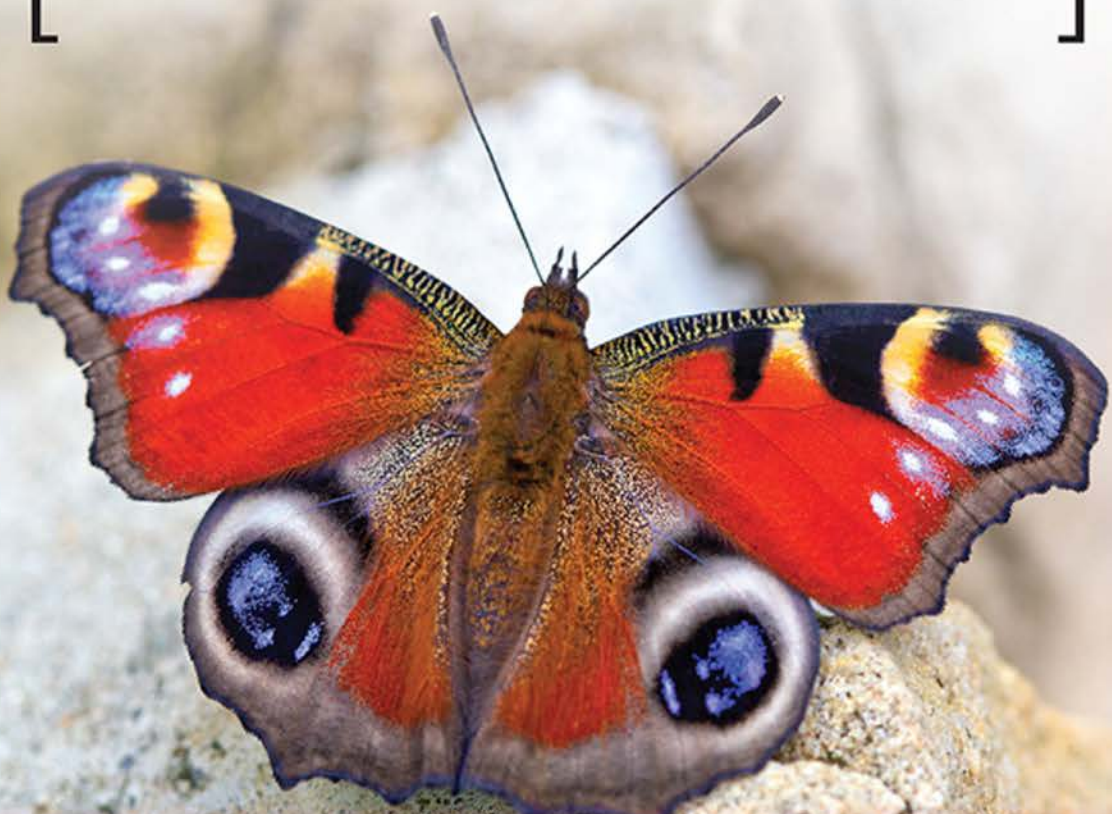
SECOND EDITION

COMPUTER SECURITY

[ART and SCIENCE]



MATT BISHOP

With contributions from ELISABETH SULLIVAN *and* MICHELLE RUPPEL

Computer Security

Second Edition

This page intentionally left blank

Computer Security

Art and Science

Second Edition

Matt Bishop

with contributions from

Elisabeth Sullivan and Michelle Ruppel

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2018950017

Cover illustration: Captiva55/Shutterstock

ISBN-13: 978-0-321-71233-2
ISBN-10: 0-321-71233-1

ScoutAutomatedPrintCode

To my dear Holly; our children Heidi, Steven, David, and Caroline; our grandchildren Skyler and Sage; and our friends Seaview, Tinker Belle, Stripe, Baby Windsor, Scout, Fur, Puff, Mouse, Shadow, Fuzzy, Dusty, and the rest of the menagerie.

This page intentionally left blank

Contents

This page intentionally left blank

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.

Preface to the Second Edition

Since the first edition of this book was published, the number of computer and information security incidents has increased dramatically, as has their seriousness. In 2010, a computer worm infected the software controlling a particular type of centrifuge used in uranium-enrichment sites [1116, 1137]. In 2013, a security breach at Target, a large chain of stores in the United States, compromised 40 million credit cards [1497, 1745, 2237]. Also in 2013, Yahoo reported that an attack compromised more than 1 billion accounts [779]. In 2017, attackers spread ransomware that crippled computers throughout the world, including computers used in hospitals and telecommunications companies [1881]. Equifax estimated that attackers also compromised the personal data of over 100,000,000 people [176].

These attacks exploit vulnerabilities that have their roots in vulnerabilities of the 1980s, 1970s, and earlier. They seem more complex because systems have become more complex, and thus the vulnerabilities are more obscure and require more complex attacks to exploit. But the principles underlying the attacks, the vulnerabilities, and the failures of the systems have not changed—only the arena in which they are applied has.

Consistent with this philosophy, the second edition continues to focus on the principles underlying the field of computer and information security. Many

newer examples show how these principles are applied, or not applied, today; but the principles themselves are as important today as they were in 2002, and earlier. Some have been updated to reflect a deeper understanding of people and systems. Others have been applied in new and interesting ways. But they still ring true.

That said, the landscape of security has evolved greatly in the years since this book was first published. The explosive growth of the World Wide Web, and the consequent explosion in its use, has made security a problem at the forefront of our society. No longer can vulnerabilities, both human and technological, be relegated to the background of our daily lives. It is one of the elements at the forefront, playing a role in everyone's life as one browses the web, uses a camera to take and send pictures, and turns on an oven remotely. We grant access to our personal lives through social media such as Facebook, Twitter, and Instagram, and to our homes through the Internet of Things and our connections to the Internet. To ignore security issues, or consider them simply ancillary details that "someone will fix somehow" or threats unlikely to be realized personally is dangerous at best, and potentially disastrous at worst.

Ultimately, little has changed. The computing ecosystem of our day is badly flawed. Among the manifestations of these technological flaws are that security problems continue to exist, and continue to grow in magnitude of effect. An interesting question to ponder is what might move the paradigm of security away from the cycle of "patch and catch" and "let the buyer beware" to a stable and safer ecosystem.

But we must continue to improve our understanding of, and implementation of, security. Security nihilism—simply giving up and asserting that we cannot make things secure, so why try—means we accept these invasions of our privacy, our society, and our world. Like everything else, security is imperfect, and always will be—meaning we can improve the state of the art. This book is directed towards that goal.

Updated Roadmap

The dependencies of the chapters are the same as in the first edition (see p. xl), with two new chapters added.

Chapter 7, which includes a discussion of denial of service attack models, contains material useful for Chapters 23, 24, 27, and 28. Similarly, Chapter 27 draws on material from the chapters in Part III as well as Chapters 23, 25, 26, and all of Part VIII.

In addition to the suggestions in the preface to the first edition on p. xli about topics for undergraduate classes, the material in Chapter 27 will introduce undergraduates to how attacks occur, how they can be analyzed, and what their effects are. Coupled with current examples drawn from the news, this chapter should prove fascinating to undergraduates.

As for graduate classes, the new material in Chapter 7 will provide students with some background on resilience, a topic increasing in importance. Otherwise, the recommendations are the same as for the first edition (see p. xlii).

Changes to the First Edition

The second edition has extensively revised many examples to apply the concepts to technologies, methodologies, and ideas that have emerged since the first edition was published. Here, the focus is on new material in the chapters; changes to examples are mentioned only when necessary to describe that material. In addition to what is mentioned here, much of the text has been updated.

Chapter 1, "An Overview of Computer Security": This chapter is largely unchanged.

Chapter 2, "Access Control Matrix": Section 2.2.2, "Access Controlled by History" has been changed to use the problem of preventing downloaded programs from accessing the system in unauthorized ways, instead of updating a database. Section 2.4.3, "Principle of Attenuation of Privilege," has been expanded slightly, and exercises added to point out differing forms of the principle.

Chapter 3, "Foundational Results": Definition 3–1 has been updated to make clear that "leaking" refers to a right being added to an element of the access control matrix that did not contain it initially, and an exercise has been added to demonstrate the difference between this definition and the one in the first edition. Section 3.6 discusses comparing security properties of models.

Chapter 4, "Security Policies": Section 4.5.1, "High-Level Policy Languages," now uses Ponder rather than a Java policy constraint language. Section 4.6, "Example: Academic Computer Security Policy," has been updated to reflect changes in the university policy.

Chapter 5, "Confidentiality Policies": Section 5.3.1 discusses principles for declassifying information.

Chapter 6, "Integrity Policies": Section 6.5 presents trust models.

Chapter 7, "Availability Policies": This chapter is new.

Chapter 8, "Hybrid Policies": Section 8.1.3 modifies one of the assumptions of the Chinese Wall model that is unrealistic. Section 8.3.1 expands the discussion of ORCON to include DRM. Section 8.4 adds a discussion of several types of RBAC models.

Chapter 9, "Noninterference and Policy Composition": This chapter adds Section 9.6, which presents side channels in the context of deducibility.

Chapter 10, "Basic Cryptography": This chapter has been extensively revised. The discussion of the DES (Section 10.2.3) has been tightened and the algorithm

moved to Appendix F. Discussions of the AES (Section 10.2.5) and elliptic curve cryptography (Section 10.3.3) have been added, and the section on digital signatures moved from Chapter 11 to Section 10.5. Also, the number of digits in the integers used in examples for public key cryptography has been increased from 2 to at least 4, and in many cases more.

Chapter 11, "Key Management": Section 11.4.3 discusses public key infrastructures. Section 11.5.1.4, "Other Approaches," now includes a brief discussion of identity-based encryption.

Chapter 12, "Cipher Techniques": Section 12.1, "Problems," now includes a discussion of type flaw attacks. Section 12.3 discusses authenticated encryption with associated data, and presents the CCM and GCM modes of block ciphers. A new section, Section 12.5.2, discusses the Signal Protocol. Section 12.5.3, "Security at the Transport Layer: TLS and SSL," has been expanded and focuses on TLS rather than SSL. It also discusses cryptographic weaknesses in SSL, such as the POODLE attack, that have led to the use of SSL being strongly discouraged.

Chapter 13, "Authentication": A discussion of graphical passwords has been added as Section 13.3.4. Section 13.4.3 looks at quantifying password strength in terms of entropy. The discussion of biometrics in Section 13.7 has been expanded to reflect their increasing use.

Chapter 14, "Design Principles": The principle of least authority follows the principle of least privilege in Section 14.2.1, and the principle of least astonishment now supersedes the principle of psychological acceptability in Section 14.2.8.

Chapter 15, "Representing Identity": Section 15.5, "Naming and Certificates," now includes a discussion of registration authorities (RAs). Section 15.6.1.3 adds a discussion of the DNS security extensions (DNSSEC). Section 15.7.2 discusses onion routing and Tor in the context of anonymity.

Chapter 16, "Access Control Mechanisms": Section 16.2.6 discusses sets of privileges in Linux and other UNIX-like systems.

Chapter 17, "Information Flow": In contrast to the confidentiality-based context of information flow in the main part of this chapter, Section 17.5 presents information flow in an integrity context. In Section 17.6, the SPI and SNSMG examples of the first edition have been replaced by Android cell phones (Section 17.6.1) and firewalls (Section 17.6.2).

Chapter 18, "Confinement Problem": Section 18.2 has been expanded to include library operating systems (Section 18.2.1.2) and program modification techniques (Section 18.2.2).

Chapter 19, "Introduction to Assurance": Section 19.2.3, which covers agile software development, has been added.

Chapter 20, "Building Systems with Assurance": The example decomposition of Windows 2000 into components has been updated to use Windows 10.

Chapter 21, "Formal Methods": A new section, Section 21.5, discusses functional programming languages, and another new section, 21.6, discusses formally verified products.

Chapter 22, "Evaluating Systems": Sections 22.7, on FIPS 140, and 22.8, on the Common Criteria, have been extensively updated.

Chapter 23, "Malware": Section 23.5 presents botnets, and Sections 23.6.3, 23.6.4, 23.6.5, and 23.6.6 discuss adware and spyware, ransomware, and phishing. While not malware, phishing is a common vector for getting malware onto a system and so it is discussed here.

Chapter 24, "Vulnerability Analysis": Section 24.2.5 reviews several penetration testing frameworks used commercially and based on the Flaw Hypothesis Methodology. Section 24.5 presents the widely used CVE and CWE standards.

Chapter 25, "Auditing": Section 25.3.3, which discusses sanitization, has been expanded.

Chapter 26, "Intrusion Detection": Section 26.3.1 has been expanded to include several widely used machine learning techniques for anomaly detection. Incident response groups are discussed in Section 27.3.

Chapter 27, "Attacks and Responses": This chapter is new.

Chapter 28, "Network Security": The discussion of what firewalls are has been moved to Section 17.6.2, but the discussion of how the Drib configures and uses them remains in this chapter. The Drib added wireless networks, which are discussed in Section 28.3.3.1. Its analysis of using the cloud is in Section 28.3.3.2. The rest of the chapter has been updated to refer to the new material in previous chapters.

Chapter 29, "System Security": This chapter has been updated to refer to the new material in previous chapters.

Chapter 30, "User Security": Section 30.2.2 describes the two-factor authentication procedure used by the Drib. The rest of the chapter has been updated to refer to the new material in previous chapters.

Chapter 31, "Program Security": This chapter has been updated to refer to the new material in previous chapters.

Two new appendices have been added. Appendix F presents the DES and AES algorithms, and Appendix H collects the rules in Chapter 31 for easy reference. In addition, Appendix D examines some hardware enhancements to aid virtualization, and Appendix G contains the full academic security policy discussed in Section 4.6.

Preface to the First Edition[1]

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 [842].[2] 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 [670].

After an attack on Tsutomu Shimomura's computer system, and the fascinating way Shimomura followed the attacker's trail, which led to his arrest [1736], 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* instead. Paramount Pictures denied doing this [869].) 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.

---

[1] Chapter numbers have been updated to correspond to the chapters in the second edition.

[2] Section 23.4 discusses computer worms.

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 non-computer 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, is an example of this expansion.[3] 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 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.[4] 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 non-secure 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

---

[3] Section 23.9.5.1 discusses proof-carrying code.

[4] Chapter 14 discusses these principles.

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[5] and the undecidability of the general security question[6] 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 non-obvious 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.

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

[5] See Chapter 9, "Noninterference and Policy Composition."
[6] See Section 3.2, "Basic Results."

on realistic situations. This natural progression emphasizes the development and application of the principles existent in computer security.

Part I, "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 II, "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 III, "Policy," probes the relationship between policy and security. The definition of "security" depends on policy. In Part III 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 IV, "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 III.

Part V, "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 VI, "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 VI shows how the progression of standards has enhanced our understanding of assurance techniques.

Part VII, "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 VIII, "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 VIII 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 III build on one another. The formalisms in Chapter 5 are called on in Chapters 20 and 21, 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 9 requires a degree of mathematical maturity, and this material is used sparingly elsewhere. Like Chapter 3, Chapter 9 can be skipped by the reader whose interests lie elsewhere.

Chapters 10, 11, and 12 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 9 require. Chapter 13 does not require material from Chapter 11 or Chapter 12, but it does require material from Chapter 10.

Chapter 14 is required for all of Part V. A reader who has studied operating systems at the undergraduate level will have no trouble with Chapter 16. Chapter 15 uses the material in Chapters 10 and 11; Chapter 17 builds on material in Chapters 5, 14, and 16; and Chapter 18 uses material in Chapters 4, 14, and 17.

Chapter 19 relies on information in Chapter 4. Chapter 20 builds on Chapters 5, 14, 16, and 19. Chapter 21 presents highly mathematical concepts and uses material from Chapters 19 and 20. Chapter 22 is based on material in Chapters 5, 19, and 20; it does not require Chapter 21. For all of Part VI, a knowledge of software engineering is very helpful.

Chapter 23 draws on ideas and information in Chapters 5, 6, 10, 14, 16, and 18 (and for Section 23.8, the reader should read Section 3.1). Chapter 24 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 25 uses information from Chapter 4, and Chapter 26 uses material from Chapter 25.

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 21) 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 IX, the appendices, 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, 10, and 14, 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 effectively 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, non-cryptographic mechanisms, and assurance. The following are sections appropriate for non-mathematical 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, 8.1.1, and 8.2, which describe several policy models informally. Section 8.4 discusses role-based access control.
- *Cryptography*: Key distribution is discussed in Sections 11.1 and 11.2, and a common form of public key infrastructures (called PKIs) is discussed in Section 11.4.2. Section 12.1 points out common errors

in using cryptography. Section 12.4 shows how cryptography is used in networks, and the instructor should use one of the protocols in Section 12.5 as an example. Chapter 13 offers a look at various forms of authentication, including non-cryptographic methods.

- *Non-cryptographic mechanisms*: Identity is the basis for many access control mechanisms. Sections 15.1 through 15.4 discuss identity on a system, and Section 15.6 discusses identity and anonymity on the Web. Sections 16.1 and 16.2 explore two mechanisms for controlling access to files, and Section 16.4 discusses the ring-based mechanism underlying the notion of multiple levels of privilege. If desired, the instructor can cover sandboxes by using Sections 18.1 and 18.2, but because Section 18.2 uses material from Section 4.5, the instructor will need to go over those sections as well.
- *Assurance*: Chapter 19 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, 10, and 14. 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 III 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 9 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 IV focuses on the applications of cryptography, not on cryptography's mathematical underpinnings.[7] It discusses areas of interest critical to the use of cryptography, such as key management and some basic cryptographic protocols used in networking.
- *Non-cryptographic mechanisms*: Issues of identity and certification are complex and generally poorly understood. Section 15.5 covers these problems. Combining this with the discussion of identity on the Web

[7]The interested reader will find a number of books covering aspects of this subject [440, 787, 788, 914, 1092, 1093, 1318, 1826].

(Section 15.6) raises issues of trust and naming. Chapters 17 and 18 explore issues of information flow and confining that flow.

- *Assurance*: Traditionally, assurance is taught as formal methods, and Chapter 21 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 20 emphasizes these topics. Chapter 22 discusses evaluation standards and relies heavily on the material in Chapters 19 and 20 and some of the ideas in Chapter 21.
- *Miscellaneous Topics*: Section 23.8 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 24.2, and the more formal approach in Section 24.6, illuminate the analysis of systems for vulnerabilities. If the instructor chooses to cover intrusion detection (Chapter 26) in depth, it should be understood that this discussion draws heavily on the material on auditing (Chapter 25).
- *Practicum*: The practicum (Part VIII) 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.

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 VIII, 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.

This page intentionally left blank

Acknowledgments

It is not possible to separate those who contributed to the second edition from those who contributed to the first edition, because everything done for the first edition, especially after the first printing, has contributed to the second. So these acknowledgments apply to both editions. That said ...

Special Acknowledgments

Elisabeth Sullivan and Michelle Ruppel contributed the assurance part of this book.

For the first edition, Liz 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.

For the second edition, Michelle stepped in to update that part based on her extensive experience and real-world knowledge as a practitioner. She was careful to maintain the tone and style of Liz's writing, and her contributions strengthened the assurance part. I am grateful to her for agreeing to step in, for the exceptional effort she put forth, and the high quality that resulted.

In summary, I am very grateful for their contributions.

Acknowledgments

Many people offered comments, suggestions, and ideas for the second edition. Thanks to Marvin Schaefer, Sean Peisert, Prof. Christian Probst, Carrie Gates, and Richard Ford for their reviews of the various chapters. I appreciate Prof. Ken

Rosen and Prof. Alfred Menezes for their help with Chapter 10, Steven Templeton and Kara Nance for their suggestions on Chapter 27, Karl Levitt for his comments on Chapter 26, and Richard Ford for his many suggestions on Chapter 23. Their advice and suggestions were invaluable in preparing this edition. Of course, any errors in the text are my responsibility, and usually occurred because I did not always follow their advice.

Thanks also to Pasquale Noce, who sent me a thorough analysis of many of the theorems, proving them constructively as opposed to how they were done in the book. He made many other helpful comments and caught some errors.

The students in Peter Reiher's COM SCI 236-80, Computer Security, class at UCLA in the Spring Quarter 2018, and the students in my ECS 153, Computer Security, classes over the past few years at UC Davis used parts of this edition in various stages of preparation. I thank them for their feedback, which also improved the book.

Many others contributed to this book in various ways. Special thanks to Steven Alexander, Amin Alipour, Jim Alves-Foss, Bill Arbaugh, Andrew Arcilla, Alex Aris, Rebecca Bace, Belinda Bashore, Vladimir Berman, Rafael Bhatti, Ziad El Bizri, David Bover, Logan Browne, Terry Brugger, Gordon Burns, Serdar Cabuk, Raymond Centeno, Yang Chen, Yi Chen, HakSoo Choi, Lisa Clark, Michael Clifford, Christopher Clifton, Dan Coming, Kay Connelly, Crispin Cowan, Shayne Czyzewski, Tom Daniels, Dimitri DeFigueiredo, Joseph-Patrick Dib, Till Dörges, Felix Fan, Robert Fourney, Guillermo Francia III, Jeremy Frank, Conny Francke, Martin Gagne, Nina Gholami, Ron Gove, James Hinde, James Hook, Xuxian Jiang, Jesper Johansson, Mark Jones, Calvin Ko, Mark-Neil Ledesma, Ken Levine, Karl Levitt, Luc Longpre, Yunhua Lu, Gary McGraw, Alexander Meau, Nasir Memon, Katherine Moore, Mark Morrissey, Ather Nawaz, Iulian Neamtiu, Dan Nerenburg, Kimberly Nico, Stephen Northcutt, Rafael Obelheiro, Josko Orsulic, Holly Pang, Sean Peisert, Ryan Poling, Sung Park, Ashwini Raina, Jorge Ramos, Brennen Reynolds, Peter Rozental, Christoph Schuba, night SH, David Shambroom, Jonathan Shapiro, Clay Shields, Sriram Srinivasan, Mahesh V. Tripunitara, Vinay Vittal, Tom Walcott, James Walden, Dan Watson, Guido Wedig, Chris Wee, Han Weili, Patrick Wheeler, Paul Williams, Bonnie Xu, Charles Yang, Xiaoduan Ye, Xiaohui Ye, Lara Whelan, John Zachary, Linfeng Zhang, Aleksandr Zingorenko, and to everyone in my and others' computer security classes, who (knowingly or unknowingly) helped me develop and test this material.

The Pearson folks, in particular my editors Laura Lewin and Malobika Chakraborty, and Sheri Replin, were incredibly helpful and patient. Their patience and enthusiasm ensured this second edition was completed, although a bit later than expected. The production people, especially Julie Nahil, Ramya Gangadharan, and Charles Roumeliotis, moved the book smoothly into print, and I thank them for making it as painless as possible. I owe them many thanks. Similarly, for the first edition, 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. My wife Holly, our children Heidi, Steven, David, and Caroline, and grandchildren Skyler and Sage were very patient and understanding and made sure I had time to work on the book. They also provided delightful distractions. To them all, my love and gratitude.

This page intentionally left blank

About the Author

Matt Bishop is a professor in the Department of Computer Science at the University of California at Davis. He received his Ph.D. in computer science from Purdue University, where he specialized in computer security, in 1984. He was a systems programmer at Megatest Corporation, a research scientist at the Research Institute of Advanced Computer Science and was on the faculty at Dartmouth College.

His main research area is the analysis of vulnerabilities in computer systems, including modeling them, building tools to detect vulnerabilities, and ameliorating or eliminating them. This includes detecting and handling all types of malicious logic. He works in the areas of network security, the study of denial of service attacks and defenses, policy modeling, software assurance testing, resilience, and formal modeling of access control. He has worked extensively in electronic voting, was one of the members of the RABA study for Maryland, and was one of the two principle investigators of the California Top-to-Bottom Review, which performed a technical review of all electronic voting systems certified in the State of California.

He is active in information assurance education. He was co-chair of the Joint Task Force that developed the *Cybersecurity Curricula 2017: Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity*, released in December 2017. He teaches introductory programming, software engineering, operating systems, and (of course) computer security.

This page intentionally left blank

Part I

Introduction

Writers say "To write a good book, tell them what you are going to tell them, then tell them, then tell them what you told them." This is the "what we're going to tell you" part.

Chapter 1, "An Overview of Computer Security," presents the underpinnings of computer security and an overview of the important issues to place them in context. It begins with a discussion of what computer security is and how threats are connected to security services. The combination of desired services makes up a policy, and mechanisms enforce the policy. All rely on underlying assumptions, and the systems built on top of these assumptions lead to issues of assurance. Finally, the operational and human factors affect the mechanisms used as well as the policy.

This page intentionally left blank

Chapter 1

An Overview of Computer Security

ANTONIO: Whereof what's past is prologue, what to come
In yours and my discharge.
— *The Tempest*, II, i, 257–258.

This chapter presents the basic concepts of computer security. The remainder of this book will elaborate on these concepts in order to reveal the logic underlying the principles of these concepts.

We begin with basic security-related services that protect against threats to the security of the system. The next section discusses security policies that identify the threats and define the requirements for ensuring a secure system. Security mechanisms detect and prevent attacks and recover from those that succeed. Analyzing the security of a system requires an understanding of the mechanisms that enforce the security policy. It also requires a knowledge of the related assumptions and trust, which leads to the threats and the degree to which they may be realized. Such knowledge allows one to design better mechanisms and policies to neutralize these threats. This process leads to risk analysis. Human beings are the weakest link in the security mechanisms of any system. Therefore, policies and procedures must take people into account. This chapter discusses each of these topics.

1.1 The Basic Components

Computer security rests on confidentiality, integrity, and availability. The interpretations of these three aspects vary, as do the contexts in which they arise. The interpretation of an aspect in a given environment is dictated by the needs of the individuals, customs, and laws of the particular organization.

1.1.1 Confidentiality

Confidentiality is the concealment of information or resources. The need for keeping information secret arises from the use of computers in institutions with sensitive information such as government and industry. For example, military and civilian institutions in the government often restrict access to information to those who need that information. The first formal work in computer security was motivated by the military's attempt to implement controls to enforce a "need to know" principle. This principle also applies to industrial firms, which keep their proprietary designs secure lest their competitors try to steal the designs. As a further example, all types of institutions keep some types of personnel records secret.

Access control mechanisms support confidentiality. One access control mechanism for preserving confidentiality is *cryptography*, which transforms data to make it incomprehensible. A *cryptographic key* controls access to the untransformed data, but then the cryptographic key itself becomes another datum to be protected.

EXAMPLE: Enciphering an income tax return will prevent anyone without the key from reading the taxable income on the return. If the owner needs to see the return, it must be deciphered. Only the possessor of the cryptographic key can enter it into a deciphering program. However, if someone else can read the key when it is entered into the program and has access to the enciphered return, the confidentiality of the tax return has been compromised.

Other system-dependent mechanisms can prevent information from being illicitly accessed. Data protected only by these controls can be read when the controls fail or are bypassed. Then the controls' advantage is offset by a corresponding disadvantage. They can protect the secrecy of data more completely than cryptography, but if they fail or are evaded, the data becomes visible.

Confidentiality also applies to the existence of data, which is sometimes more revealing than the data itself. The precise number of people who distrust a politician may be less important than knowing that such a poll was taken by the politician's staff. How a particular government agency harassed citizens in its country may be less important than knowing that such harassment occurred. Access control mechanisms sometimes conceal the mere existence of data, lest the existence itself reveal information that should be protected.

Resource hiding is another important aspect of confidentiality. Organizations often wish to conceal their network configuration as well as what systems they are using. They may not wish others to know about specific equipment (because it could be used without authorization or in inappropriate ways), and a company renting time from a service provider may not want others to know what resources it is using. Access control mechanisms provide these capabilities as well.

All the mechanisms that enforce confidentiality require supporting services from the system. The assumption is that the security services can rely on the kernel,

and other agents, to supply correct data. Thus, assumptions and trust underlie confidentiality mechanisms.

1.1.2 Integrity

Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called *authentication*). The source of the information may bear on its accuracy and credibility and on the trust that people place in the information.This dichotomy illustrates the principle that the aspect of integrity known as credibility is central to the proper functioning of a system. We will return to this issue when discussing malicious logic.

EXAMPLE: A newspaper may print information obtained from a leak at the White House but attribute it to the wrong source. The information is printed as received (preserving data integrity), but its source is incorrect (corrupting origin integrity).

Integrity mechanisms fall into two classes: *prevention* mechanisms and *detection* mechanisms.

Prevention mechanisms seek to maintain the integrity of the data by blocking any unauthorized attempts to change the data or any attempts to change the data in unauthorized ways. The distinction between these two types of attempts is important. The former occurs when a user tries to change data that she has no authority to change. The latter occurs when a user authorized to make certain changes in the data tries to change the data in other ways. For example, suppose an accounting system is on a computer. Someone breaks into the system and tries to modify the accounting data. Here an unauthorized user has tried to violate the integrity of the accounting database. But if an accountant hired by the firm to maintain its books tries to embezzle money by sending it overseas and hiding the transactions, a user (the accountant) has tried to change data (the accounting data) in unauthorized ways (by not entering the transfer of funds to a Swiss bank account). Adequate authentication and access controls will generally stop the break-in from the outside, but preventing the second type of attempt requires very different controls.

Detection mechanisms do not try to prevent violations of integrity; they simply report that the data's integrity is no longer trustworthy. Detection mechanisms may analyze system events (user or system actions) to detect problems or (more commonly) may analyze the data itself to see if required or expected constraints still hold. The mechanisms may report the actual cause of the integrity violation (a specific part of a file was altered), or they may simply report that the file is now corrupt.

Working with integrity is very different than working with confidentiality. With confidentiality, the data is either compromised or it is not, but integrity includes both the correctness and the trustworthiness of the data. The origin of the data (how and from whom it was obtained), how well the data was protected

before it arrived at the current machine, and how well the data is protected on the current machine all affect the integrity of the data. Thus, evaluating integrity is often very difficult, because it relies on assumptions about the source of the data and about trust in that source—two underpinnings of security that are often overlooked.

1.1.3 Availability

Availability refers to the ability to use information or resources. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all. The aspect of availability that is relevant to security is that someone may deliberately arrange to deny access to data or to a service by making it unavailable or unusable. System designs usually assume a statistical model to analyze expected patterns of use, and mechanisms ensure availability when that statistical model holds. Someone may be able to manipulate use (or parameters that control use, such as network traffic) so that the assumptions of the statistical model are no longer valid. This means that the mechanisms for keeping the resource or data available are working in an environment for which they were not designed. As a result, they will often fail.

EXAMPLE: Suppose Anne has compromised a bank's secondary system server, which supplies bank account balances. When anyone else asks that server for information, Anne can supply any information she desires. Merchants validate checks by contacting the bank's primary balance server. If a merchant gets no response, the secondary server will be asked to supply the data. Anne's colleague prevents merchants from contacting the primary balance server, so all merchant queries go to the secondary server. Anne will never have a check turned down, regardless of her actual account balance. Notice that if the bank had only one server (the primary one) and that server were unavailable, this scheme would not work. The merchant would be unable to validate the check.

Attempts to block availability, called *denial of service (DoS) attacks*, can be the most difficult to detect, because the analyst must determine if the unusual access patterns are attributable to deliberate manipulation of resources or of environment. Complicating this determination is the nature of statistical models. Even if the model accurately describes the environment, atypical events simply contribute to the nature of the statistics. A deliberate attempt to make a resource unavailable may look like, or be, an atypical event. In some environments, it may not even appear atypical.

1.2 Threats

A *threat* is a potential violation of security. The violation need not actually occur for there to be a threat. The fact that the violation *might* occur means that those

actions that could cause it to occur must be guarded against (or prepared for). Those actions are called *attacks*. Those who execute such actions, or cause them to be executed, are called *attackers*.

The three security services—confidentiality, integrity, and availability—counter threats to the security of a system. Shirey [1739] divides threats into four broad classes: *disclosure*, or unauthorized access to information; *deception*, or acceptance of false data; *disruption*, or interruption or prevention of correct operation; and *usurpation*, or unauthorized control of some part of a system. These four broad classes encompass many common threats. Because the threats are ubiquitous, an introductory discussion of each one will present issues that recur throughout the study of computer security.

Snooping or *eavesdropping*, the unauthorized interception of information, is a form of disclosure. It is passive, suggesting simply that some entity is listening to (or reading) communications or browsing through files or system information. *Passive wiretapping* is a form of snooping in which a network is monitored. (It is called "wiretapping" because of the "wires" that compose the network, although the term is used even if no physical wiring is involved.) Confidentiality services seek to counter this threat.

Modification or *alteration*, an unauthorized change of information, covers three classes of threats. The goal may be deception, in which some entity relies on the modified data to determine which action to take, or in which incorrect information is accepted as correct and is released. If the modified data controls the operation of the system, the threats of disruption and usurpation arise. Unlike snooping, modification is active; it results from an entity changing information. *Active wiretapping* is a form of modification in which data moving across a network is altered, new data is injected, or parts of the data are deleted; the term "active" distinguishes it from snooping ("passive" wiretapping). An example is the *man-in-the-middle* attack, in which an intruder reads messages from the sender and sends (possibly modified) versions to the recipient, in hopes that the recipient and sender will not realize the presence of the intermediary. Integrity services seek to counter this threat.

Masquerading or *spoofing*, an impersonation of one entity by another, is a form of both deception and usurpation. It lures a victim into believing that the entity with which it is communicating is a different entity. For example, if a user tries to log into a computer across the Internet but instead reaches another computer that claims to be the desired one, the user has been spoofed. Similarly, if a user tries to read a web page, but an attacker has arranged for the user to be given a different page, another spoof has taken place. This may be a passive attack (in which the user simply accesses the web page), but it is usually an active attack (in which the attacker issues responses dynamically to mislead the user about the web page). Although masquerading is primarily deception, it is often used to usurp control of a system by an attacker impersonating an authorized manager or controller. Integrity services (called "authentication services" in this context) seek to counter this threat.

Some forms of masquerading may be allowed. *Delegation* occurs when one entity authorizes a second entity to perform functions on its behalf. The distinctions between delegation and masquerading are important. If Susan delegates to

Thomas the authority to act on her behalf, she is giving permission for him to perform specific actions as though she were performing them herself. All parties are aware of the delegation. Thomas will not pretend to be Susan; rather, he will say, "I am Thomas and I have authority to do this on Susan's behalf." If asked, Susan will verify this. On the other hand, in a masquerade, Thomas will pretend to be Susan. No other parties (including Susan) will be aware of the masquerade, and Thomas will say, "I am Susan." Should anyone discover that he or she is dealing with Thomas and ask Susan about it, she will deny that she authorized Thomas to act on her behalf. Even though masquerading is a violation of security, delegation is not.

Repudiation of origin, a false denial that an entity sent (or created) something, is a form of deception. For example, suppose a customer sends a letter to a vendor agreeing to pay a large amount of money for a product. The vendor ships the product and then demands payment. The customer denies having ordered the product and, according to a law in the customer's state, is therefore entitled to keep the unsolicited shipment without payment. The customer has repudiated the origin of the letter. If the vendor cannot prove that the letter came from the customer, the attack succeeds. A variant of this is denial by a user that he created specific information or entities such as files. Integrity mechanisms try to cope with this threat.

Denial of receipt, a false denial that an entity received some information or message, is a form of deception. Suppose a customer orders an expensive product, but the vendor demands payment before shipment. The customer pays, and the vendor ships the product. The customer then asks the vendor when he will receive the product. If the customer has already received the product, the question constitutes a denial of receipt attack. The vendor can defend against this attack only by proving that the customer did, despite his denials, receive the product. Integrity and availability mechanisms attempt to guard against these attacks.

Delay, a temporary inhibition of a service, is a form of usurpation, although it can play a supporting role in deception. Typically, delivery of a message or service requires some time t; if an attacker can force the delivery to take more than time t, the attacker has successfully delayed delivery. This requires manipulation of system control structures, such as network components or server components, and hence is a form of usurpation. If an entity is waiting for an authorization message that is delayed, it may query a secondary server for the authorization. Even though the attacker may be unable to masquerade as the primary server, she might be able to masquerade as that secondary server and supply incorrect information. Availability mechanisms can often thwart this threat.

Denial of service, a long-term inhibition of service, is a form of usurpation, although it is often used with other mechanisms to deceive. The attacker prevents a server from providing a service. The denial may occur at the source (by preventing the server from obtaining the resources needed to perform its function), at the destination (by blocking the communications from the server), or along the intermediate path (by discarding messages from either the client or the server, or both). Denial of service poses the same threat as an infinite delay. Availability mechanisms seek to counter this threat.

Denial of service or delay may result from direct attacks or from problems unrelated to security. From our point of view, the cause and result are important; the intention underlying them is not. If delay or denial of service compromises system security, or is part of a sequence of events leading to the compromise of a system, then we view it as an attempt to breach system security. But the attempt may not be deliberate; indeed, it may be a user error, or the product of environmental characteristics, rather than specific actions of an attacker.

1.3 Policy and Mechanism

Critical to our study of security is the distinction between policy and mechanism:

> **Definition 1–1.** A *security policy* is a statement of what is, and what is not, allowed.

> **Definition 1–2.** A *security mechanism* is a method, tool, or procedure for enforcing a security policy.

Mechanisms can be nontechnical, such as requiring proof of identity before changing a password; in fact, policies often require some procedural mechanisms that technology cannot enforce.

As an example, suppose a university's computer science laboratory has a policy that prohibits any student from copying another student's homework files. The computer system provides mechanisms for preventing others from reading a user's files. Anna fails to use these mechanisms to protect her homework files, and Bill copies them. A breach of security has occurred, because Bill has violated the security policy. Anna's failure to protect her files does not authorize Bill to copy them.

In this example, Anna could easily have protected her files. In other environments, such protection may not be easy. For example, the Internet provides only the most rudimentary security mechanisms, which are not adequate to protect information sent over that network. Nevertheless, acts such as the recording of passwords and other sensitive information violate an implicit security policy of most sites (specifically, that passwords are a user's confidential property and cannot be recorded by anyone).

Policies may be presented mathematically, as a list of allowed (secure) and disallowed (nonsecure) states. For our purposes, we will assume that any given policy provides an axiomatic description of secure states and nonsecure states. In practice, policies are rarely so precise; they normally describe in English, or some other natural language, what users and staff are allowed to do. The ambiguity inherent in such a description leads to states that are not classified as "allowed" or "disallowed." For example, consider the homework policy discussed previously. If someone looks through another user's directory without copying homework