipped by the reader whose interests lie elsewhere.

Chapters 9, 10, and 11 also build on one another in order. A reader who has encountered basic cryptography will have an easier time with the material than one who has not, but the chapters do not demand the level of mathematical experience that Chapters 3 and 8 require. Chapter 12 does not require material from Chapter 10 or Chapter 11, but it does require material from Chapter 9.

Chapter 13 is required for all of Part 5. A reader who has studied operating systems at the undergraduate level will have no trouble with Chapter 15. Chapter 14 uses the material in Chapter 11; Chapter 16 builds on material in Chapters 5, 13, and 15; and Chapter 17 uses material in Chapters 4, 13, and 16.

Chapter 18 relies on information in Chapter 4. Chapter 19 builds on Chapters 5, 13, 15, and 18. Chapter 20 presents highly mathematical concepts and uses material from Chapters 18 and 19. Chapter 21 is based on material in Chapters 5, 18, and 19; it does not require Chapter 20. For all of Part 5, a knowledge of software engineering is very helpful.

Chapter 22 draws on ideas and information in Chapters 5, 6, 9, 13, 15, and 17 (and for Section 22.6, the reader should read Section 3.1). Chapter 23 is self-contained, although it implicitly uses many ideas from assurance. It also assumes a good working knowledge of compilers, operating systems, and in some cases networks. Many of the flaws are drawn from versions of the UNIX operating system, or from Windows systems, and so a working knowledge of either or both systems will make some of the material easier to understand. Chapter 24 uses information from Chapter 4, and Chapter 25 uses material from Chapter 24.

The practicum chapters are self-contained and do not require any material beyond Chapter 1. However, they point out relevant material in other sections that augments the information and (we hope) the reader's understanding of that information.

Background

The material in this book is at the advanced undergraduate level. Throughout, we assume that the reader is familiar with the basics of compilers and computer architecture (such as the use of the program stack) and operating systems. The reader should also be comfortable with modular arithmetic (for the material on cryptography). Some material, such as that on formal methods (Chapter 20) and the mathematical theory of computer security (Chapter 3 and the formal presentation of policy models), requires considerable mathematical maturity. Other specific recommended background is presented in the preceding section. Part 9, "End Matter," contains material that will be helpful to readers with backgrounds that lack some of the recommended material.

Examples are drawn from many systems. Many come from the UNIX operating system or variations of it (such as Linux). Others come from the Windows family of systems. Familiarity with these systems will help the reader understand many examples easily and quickly.

Undergraduate Level

An undergraduate class typically focuses on applications of theory and how students can use the material. The specific arrangement and selection of material depends on the focus of the class, but all classes should cover some basic material—notably that in Chapters 1, 9, and 13, as well as the notion of an access control matrix, which is discussed in Sections 2.1 and 2.2.

Presentation of real problems and solutions often engages undergraduate students more effecively than presentation of abstractions. The special topics and the practicum provide a wealth of practical problems and ways to deal with them. This leads naturally to the deeper issues of policy, cryptography, noncryptographic mechanisms, and assurance. The following are sections appropriate for nonmathematical undergraduate courses in these topics.

- *Policy:* Sections 4.1 through 4.4 describe the notion of policy. The instructor should select one or two examples from Sections 5.1, 5.2.1, 6.2, 6.4, 7.1.1, and 7.2, which describe several policy models informally. Section 7.4 discusses role-based access control.
- *Cryptography:* Key distribution is discussed in Sections 10.1 and 10.2, and a common form of public key infrastructures (called PKIs) is discussed in Section 10.4.2. Section 11.1 points out common errors in using cryptography. Section 11.3 shows how cryptography is used in networks, and the instructor should use one of the protocols in Section 11.4 as an example. Chapter 12 offers a look at various forms of authentication, including noncryptographic methods.
- *Noncryptographic mechanisms:* Identity is the basis for many access control mechanisms. Sections 14.1 through 14.4 discuss identity on a system, and Section 14.6 discusses identity and anonymity on the Web. Sections 15.1 and 15.2 explore two mechanisms for controlling access to files, and Section 15.4 discusses the ring-based mechanism underlying the notion of multiple levels of privilege. If desired, the instructor can cover sandboxes by using Sections 17.1 and 17.2, but because Section 17.2 uses material from Sections 4.5 and 4.5.1, the instructor will need to go over those sections as well.
- *Assurance:* Chapter 18 provides a basic introduction to the often overlooked topic of assurance.

Graduate Level

A typical introductory graduate class can focus more deeply on the subject than can an undergraduate class. Like an undergraduate class, a graduate class should cover Chapters 1, 9, and 13. Also important are the undecidability results in Sections 3.1 and 3.2, which require that Chapter 2 be covered. Beyond that, the instructor can

choose from a variety of topics and present them to whatever depth is appropriate. The following are sections suitable for graduate study.

- *Policy models:* Part 3 covers many common policy models both informally and formally. The formal description is much easier to understand once the informal description is understood, so in all cases both should be covered. The controversy in Section 5.4 is particularly illuminating to students who have not considered the role of policy and the nature of a policy. Chapter 8 is a highly formal discussion of the foundations of policy and is appropriate for students with experience in formal mathematics. Students without such a background will find it quite difficult.
- *Cryptography:* Part 4 focuses on the applications of cryptography, not on cryptography's mathematical underpinnings.[6] It discusses areas of interest critical to the use of cryptography, such as key management and some basic cryptographic protocols used in networking.
- *Noncryptographic mechanisms:* Issues of identity and certification are complex and generally poorly understood. Section 14.5 covers these problems. Combining this with the discussion of identity on the Web (Section 14.6) raises issues of trust and naming. Chapters 16 and 17 explore issues of information flow and confining that flow.
- *Assurance:* Traditionally, assurance is taught as formal methods, and Chapter 20 serves this purpose. In practice, however, assurance is more often accomplished by using structured processes and techniques and informal but rigorous arguments of justification, mappings, and analysis. Chapter 19 emphasizes these topics. Chapter 21 discusses evaluation standards and relies heavily on the material in Chapters 18 and 19 and some of the ideas in Chapter 20.
- *Miscellaneous Topics:* Section 22.6 presents a proof that the generic problem of determining if a generic program is a computer virus is in fact undecidable. The theory of penetration studies in Section 23.2, and the more formal approach in Section 23.5, illuminate the analysis of systems for vulnerabilities. If the instructor chooses to cover intrusion detection (Chapter 25) in depth, it should be understood that this discussion draws heavily on the material on auditing (Chapter 24).
- *Practicum:* The practicum (Part 8) ties the material in the earlier part of the book to real-world examples and emphasizes the applications of the theory and methodologies discussed earlier.

[6] The interested reader will find a number of books covering aspects of this subject [240, 588, 693, 700, 885, 894, 995].

Practitioners

Practitioners in the field of computer security will find much to interest them. The table of contents and the index will help them locate specific topics. A more general approach is to start with Chapter 1 and then proceed to Part 8, the practicum. Each chapter has references to other sections of the text that explain the underpinnings of the material. This will lead the reader to a deeper understanding of the reasons for the policies, settings, configurations, and advice in the practicum. This approach also allows readers to focus on those topics that are of most interest to them.

Special Acknowledgment

Elisabeth Sullivan contributed the assurance part of this book. She wrote several drafts, all of which reflect her extensive knowledge and experience in that aspect of computer security. I am particularly grateful to her for contributing her real-world knowledge of how assurance is managed. Too often, books recount the mathematics of assurance without recognizing that other aspects are equally important and more widely used. These other aspects shine through in the assurance section, thanks to Liz. As if that were not enough, she made several suggestions that improved the policy part of this book. I will always be grateful for her contribution, her humor, and especially her friendship.

Acknowledgments

Many people have contributed to this book. Peter Salus' suggestion first got me thinking about writing it, and Peter put me in touch with Addison-Wesley. Midway through the writing, Blaine Burnham reviewed the completed portions and the proposed topics, and suggested that they be reorganized in several ways. The current organization of the book grew from his suggestions. Marvin Schaefer reviewed parts of the book with a keen eye, made suggestions that improved many parts, and encouraged me at the end. I thank these three for their contributions.

Many others contributed to this book in various ways. Special thanks to Jim Alves-Foss, Bill Arbaugh, Andrew Arcilla, Rebecca Bace, Belinda Bashore, Terry Brugger, Michael Clifford, Crispin Cowan, Dimitri DeFigueiredo, Jeremy Frank, Robert Fourney, Ron Gove, Jesper Johansson, Calvin Ko, Karl Levitt, Gary McGraw, Alexander Meau, Nasir Memon, Mark Morrissey, Stephen Northcutt, Holly Pang, Sung Park, Ashwini Raina, Brennen Reynolds, Christoph Schuba, Jonathan Shapiro, Clay Shields, Tom Walcott, Dan Watson, and Chris Wee, and to everyone in my computer security classes, who (knowingly or unknowingly) helped me develop and test this material.

The Addison-Wesley folks, Kathleen Billus, Susannah Buzard, Bernie Gaffney, Amy Fleischer, Helen Goldstein, Tom Stone, Asdis Thorsteinsson, and most especially my editor, Peter Gordon, were incredibly patient and helpful, despite fears that this book would never materialize. The fact that it did so is in great measure attributable to their hard work and encouragement. I also thank the production people at Argosy, especially Beatriz Valdés and Craig Kirkpatrick, for their wonderful work.

Dorothy Denning, my advisor in graduate school, guided me through the maze of computer security when I was just beginning. Peter Denning, Barry Leiner, Karl Levitt, Peter Neumann, Marvin Schaefer, Larry Snyder, and several others influenced my approach to the subject. I hope this work reflects in some small way what they gave to me and passes a modicum of it along to my readers.

I also thank my parents, Leonard Bishop and Linda Allen. My father, a writer, gave me some useful tips on writing, which I tried to follow. My mother, a literary agent, helped me understand the process of getting the book published, and supported me throughout.

Finally, I would like to thank my family for their support throughout the writing. Sometimes they wondered if I would ever finish. My wife Holly and our children Steven, David, and Caroline were very patient and understanding and made sure I had time to work on the book. Our oldest daughter Heidi and her husband Mike also provided much love and encouragement and the most wonderful distraction: our grandson—Skyler. To all, my love and gratitude.