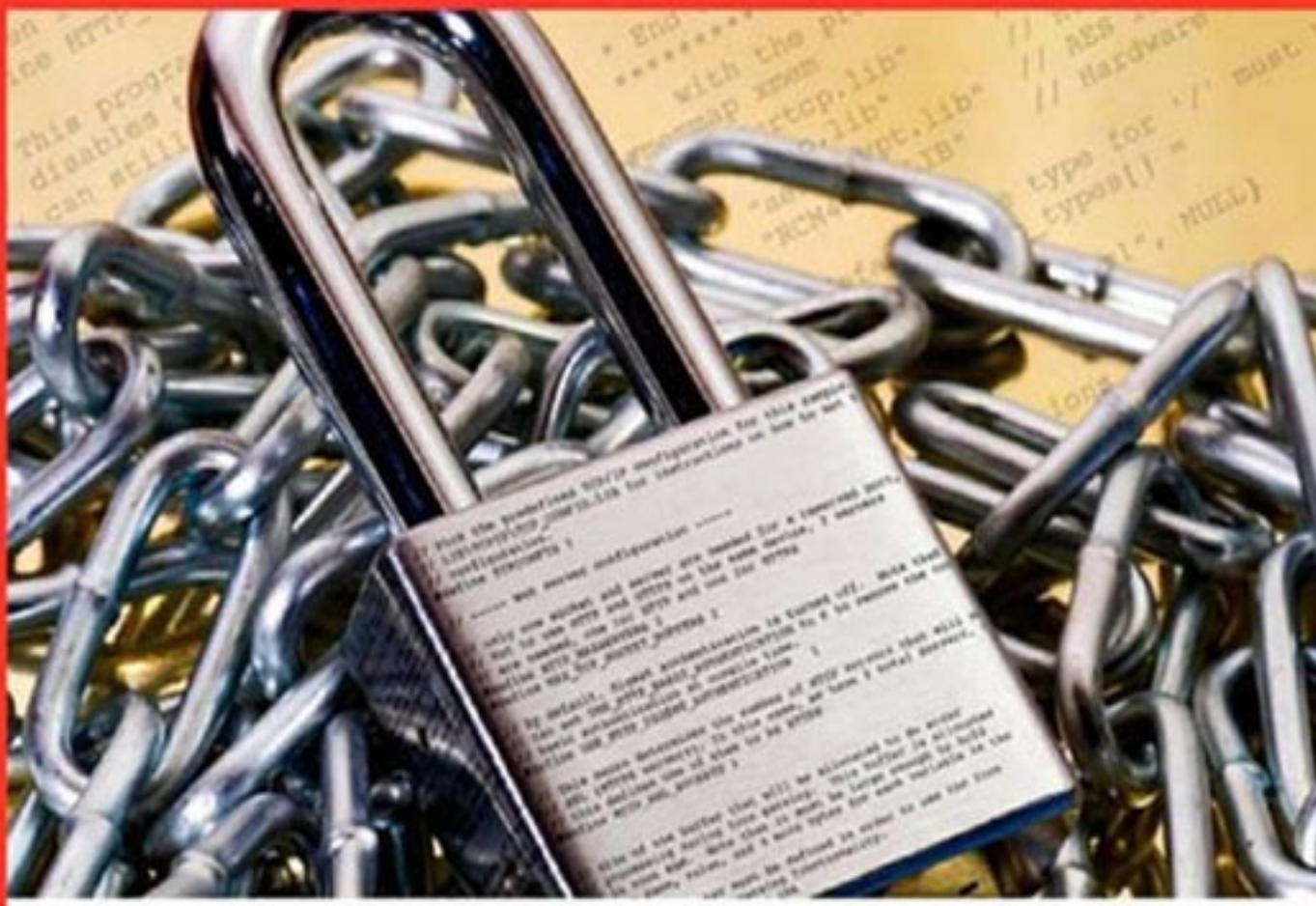
EMBEDDED TECHNOLOGY™ SERIES



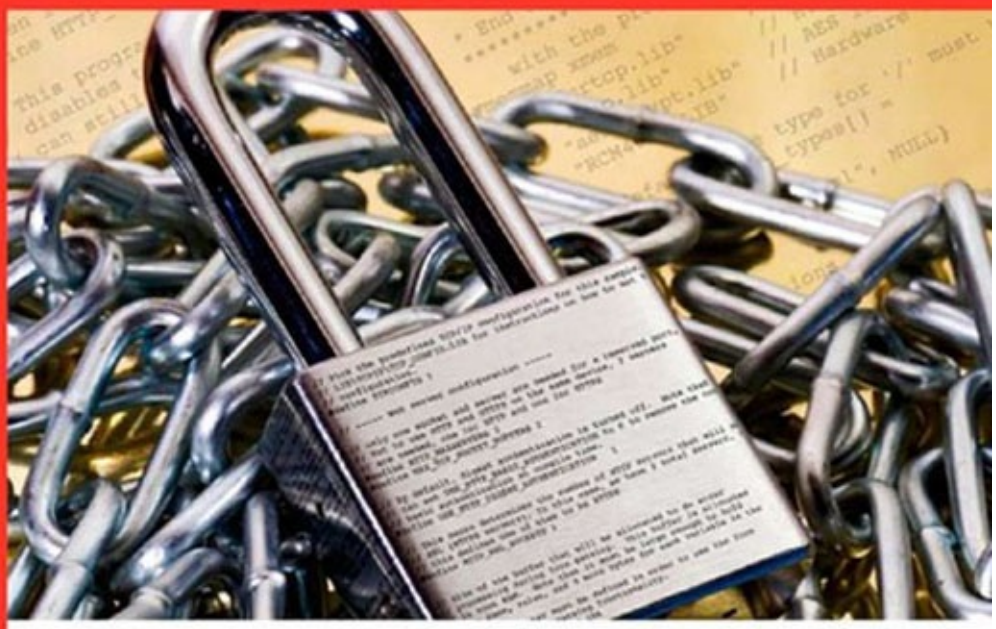
Practical Embedded Security

Building Secure Resource-Constrained Systems

Timothy Stapko

Newnes

EMBEDDED TECHNOLOGY™ SERIES



Practical Embedded Security

Building Secure Resource-Constrained Systems

Timothy Stapko

Newnes

Practical Embedded Security

Building Secure Resource-Constrained Systems

Timothy Stapko

Newnes

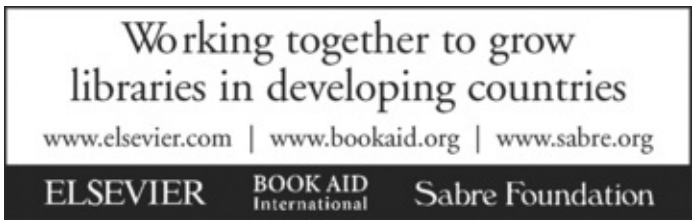
Table of Contents

Copyright

Newnes is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
Linacre House, Jordan Hill, Oxford OX2 8DP, UK

Cover image by iStockphoto

Library of Congress Cataloging-in-Publication Data
Application submitted
British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.
ISBN: 978-0-7506-8215-2
For information on all Newnes publications
visit our Web site at www.books.elsevier.com
07 08 09 10 10 9 8 7 6 5 4 3 2 1
Printed in the United States of America



Preface

Living in a Connected World

1:37 AM. Hoover Dam, straddling the border of Nevada and Arizona, is quietly generating electricity for millions of Americans. The power plant, having recently been retrofitted with a new, remotely controlled automation system, is devoid of life, except for the blinking lights of the network hubs and automated hardware. Suddenly, the control room is ablaze with light, and the whirring of machines breaks the silence. The enormous floodgates open, a torrent of water rushing forth, sending a wave of destruction toward the unsuspecting communities downstream on the Colorado River. The turbines grind to a halt, plunging the desert into darkness. All the while, a teenager in Florida is laughing in the glow of his computer monitor.

Obviously, no one in his or her right mind would trust the control of Hoover Dam to a system with such gaping vulnerabilities, but the hyperbole of the example above does bring up an important point: as more and more of the world goes "online," we are putting more and more trust in the embedded systems that are designed to help us out. Obviously, something like the Hoover Dam would not be automated and connected to the Internet without a large investment in security, if it was automated at all. However, something far simpler, such as a home automation system, would likely not be subject to the same rigorous treatment as a vital hydroelectric power plant. This split between the security requirements of different embedded systems helps to illustrate the challenge of security design for embedded systems engineers. While the cutting edge of security is continually being pushed, low-end hardware and inexpensive systems are often left behind. However, these inexpensive systems are increasingly being networked and used to control more and more vital systems. This leads to an interesting and disturbing problem: Security implementations are often jealously guarded proprietary solutions that sell for thousands of dollars, which is directly at odds with the idea of using inexpensive microcontrollers. There are some options, such as various open-source implementations, but these can be unwieldy and are designed for PCs. If you want to design an inexpensive system and make it secure, there just are not many options.

One of the biggest problems with security in both the Hoover Dam example and home automation is the continual need for updates to keep up

with malicious hackers. Anyone with a PC running Microsoft Windows knows about this from the continual stream of updates and patches for various security issues. One way to alleviate the continual update problem is to design security into the system and develop a solid application to begin with. The primary goal of this book is to introduce the users of inexpensive microcontrollers and embedded processors to the basic practical application of security and to provide some tools and pointers to assist in designing more secure applications with limited resources.

Many of the topics discussed in this book are covered in depth in hundreds of academic papers and tomes filled with arcane symbols. If you are interested in the mathematical underpinnings of cryptography, you are going to want to look elsewhere. However, if you work with microcontrollers or inexpensive embedded systems and security has been something of interest but you have been intimidated by it, then this book is for you. Security is a hard problem, and a lot of very smart people have spent a lot of time working on it. The result is that the topic of security has taken on an intimidating air, especially when it comes to cryptography. This book aims to leverage the large body of work already done on security and adapt it for systems that usually aren't deemed powerful enough. As you will see, it is possible to implement security for some of even the most modest of architectures, such as porting AES to a PIC and using SSL on an 8-bit microprocessor (both of these are covered in extensive case studies of working implementations).

This book covers the practical side of implementing security for embedded systems, using publicly available and inexpensive proprietary implementations whenever possible. However, just having a cryptographic algorithm does not mean you have security. There are a number of issues to consider when using cryptography. We will cover some of them and hopefully provide some insight into how you can find them on your own.

Security in Shades of Gray

There is no such thing as perfect security. Think about it. As long as there is information that can be exploited, there will be someone trying to get it, and with enough resources, any security can be broken. The only way to be sure that information is completely "safe" is to destroy it and kill all the people who know about it, which obviously does no one any good. For this reason, secure systems are built using the idea of hazard tolerance—that is, each system has to have security that meets the requirements for the system. For example, a credit card system needs more security than your personal email (which we all know is completely *in*secure).[1] Unfortunately, security is an inherently difficult problem to solve, since the worst problems are necessarily

those that you cannot possibly predict. The only way to ensure a high level of security is to make your system as robust as possible, and keep it simple enough to understand so you can at least predict some of the more difficult problems. The fewer legitimate access points into your system you implement, the higher the probability it is safe. The more features and possible outside connections available, the more likely it is that you will have an unintended back door. Legitimate entry points need to be secured using a number of different mechanisms, depending on the desired level of security and the application. These mechanisms range from simple password schemes that provide only a small illusion of security to full security protocols that require vast amounts of computing power. Many of the mechanisms used to protect data in full security protocols, such as cryptography, usually require rather high levels of computing power, since they are based on powerful mathematical algorithms that require millions of calculations to be performed. Most security protocols work under the assumption that only the most powerful hardware is available. The problem with this assumption, of course, is that the most powerful hardware is very often, for economic or other reasons, not available.

Enter the *resource constrained system.* Embedded systems that utilize low-cost components may not have the resources required to implement one of these "true" security solutions. Many security protocols, algorithms, and mechanisms are built for the latest and greatest hardware—usually PCs or the most expensive embedded controllers and processors. The vendors will tell you that you need all that power to build a secure system, but the reality is that it really only depends on your application. You may not have access to that kind of power but still need security, so are you simply out of luck? This is why we introduced the chapter with the discussion on hazard tolerance: To build secure systems in a resource-constrained environment, we need to adapt the security or the application so that they work together without bringing the entire system to a halt (potentially a dangerous problem if the device is controlling a large automated punch press, for example). The idea behind this book is that it *is* possible to build secure *and* cost-effective systems.

Who This Book Is For

This book is for anyone interested in making the world more secure. Embedded systems make up the lion's share of the technology market (in volume, not necessarily in revenue) and are as pervasive as the products they help build and control. This book is particularly suited to embedded systems designers and engineers, but it may serve engineering managers as an introduction to a very important subject. Software engineers (embedded or not) are not the target audience, since many of the topics contained herein are covered in more depth in computer science courses and reference materials, but the case studies may still be of some interest. This book takes a practical approach to implementing security using available implementations and does not delve deeply into the mathematical and theoretical foundations of security. For that, the reader is encouraged to attend university computer security courses or read anything by Bruce Schneier.[2]

The content, though technical, should not be outside the reach of nonengineers (although knowledge of programming and Internet technologies will definitely help). The idea is to juxtapose technical content with a higher-level discussion of the subject to get the reader interested in the material and to think about the implications of deploying unsecured applications.

What This Book Is and What It Is Not

The goal of this book is to be a resource for all embedded systems designers—the first place to turn to when looking into security. The scope of computer security is so broad that no single text will provide all the answers you might need. This book aims to give the reader a start in the right direction, looking at some of the technologies available, providing a context in which to discuss those technologies, and giving a starting point for designing secure embedded systems. It should be considered as a first read for embedded security research, to get a quick overview of the challenges involved, and to get some ideas to move forward with. This book is organized so that the reader can quickly locate the information he or she needs and use it as a basis for the research into the project at hand.

This book is not a complete treatment of computer security. Indeed, it is not even a complete treatment of secure embedded systems design. The assumption is that the material presented will get the reader thinking about his or her own project, provide a platform of information to start off with. We will leave the goriest details to texts devoted to rigorous mathematical treatments and detailed security protocols that already flood the market (as an example, the de facto standard reference text for the Secure Sockets Layer by itself is over 400 pages long!).[3]

Why Embedded Security?

Some people may ask why this book is necessary. There are so many general-purpose texts on computer security that it seems one could simply buy a few books and be able to design a secure embedded system. The problem is, as has been mentioned previously, that these texts usually cover security under ideal conditions—that is, the hardware can support the mechanisms that are used and generally can support many mechanisms simultaneously. There are also some who believe that if you need security for a system, you should just buy the latest and greatest hardware and use standard (usually PC-centric) security protocols. The real world of embedded design does not always work like that. Economics plays a big role. Some applications require thousands or millions of units, and saving a few dollars on components really adds up. Why should we have to upgrade to expensive hardware just so that we can use a cookie-cutter implementation designed to work on a PC? In fact, many vendors will claim that you need the most feature-packed hardware package they offer or you will not have any security at all. Since security is considered “voodoo” or “black magic” by many people,

these claims seem reasonable. This couldn't be farther from the truth. There is absolutely no reason we should not expect some level of security from even the most modest hardware. There are things that can be done to provide protection on any system, and we shouldn't have to choose between cost-effectiveness and peace of mind.

The key point to remember is that embedded security is *application dependent*. In the personal computer–dominated Internet, security solutions are typically designed to be general and flexible, reflecting the properties of the systems being protected (the general-purpose and flexible PCs). In the embedded world, the systems needing protection have unique properties and are particularly suited to specific applications. The security needed by these applications is similarly unique and specific. The general solutions typically employed in the existing Internet often do not work "out of the box" for embedded systems, and simply porting existing security protocols can lead to code bloat for features that are unnecessary. Instead, the security for an embedded system needs to be specifically tailored to the particular application being developed. This is not to say that existing protocols and policies cannot be used (and indeed *should* be used), but rather that we need to adapt Internet security to an embedded world through analysis of the applications to which these concepts are applied.

The reader is encouraged to use this book as a way to start learning how to look at security for embedded systems, rather than as a universal solution to all your security needs. Reading the book cover to cover will definitely increase your knowledge of the subject matter, but it is by no means necessary in order to get what you need out of the book. It is recommend that you read the following three chapters, covering computer security fundamentals, Internet security, and the principles of embedded Internet security, respectively. These chapters will give you a foundation for the rest of the text. If you are already familiar with computer and Internet security, then go ahead and skip to Chapter 3 (where we cover security algorithms and protocols) unless you need a refresher (which is recommended anyway). After reading those chapters, the rest of the book is organized so that you can easily find a particular topic of interest, or just continue reading through all the chapters to get a comprehensive coverage of the content. We will look at the various aspects of embedded Internet security, from communications, to software security implementations, to hardware for security.

Roadmap

Chapter 1 introduces (or reintroduces) readers to the basics of computer security, with some light theoretical background, a look at the different

subfields within computer security, and, most importantly, a look at the security mechanisms that will be covered in the rest of the book. This information is provided as background for readers unfamiliar with security and cryptography. Those readers with some background may wish to skip ahead to later chapters.

Our security introduction will begin with some light theory to give the reader a foundation for the rest of the chapters. The basic premise of computer security is *access control*—who can and cannot control and access some particular application or data. We will look at the theory behind access control and introduce some analysis techniques based on the access control matrix model.

In Chapter 2, we will look at some low-level networking protocols and what to look out for in using them. Part of the reason for looking at this low-level functionality is to get used to analyzing the protocols you intend to use. Without some understanding of how the low-level things work, you cannot have much assurance that your application is secure.

Some of the topics to cover are TCP/IP. UDP, Serial communications, PPP, and Ethernet (wireless protocols will be discussed later). There will be a brief description of each protocol, what it is used for, and how it works with security mechanisms. There are definite differences between all of these networking technologies that may lead to issues when attempting to secure an application. Choosing a communications technology wisely can save headache and extra work later. We will try to sort out the mess of available options in an attempt to help the reader choose the technology that is best for them.

In Chapter 3 we will look at the world of Internet security, from simple hashing techniques used in web-based applications to full-blown security protocols like SSL. In this chapter we look at various standard cryptographic algorithms and how they are used. This chapter covers the algorithms that are the building blocks of secure networked systems.

Chapter 4 is all about SSL. The Secure Sockets Layer is so important to Internet security that we devote an entire chapter to it. SSL is the de facto standard for secure Internet transactions. It has achieved this status by being not only secure, but being highly generic as well. SSL exists in the network layer between TCP and your application, providing blanket security to all data transferred over the network. The API for SSL is typically very similar to the standard network sockets API (POSIX-style). For this reason, it is simple to transform any plain TCP/IP application into a secure Internet application with very little effort. We will look at how to implement SSL for embedded platforms and cover the options we have in adapting the protocol to work for specific applications. SSL is very component-driven, and we can use this to our advantage.

Chapter 5 covers security from the embedded systems developer's point of view. First and foremost, the primary concept that will be introduced here and reinforced throughout the text is that the embedded systems we are covering are *NOT PCs or expensive devices running Windows or Linux*. In this chapter we will also introduce strategies for choosing algorithms and protocols for a particular platform and for choosing a platform based on security requirements. There are many tradeoffs to be considered, and we will look at some examples to get an idea of what is available to the embedded developer. We will look at some of the features, pros and cons, of different protocols and algorithms. The reader can then use this section as a guide to using the rest of the book for their applications.

Also in Chapter 5, we will look at embedded security protocols—or, rather, the lack of them. The problem is that unlike a PC, which can support a myriad of protocols and algorithms simultaneously, each embedded device can only have one or two protocols resident at any given time. We will spend the rest of this chapter looking at why designing an embedded security protocol is extremely difficult. The requirements vary greatly between applications, and the capabilities of the hardware have a similar variance. This section will provide a justification for the treatment of the protocols and algorithms throughout the remainder of the text.

By Chapter 6 the reader should have a fairly decent understanding of the concepts involved in securing a system, and he/she should have a relatively clear picture of why securing embedded systems represents a significant challenge. This chapter covers networking technologies that will allow a device to be connected wirelessly to the Internet or each other. The vast majority of new resource-constrained networked systems will likely be in the wireless and cellular arena. Wireless networking has taken off as a hot new technology and represents one of the fastest-growing markets in the industry. We will look at protocols that are commonly used and appropriate for embedded development, such as 802.11 (Wi-Fi), Bluetooth, and ZigBee. Beyond this, we will look at each protocol for the purposes of securing data being sent and received.

In Chapter 7 we look at client/server applications and their relevance to the embedded world. The World Wide Web is by far the most recognizable and widely used application of the Internet and represents the fundamental client/server relationship. As such, it stands to reason that a large number of networked embedded applications will have some type of Web service functionality. For this reason, we will start our tour of secure embedded applications with a look at the Web and other client/server applications.

Chapter 8 covers some basic ideas and common pitfalls related to the optimization of security mechanisms. Here we look at cryptographic

algorithms and how to make them work for embedded machines. Cryptography presents the biggest challenge to any security solution to be implemented for a resource-constrained embedded device because of the requirements of these computationally complex algorithms. Cryptography is notoriously expensive when it comes to clock cycles, and many algorithms are not too friendly to small storage devices (code and data). We will look at various strategies for choosing algorithms for specific applications and look at some specific algorithms, as well as some strategies to avoid some of the more problematic resource issues.

In Chapter 9 we look at hardware alternatives that allow embedded developers to get the most out of their systems. These alternatives include hardware assistance and complete hardware security solutions. Hardware assistance involves dedicated circuitry that provides part or all of the functionality of a security mechanism at the speed of hardware. Complete hardware solutions include entire security protocols implemented in silicon for the maximum possible performance. Both of these ideas can be used to augment an embedded system without sacrificing the affordability of the base hardware. We will also briefly cover physical security. When applying security to an embedded system, this is an important point that may be overlooked. The reason for this is that, unlike PCs and servers, which can be deployed in controlled environments, embedded devices are deployed practically everywhere. We will look at some different technologies to consider when choosing an embedded device for a secure embedded system.

In Chapter 10 we cover some miscellaneous issues with security, such as programming gotchas in languages like C and recognizing and dealing with attacks. We finish up the chapter with a brief mention about the political issues regarding cryptography and exported applications. We start the chapter by looking at how development tools and languages factor into the security of an application. Some languages, such as Java, provide some built-in safeguards that help with overall security, such as strict typing. C is the predominant language used for many embedded devices, but it suffers from many shortcomings that make it difficult to use in implementing a secure system. We will look at some of the features of different languages and discuss what to look for when choosing tools and languages and when developing software for your system. We will also briefly cover attacks and how to deal with them. Any secure system, or for that matter, *any* system connected to a network (the Internet or proprietary) *will* be subject to attacks—both intentional (i.e., malicious hackers) and inadvertently (i.e., heavy network traffic leading to Denial of Service; we will classify accidents as attacks to simplify this chapter). We will look at how to deal with future attacks, currently occurring attacks, and the aftermath after a successful attack.

Finally, we will briefly look at export issues, primarily to inform the reader of their existence, since dealing with political issues is far beyond the scope of this text. We close out Chapter 10 with some further reading and a discussion about the future of embedded security.

In Chapters 11 and 12 we look at some application case studies. In these chapters we will develop two applications from requirements to implementation that use inexpensive hardware but also need some level of security. These chapters are provided to help the reader see embedded security applied to real-world implementations. We will go through the entire development of each application, from initial requirements, to developing a security policy, verifying the design, and finally deploying the application. There is complete code for the two applications, with full listings of the programs (not including library code) in Appendix A. The applications are working implementations on real hardware (using a PIC and the 8-bit Rabbit 4000 microprocessor) that provide some insights into the development of a secure application for a platform with limited resources.

---

[1]This may not be completely true anymore, as there are various applications (such as PGP) that provide encryption services for email. Generally speaking, though, you really should never send your credit card number via email.

[2]Bruce Schneier is widely known as a foremost expert in computer security and is the author of several excellent books on the subject, most notably *Applied Cryptography*.

[3]*SSL and TLS: Designing and Building Secure Systems*, by Eric Rescorla.

CHAPTER 1

Computer Security Introduction and Review

This chapter is intended to provide a quick introduction to computer security for embedded systems engineers who may not have a formal background in computer science and computer security. For the more advanced reader, this chapter serves as a review of computer security before delving into the later material. This chapter is by no means a complete treatment of the theory behind computer security—literally hundreds of books have been written on the subject—but it should at least provide a basic context for all readers. At the end of the chapter, we will provide a list of further reading for readers wanting a deeper treatment of the theory. We will briefly touch on the most important concepts, spending most of the discussion on those ideas that are most pertinent to embedded and resource-constrained systems.

Computer security is a rapidly evolving field; every new technology is a target for hackers, crackers, spyware, trojans, worms, and malicious viruses. However, the threat of computer attacks dates back to the earliest days of mainframes used in the 1960s. As more and more companies turned to computer technology for important tasks, attacks on computer systems became more and more of a worry. In the early days of the Personal Computer, the worry was viruses. With the advent of the World Wide Web and the exponential expansion of the Internet in the late 1990s, the worry became hackers and denial of service attacks. Now, at the dawn of the new millennium, the worry has become spam, malware/spyware, email worms, and identity theft. All of this begs the question: How do we protect ourselves from this perpetual onslaught of ever-adapting attacks?

The answer, as you may have guessed, is to be vigilant, staying one step ahead of those who would maliciously compromise the security of your system. Utilizing cryptography, access control policies, security protocols, software engineering best practices, and good old common sense, we can improve the security of any system. As is stated by Matt Bishop,[1] computer security is both a science *and* an art. In this chapter, we will introduce this idea to embedded systems engineers and review the basic foundations of computer security to provide a foundation for the rest of the book.

What Is Security?

To begin, we need to define *security* in a fashion appropriate for our discussion. For our purposes, we will define computer security *as follows*:

> *Definition:* Computer Security. *Computer security is the protection of personal or confidential information and/or computer resources from individuals or organizations that would willfully destroy or use said information for malicious purposes.*

Another important point often overlooked in computer security is that the security does not need to be limited to simply the protection of resources from malicious sources—it could actually involve protection from the application itself. This is a topic usually covered in software engineering, but the concepts used there are very similar to the methods used to make an application secure. Building a secure computer system also involves designing a robust application that can deal with internal failures; no level of security is useful if the system crashes and is rendered unusable. A truly secure system is not only safe from external forces, but from internal problems as well. The most important point is to remember that *any* flaw in a system can be exploited for malicious purposes.

If you are not familiar with computer security, you are probably thinking, "What does 'protection' actually mean for a computer system?" It turns out that there are many factors that need to be considered, since any flaw in the system represents a potential vulnerability. In software, there can be buffer overflows, which potentially allow access to protected resources within the system. Unintended side effects and poorly understood features can also be gaping holes just asking for someone to break in. Use of cryptography does not guarantee a secure system either; using the strongest cryptography available does not help if someone can simply hack into your machine and steal that data directly from the source. Physical security also needs to be considered. Can a malicious individual gain access to an otherwise protected system by compromising the physical components of the system (this is especially important for embedded systems)? Finally, there is the human factor. Social engineering, essentially the profession practiced by con artists, turns out to be a major factor in many computer system security breaches. This book will cover all of the above issues, except the human factor. There is little that can be done to secure human activities, and it is a subject best left to lawyers and politicians.

What Can We Do?

In the face of all these adversities, what can we do to make the system less vulnerable? Next we will look at the basics of computer security from a general level to familiarize the reader with the concepts that will be reiterated