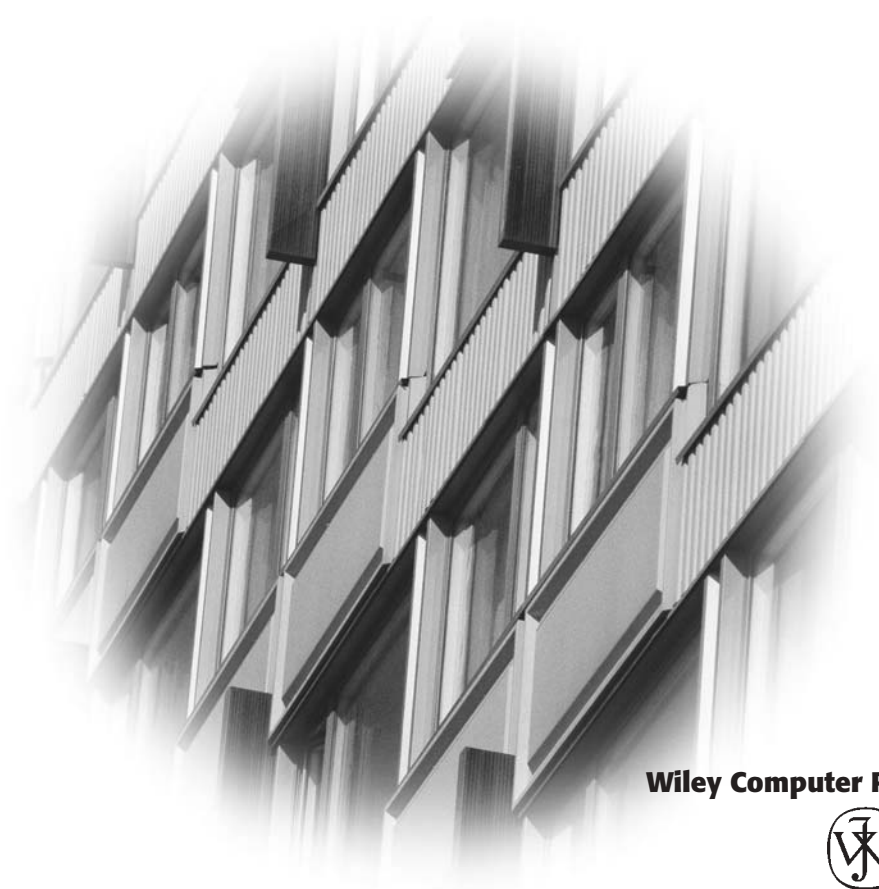


# Designing Security Architecture Solutions

Jay Ramachandran



**Wiley Computer Publishing**



**John Wiley & Sons, Inc.**

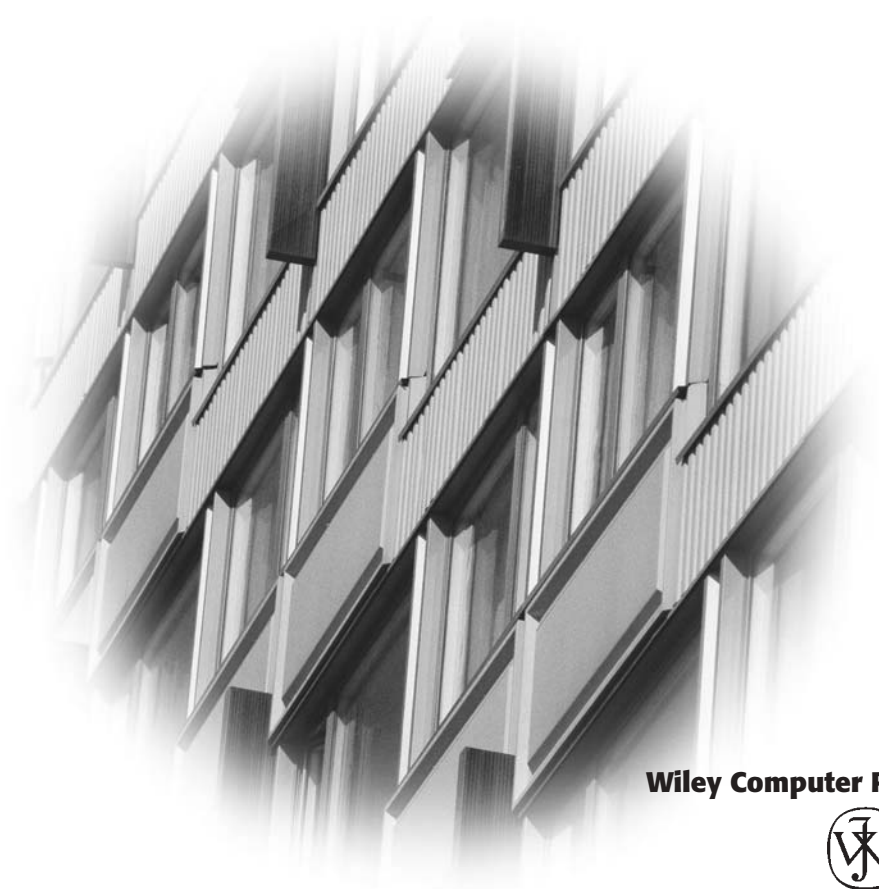


# **Designing Security Architecture Solutions**



# Designing Security Architecture Solutions

Jay Ramachandran



**Wiley Computer Publishing**



**John Wiley & Sons, Inc.**

Publisher: Robert Ipsen  
Editor: Carol Long  
Managing Editor: Micheline Frederick  
Developmental Editor: Adaobi Obi  
Text Design & Composition: D&G Limited, LLC

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This book is printed on acid-free paper. ∞

Copyright © 2002 by Jay Ramachandran. All rights reserved.

**Published by John Wiley & Sons, Inc.**

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ @ WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

***Library of Congress Cataloging-in-Publication Data:***

Ramachandran, Jay

Designing security architecture solutions / Jay Ramachandran.

p. cm.

“Wiley Computer Publishing.”

ISBN: 0-471-20602-4 (acid-free paper)

1. Computer security. I. Title.

QA76.9.A25 R35 2002

005.8—dc21

2001006821

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

# DEDICATION

*For Ronak, Mallika, and Beena*



<b>Preface</b>	<b>xvii</b>
<b>Acknowledgments</b>	<b>xxvii</b>
<b>Part One Architecture and Security</b>	<b>1</b>
<b>Chapter 1 Architecture Reviews</b>	<b>3</b>
Software Process	3
Reviews and the Software Development Cycle	4
Software Process and Architecture Models	5
Kruchten's 4+1 View Model	6
The Reference Model for Open Distributed Processing	7
Rational's Unified Process	9
Software Process and Security	10
Architecture Review of a System	11
The Architecture Document	12
The Introduction Section	13
Sections of the Architecture Document	15
The Architecture Review Report	19
Conclusions	19
<b>Chapter 2 Security Assessments</b>	<b>21</b>
What Is a Security Assessment?	21
The Organizational Viewpoint	22
The Five-Level Compliance Model	23
The System Viewpoint	24
Pre-Assessment Preparation	26
The Security Assessment Meeting	26
Security Assessment Balance Sheet Model	27
Describe the Application Security Process	29
Identify Assets	30
Identify Vulnerabilities and Threats	30
Identify Potential Risks	30
Examples of Threats and Countermeasures	32
Post-Assessment Activities	32

Why Are Assessments So Hard?	32
Matching Cost Against Value	33
Why Assessments Are Like the Knapsack Problem	36
Why Assessments Are Not Like the Knapsack Problem	38
Enterprise Security and Low Amortized Cost Security Controls	39
Conclusion	40
<b>Chapter 3 Security Architecture Basics</b>	<b>43</b>
Security As an Architectural Goal	44
Corporate Security Policy and Architecture	45
Vendor Bashing for Fun and Profit	46
Security and Software Architecture	48
System Security Architecture Definitions	48
Security and Software Process	50
Security Design Forces against Other Goals	51
Security Principles	52
Additional Security-Related Properties	53
Other Abstract or Hard-to-Provide Properties	54
Inference	54
Aggregation	55
Least Privilege	56
Self-Promotion	56
Graceful Failure	56
Safety	57
Authentication	58
User IDs and Passwords	58
Tokens	59
Biometric Schemes	59
Authentication Infrastructures	60
Authorization	60
Models for Access Control	61
Mandatory Access Control	61
Discretionary Access Control	61
Role-Based Access Control	63
Access Control Rules	66
Understanding the Application's Access Needs	69
Other Core Security Properties	71
Analyzing a Generic System	71
Conclusion	73
<b>Chapter 4 Architecture Patterns in Security</b>	<b>75</b>
Pattern Goals	75
Common Terminology	76
Architecture Principles and Patterns	77
The Security Pattern Catalog	78
Entity	78
Principal	78

Context Holders	81
Session Objects and Cookies	81
Ticket/Token	82
Sentinel	83
Roles	83
Service Providers	84
Directory	84
Trusted Third Party	87
Validator	88
Channel Elements	89
Wrapper	89
Filter	91
Interceptor	93
Proxy	95
Platforms	96
Transport Tunnel	96
Distributor	97
Concentrator	98
Layer	98
Elevator	100
Sandbox	101
Magic	103
Conclusion	104

---

## **Part Two Low-Level Architecture 105**

### **Chapter 5 Code Review 107**

---

Why Code Review Is Important	107
Buffer Overflow Exploits	108
Switching Execution Contexts in UNIX	111
Building a Buffer Overflow Exploit	111
Components of a Stack Frame	112
Why Buffer Overflow Exploits Enjoy Most-Favored Status	113
Countermeasures Against Buffer Overflow Attacks	114
Avoidance	114
Prevention by Using Validators	114
Sentinel	115
Layer	115
Sandbox	116
Wrapper	116
Interceptors	118
Why Are So Many Patterns Applicable?	118
Stack Growth Redirection	119
Hardware Support	120

Security and Perl	120
Syntax Validation	121
Sentinel	122
Sandbox	122
Bytecode Verification in Java	123
Good Coding Practices Lead to Secure Code	125
Conclusion	126
<b>Chapter 6</b>	<b>Cryptography</b>
	<b>129</b>
The History of Cryptography	130
Cryptographic Toolkits	132
One-Way Functions	133
Encryption	133
Symmetric Encryption	134
Encryption Modes	135
Asymmetric Encryption	136
Number Generation	137
Cryptographic Hash Functions	138
Keyed Hash Functions	138
Authentication and Digital Certificates	139
Digital Signatures	139
Signed Messages	140
Digital Envelopes	140
Key Management	141
Cryptanalysis	142
Differential Cryptanalysis	142
Linear Cryptanalysis	142
Cryptography and Systems Architecture	143
Innovation and Acceptance	143
Cryptographic Flaws	144
Algorithmic Flaws	145
Protocol Misconstruction	145
Implementation Errors	145
Wired Equivalent Privacy	146
Performance	147
Comparing Cryptographic Protocols	148
Conclusion	149
<b>Chapter 7</b>	<b>Trusted Code</b>
	<b>151</b>
Adding Trust Infrastructures to Systems	152
The Java Sandbox	153
Running Applets in a Browser	154
Local Infrastructure	155
Local Security Policy Definition	155
Local and Global Infrastructure	156

Security Extensions in Java	156
Systems Architecture	157
Microsoft Authenticode	157
Global Infrastructure	157
Local Infrastructure	158
Structure within the Local Machine	158
Authenticode and Safety	159
Internet Explorer Zones	159
Customizing Security within a Zone	159
Role-Based Access Control	160
Accepting Directives from Downloaded Content	160
Netscape Object Signing	162
Signed, Self-Decrypting, and Self-Extracting Packages	163
Implementing Trust within the Enterprise	163
Protecting Digital Intellectual Property	165
Thompson's Trojan Horse Compiler	170
Some Notation for Compilers and Programs	171
Self-Reproducing Programs	171
Looking for Signatures	173
Even Further Reflections on Trusting Trust	175
An Exercise to the Reader	176
Perfect Trojan Horses	176
Conclusion	177
<b>Chapter 8 Secure Communications</b>	<b>179</b>
The OSI and TCP/IP Protocol Stacks	180
The Structure of Secure Communication	182
The Secure Sockets Layer Protocol	182
SSL Properties	183
The SSL Record Protocol	184
The SSL Handshake Protocol	184
SSL Issues	186
The IPsec Standard	187
IPsec Architecture Layers	188
IPsec Overview	189
Policy Management	190
IPsec Transport and Tunnel Modes	191
IPsec Implementation	192
Authentication Header Protocol	192
Encapsulating Security Payload	193
Internet Key Exchange	193
Some Examples of Secure IPsec Datagrams	194
IPsec Host Architecture	195
IPsec Issues	195
Conclusion	198

<b>Part Three</b>	<b>Mid-Level Architecture</b>	<b>199</b>
<b>Chapter 9</b>	<b>Middleware Security</b>	<b>201</b>
	Middleware and Security	202
	Service Access	202
	Service Configuration	202
	Event Management	203
	Distributed Data Management	204
	Concurrency and Synchronization	204
	Reusable Services	205
	The Assumption of Infallibility	206
	The Common Object Request Broker Architecture	207
	The OMG CORBA Security Standard	208
	The CORBA Security Service Specification	208
	Packages and Modules in the Specification	209
	Vendor Implementations of CORBA Security	211
	CORBA Security Levels	212
	Secure Interoperability	212
	The Secure Inter-ORB Protocol	213
	Secure Communications through SSL	214
	Why Is SSL Popular?	215
	Application-Unaware Security	216
	Application-Aware Security	218
	Application Implications	220
	Conclusion	221
<b>Chapter 10</b>	<b>Web Security</b>	<b>223</b>
	Web Security Issues	225
	Questions for the Review of Web Security	226
	Web Application Architecture	227
	Web Application Security Options	228
	Securing Web Clients	230
	Active Content	230
	Scripting Languages	231
	Browser Plug-Ins and Helper Applications	231
	Browser Configuration	231
	Connection Security	232
	Web Server Placement	232
	Securing Web Server Hosts	233
	Securing the Web Server	235
	Authentication Options	235
	Web Application Configuration	236
	Document Access Control	237
	CGI Scripts	237
	JavaScript	238
	Web Server Architecture Extensions	238

Enterprise Web Server Architectures	239
The Java 2 Enterprise Edition Standard	240
Server-Side Java	241
Java Servlets	241
Servlets and Declarative Access Control	242
Enterprise Java Beans	243
Conclusion	244
<b>Chapter 11 Application and OS Security</b>	<b>247</b>
Structure of an Operating System	249
Structure of an Application	251
Application Delivery	253
Application and Operating System Security	254
Hardware Security Issues	254
Process Security Issues	255
Software Bus Security Issues	256
Data Security Issues	256
Network Security Issues	256
Configuration Security Issues	257
Operations, Administration, and Maintenance Security Issues	258
Securing Network Services	258
UNIX Pluggable Authentication Modules	260
UNIX Access Control Lists	262
Solaris Access Control Lists	264
HP-UX Access Control Lists	267
Conclusion	268
<b>Chapter 12 Database Security</b>	<b>269</b>
Database Security Evolution	270
Multi-Level Security in Databases	270
Architectural Components and Security	273
Secure Connectivity to the Database	274
Role-Based Access Control	276
The Data Dictionary	277
Database Object Privileges	278
Issues Surrounding Role-Based Access Control	278
Database Views	279
Security Based on Object-Oriented Encapsulation	281
Procedural Extensions to SQL	282
Wrapper	283
Sentinel	284
Security through Restrictive Clauses	285
Virtual Private Database	286
Oracle Label Security	287
Read and Write Semantics	287
Conclusion	291

<b>Part Four</b>	<b>High-Level Architecture</b>	<b>293</b>
<b>Chapter 13</b>	<b>Security Components</b>	<b>295</b>
	Secure Single Sign-On	297
	Scripting Solutions	298
	Strong, Shared Authentication	298
	Network Authentication	299
	Secure SSO Issues	299
	Public-Key Infrastructures	301
	Certificate Authority	303
	Registration Authority	303
	Repository	304
	Certificate Holders	304
	Certificate Verifiers	304
	PKI Usage and Administration	304
	PKI Operational Issues	305
	Firewalls	306
	Firewall Configurations	307
	Firewall Limitations	307
	Intrusion Detection Systems	308
	LDAP and X.500 Directories	311
	Lightweight Directory Access Protocol	312
	Architectural Issues	313
	Kerberos	314
	Kerberos Components in Windows 2000	315
	Distributed Computing Environment	317
	The Secure Shell, or SSH	318
	The Distributed Sandbox	319
	Conclusion	321
<b>Chapter 14</b>	<b>Security and Other Architectural Goals</b>	<b>323</b>
	Metrics for Non-Functional Goals	324
	Force Diagrams around Security	324
	Normal Architectural Design	325
	Good Architectural Design	327
	High Availability	328
	Security Issues	331
	Robustness	332
	Binary Patches	333
	Security Issues	334
	Reconstruction of Events	335
	Security Issues	335
	Ease of Use	336
	Security Issues	337

Maintainability, Adaptability, and Evolution	338
Security Issues	339
Scalability	340
Security Issues	340
Interoperability	341
Security Issues	341
Performance	342
Security Issues	344
Portability	345
Security Issues	346
Conclusion	347
<b>Chapter 15</b>	<b>Enterprise Security Architecture</b>
	<b>349</b>
Security as a Process	350
Applying Security Policy	351
Security Data	351
Databases of Record	352
Enterprise Security as a Data Management Problem	353
The Security Policy Repository	353
The User Repository	354
The Security Configuration Repository	354
The Application Asset Repository	355
The Threat Repository	356
The Vulnerability Repository	356
Tools for Data Management	357
Automation of Security Expertise	358
Directions for Security Data Management	359
David Isenberg and the “Stupid Network”	360
Extensible Markup Language	362
XML and Data Security	363
The XML Security Services Signaling Layer	363
XML and Security Standards	364
J2EE Servlet Security Specification	365
XML Signatures	365
XML Encryption	366
S2ML	366
SAML	367
XML Key Management Service	367
XML and Other Cryptographic Primitives	368
The Security Pattern Catalog Revisited	369
XML-Enabled Security Data	370
HGP: A Case Study in Data Management	371
Building a Single Framework for Managing Security	372
Conclusion	373

<b>Part Five</b>	<b>Business Cases and Security</b>	<b>375</b>
<b>Chapter 16</b>	<b>Building Business Cases for Security</b>	<b>377</b>
	Building Business Cases for Security	378
	Financial Losses to Computer Theft and Fraud	379
	Case Study: AT&T's 1990 Service Disruption	381
	Structure of the Invita Case Study	382
	Security at Invita Securities Corp.	384
	The Pieces of the Business Case	385
	Development Costs	385
	Operational Costs	387
	Time-Out 1: Financial Formulas	388
	Interest Rate Functions	388
	Net Present Value	388
	Internal Rate of Return	389
	Payback Period	389
	Uniform Payment	389
	Break-Even Analysis	389
	Breaking Even is Not Good Enough	390
	Time-Out 2: Assumptions in the Saved Losses Model	390
	Assumptions in the Saved Losses Model	391
	Steady State Losses	391
	Losses from a Catastrophic Network Disruption	392
	The Agenda for the Lockup	392
	Steady-State Losses	395
	Catastrophic Losses	395
	The Readout	396
	Insuring Against Attacks	397
	Business Case Conclusion	398
	A Critique of the Business Case	399
	Insurance and Computer Security	400
	Hacker Insurance	402
	Insurance Pricing Methods	403
	Conclusion	404
<b>Chapter 17</b>	<b>Conclusion</b>	<b>407</b>
	Random Advice	408
	<b>Glossary</b>	<b>413</b>
	<b>Bibliography</b>	<b>421</b>
	<b>Index</b>	<b>435</b>

**T**here is an invisible elephant in this book: your application. And, it sits at the center of every topic we touch in each chapter we present. This book is for systems architects who are interested in building security into their applications. The book is designed to be useful to architects in three ways: as an introduction to security architecture, as a handbook on security issues for architecture review, and as a catalog of designs to look for within a security product.

## Audience

---

This book is meant to be a practical handbook on security architecture. It aims to provide software systems architects with a contextual framework for thinking about security. This book is not for code writers directly, although we do talk about code when appropriate. It is targeted toward the growing technical community of people who call themselves systems architects. A systems architect is the technical leader on any large project with overall responsibility for architecture, design, interface definition, and implementation for the system. Architects play nontechnical roles, as well. They are often involved in the planning and feasibility stages of the project, helping its owners make a business case for the system. They must ensure that the project team follows corporate security guidelines and the software development process all the way to delivery. Architects have deep domain knowledge of the application, its function, and its evolution but often are not as experienced in security architecture.

The primary audience for this book consists of project managers, systems architects, and software engineers who need to secure their applications. It provides a conceptual architectural framework that answers the questions, “What is systems security architecture? How should I choose from a bewildering array of security product offerings? How should I then integrate my choices into my software? What common problems occur during this process? How does security affect the other goals of my system architecture? How can I justify the expense of building security into my application?”

If you are currently working on a large project or you have access to the architecture documentation of a software system you are familiar with, keep it handy and use its architecture to give yourself a frame of reference for the discussion. A good application can give additional depth to a particular recommendation or provide context for any architectural issues on security or software design.

We assume that you have some experience with implementing security solutions and getting your hands dirty. Although we introduce and present many security concepts, we would not recommend learning about computer security from this book, because in the interests of covering as many aspects of architecture and security as we can, we will often cheerfully commit the sin of simplification. We will always add references to more detail when we do simplify matters and hope this situation will not confuse the novice reader. We hope that by the end of the book, the systems architects among you will have gained some insights into security while the security experts wryly note our mastery of the obvious. That would mean that we have succeeded in striking the right balance!

## Software Architecture

---

Software architecture in the past 10 years has seen growing respectability. More and more software professionals are calling themselves software *architects* in recognition that enterprise systems are increasingly complex, expensive, and distributed. Applications have raised the bar for feature requirements such as availability, scalability, robustness, and interoperability. At the same time, as a business driver, enterprise security is front and center in the minds of many managers. There is a tremendously diverse community of security professionals providing valuable but complicated services to these enterprise architects. Architects have clear mandates to implement corporate security policy, and many certainly feel a need for guidelines on how to do so. We wrote this book to provide architects with a better understanding of security.

Software development converts requirements into systems, products, and services. Software architecture has emerged as an important organizing principle, providing a framework upon which we hang the mass of the application. Companies are recognizing the value of enterprise architecture guidelines, along with support for process definition, certification of architects, and training. Software architecture promises cost savings by improving release cycle time, reducing software defects, enabling reuse of architecture and design ideas, and improving technical communication.

There are many excellent books on security and on software architecture. There is also a vast and mature collection of academic literature in both areas, many listed in our bibliography. This book targets readers in the intersection of the two fields.

When we use the term *system* or *application* in this book, we mean a collection of hardware and software components on a platform to support a business function with boundaries that demark the inside and outside of the system, along with definitions of interfaces to other systems. Systems have business roles in the company. They belong to business processes and have labels: customer Web application, benefits directory, employee payroll database, customer order provisioning, billing, network management, fulfillment, library document server, and so on.

Security can be approached from perspectives other than the viewpoint of securing a system. A project might be developing a shrink-wrapped product, such as a computer

game or a PC application; or might be providing a distributed service, such as an e-mail or naming server; or be working on an infrastructure component, such as a corporate directory. Security goals change with each change in perspective. Our presentation of security principles in this book is general enough to apply to these other viewpoints, which also can benefit from secure design.

## Project Objectives versus Security Experience

Companies wish to include security policy into architecture guidelines but run into difficulties trying to chart a path on implementation decisions. Unless we realize that the problem does not lie with our talented and competent development teams but instead lies in their lack of background information about security, we will run into significant resistance project after project—repeatedly going over the same security issues at the architecture review. We must be able to present security issues in an architectural context to guide the project.

As system architects, we would like to believe that all our decisions are driven by technical considerations and business goals. We would like to believe that every time our project team meets to make a decision, we would be consistent—arriving at the same decision no matter who took the day off. Human nature and personal experience inform our decisions as well, however. On a system that is under construction within the confines of budget and time, the strengths of the lead architects and developers can strongly warp the direction and priority of functional and non-functional goals.

An object-oriented methodology guru might spend a fair amount of resources developing the data model and class diagrams. A programmer with a lot of experience building concurrent code might introduce multi-threading everywhere, creating producers and consumers that juggle mutexes, locks, and condition variables in the design. A database designer with experience in one product might bring preconceived notions of how things should be to the project that uses another database. A CORBA expert might engineer interface definitions or services with all kinds of detail to anticipate evolution, just because he knows how. A Web designer on the front-end team might go crazy with the eye candy of the day on the user interface. None of these actions are inherently bad, and much of it is very valuable and clearly useful. At the end, however, if the project does not deliver what the customer wants with adequate performance and reliability, we have failed.

What if no one on your team has much experience with security? In a conflict between an area where we are somewhat lost and another where we can accomplish a significant amount of productive work, we pick the task where we will make the most progress. The problem arises with other facets of systems architecture as well, which might fall by the wayside because of a lack of experience or a lack of priority. Project teams declare that they cannot be highly available, cannot do thorough testing, or cannot do performance modeling because they do not have the time or the money to do so. This situation might often be the case, but if no one on the team has expertise building

reliable systems or regression testing suites or queuing theoretic models of service, then human nature might drive behavior away from these tasks.

Security architecture often suffers from this syndrome. Fortunately, we have a solution to our knowledge gap: Buy security and hire experts to secure our system. This point is where vendors come in to help us integrate their solutions into our applications.

## Vendor Security Products

The Internet boom has also driven the growth of security standards and technologies. Software vendors provide feature-rich security solutions and components at a level of complexity and maturity beyond almost all projects. Building our own components is rarely an option, and security architecture work is primarily integration work. In today's environment, the emerging dominance of vendor products aiding software development for enterprise security cannot be ignored.

We interact with vendors on many levels, and our understanding of their product offerings depends on a combination of information from many sources: marketing, sales, customer service support, vendor architects, and other applications with experience with the product. We have to be careful when viewing the entire application from the perspective of the security vendor. Looking at the application through a fisheye lens to get a wide-angle view could give us a warped perspective, with all of the elements of the system distorted around one central component: their security product. Here are three architectural flaws in vendor products:

**The product enjoys a central place in the architecture.** The product places itself at the center of the universe, which might not be where you, as the architect, would place it.

**The product hides assumptions.** The product hides assumptions that are critical to a successful deployment or does not articulate these assumptions as clear architectural prerequisites and requirements to the project.

**The context behind the product is unclear.** Context describes the design philosophy behind the purpose and placement of the product in some market niche. What is the history of the company with respect to building this particular security product? The vendor might be the originator of the technology, might have diversified into the product space, acquired a smaller company with expertise in the security area, or might have a strong background in a particular competing design philosophy.

Vendors have advantages over architects.

- They tend to have comparatively greater security expertise.
- They often do not tell architects about gaps in their own product's design voluntarily. You have to ask specific questions about product features.
- They rarely present their products in terms clearly comparable with those of their competitors. Project teams have to expend effort in understanding the feature sets well enough to do so themselves.

- They deflect valid criticism of holes in their design by assigning resolution responsibility to the user, administrator, application process, or other side of an interface, and so on.
- They rarely support the evolution path of an application over a two- to three-year timeframe.

This book is meant to swing the advantage back in the architect's court. We will describe how projects can evaluate vendor products, discover limitations and boundaries within solutions, and overcome them. Vendors are not antagonistic to the project's objectives, but miscommunication during vendor management might cause considerable friction as the application evolves and we learn more about real-world deployment issues surrounding the product. Building a good relationship between application architect and lead vendor engineers is critical and holds long-run benefits for the project and vendor alike. We hope that better information will lead to better decisions on security architecture.

## Our Goals in Writing This Book

---

On a first level, we will present an overview of the software process behind systems architecture. We focus on the architecture review, a checkpoint within the software development cycle that gives the project an opportunity to validate the solution architecture and verify that it meets requirements. We will describe how to assess a system for security issues, how to organize the architecture to add security as a system feature, and how to provide architectural context information that will help minimize the impact of implementing one security choice over another. We emphasize including security early in the design cycle instead of waiting until the application is in production and adding security as an afterthought.

On a second level, this book will provide hands-on help in understanding common, repeating patterns of design in the vast array of security products available. This book will help describe the vocabulary used surrounding security products as applied to systems architecture. We borrow the term *patterns* from the Object Patterns design community but do not intend to use the term beyond its common-sense meaning. Specifically, something is a security pattern if we can give it a name, observe its design appearing repeatedly in many security products, and see some benefit in defining and describing the pattern.

On a third level, we describe common security architecture issues and talk about security issues for specific technologies. We use three layers of application granularity to examine security.

- Low-level issues regarding code review, cryptographic primitives, and trusting code.
- Mid-level issues regarding middleware or Web services, operating systems, hosts, and databases.
- High-level issues regarding security components, conflicts between security and other system goals, and enterprise security.

On the fourth and final level, we discuss security from a financial standpoint. How can we justify the expense of securing our application?

## Reading This Book

We have organized the book into five parts, and aside from the chapters in Part I, any chapter can be read on its own. We would recommend that readers with specific interests and skills try the following tracks, however:

**Project and software process managers.** Begin by reading Chapters 1, 2, 3, 4, and 15. These chapters present vocabulary and basic concerns surrounding security architecture.

**Security assessors.** Begin by reading Chapters 1, 2, 3, 4, 13, and 14. Much of the information needed to sit in a review and understand the presentation is described there.

**Developers.** Read Chapters 1 through 4 in order and then Chapters 5 through 12 in any order—looking for the particular platform or software component that you are responsible for developing.

**Systems architects.** Read the book from start to finish, one complete part at a time. The presentation order, from Process to Technology to Enterprise concerns, parallels the requirements of systems architecture for a large application. All of these topics are now considered part of the domain of software architects.

**Business executives.** Read Chapters 1, 2, 16, and 17 for a start and then continue as your interests guide you with anything in between.

## Outline of the Book

---

Each chapter is a mix of the abstract and the concrete. For more detail on any technical matter, please see the list of bibliographic references at the end of the book. Each chapter will also contain questions to ask at an architecture review on a specific subject.

**Part I, Architecture and Security,** introduces the business processes of architecture review and security assessments. We describe the basics of security architecture and a catalog of security patterns.

*Chapter 1, “Architecture Reviews,”* describes a key checkpoint in the software development cycle where architects can ask and answer the question, “Does the solution fit the problem?” We present a description of the review process along with its benefits.

*Chapter 2, “Security Assessments,”* defines the process of security assessment by using the Federal Information Technology Security Assessment Framework along with other industry standards. We describe how assessments realize many of the benefits of architecture reviews within the specific context of security.

*Chapter 3, “Security Architecture Basics,”* defines the concept of assurance. We describe the concepts of authentication, authorization, access control, auditing, confidentiality, integrity, and nonrepudiation from an architectural viewpoint. We discuss other security properties and models of access control.

*Chapter 4, “Architecture Patterns in Security,”* defines the terms architectural style and pattern and describes how each of the basic security architecture requirements in the previous chapter lead to common implementation patterns. We also present a catalog of security patterns.

**Part II, Low-Level Architecture,** describes common issues surrounding developing secure software at the code level. We introduce the basics of cryptography and discuss its application in trusting code and in communications security protocols.

*Chapter 5, “Code Review,”* discusses the importance of code review from a security viewpoint. We describe buffer overflow exploits, one of the most common sources of security vulnerabilities. We discuss strategies for preventing exploits based on this attack. We also discuss security in Perl and the Java byte code verifier.

*Chapter 6, “Cryptography,”* introduces cryptographic primitives and protocols and the difficulty an architect faces in constructing and validating the same. We present guidelines for using cryptography.

*Chapter 7, “Trusted Code,”* discusses one consequence of the growth of the Web: the emergence of digitally delivered software. We describe the risks of downloading active content over the Internet, some responses to mitigating this risk, and why code is hard to trust.

*Chapter 8, “Secure Communications,”* introduces two methods for securing sessions—the SSL protocol and IPSec—and discusses the infrastructure support needed to implement such protocols. We discuss security layering and describe why there is plenty of security work left to be done at the application level.

**Part III, Mid-Level Architecture,** introduces common issues faced by application architects building security into their systems from a component and connector viewpoint.

*Chapter 9, “Middleware Security,”* discusses the impact of platform independence, a central goal of middleware products, on security. We describe the CORBA security specification, its service modules, and the various levels of CORBA-compliant security and administrative support. We also discuss other middleware security products at a high level.

*Chapter 10, “Web Security,”* is a short introduction to Web security from an architecture viewpoint, including information on security for standards such as J2EE.

*Chapter 11, “Application and OS Security,”* reviews the components that go into the design of an application, including OS security, network services, process descriptions, interface definitions, process flow diagrams, workflow maps, and administration tools. We discuss operating systems hardening and other deployment and development issues with building secure production applications. We also discuss UNIX ACLs.

*Chapter 12, “Database Security,”* introduces the state-of-the-art in database security architecture. We discuss the evolution of databases from a security standpoint and describe several models of securing persistent data. We also discuss the security features within Oracle, a leading commercial database product.

**Part IV, High-Level Architecture,** introduces common issues faced by enterprise architects charged with guiding software architecture discipline across many individual applications, all sharing some “enterprise” characteristic, such as being components of a high-level business process or domain.

*Chapter 13, “Security Components,”* discusses the building blocks available to systems architects and some guidelines for their usage. The list includes single sign-on servers, PKI, firewalls, network intrusion detection, directories, along with audit and security management products. We discuss issues that architects should or should not worry about and components they should or should not try to use. We also discuss the impact of new technologies like mobile devices that cause unique security integration issues for architects.

*Chapter 14, “Security and Other Architectural Goals,”* discusses the myths and realities about conflicts between security and other architectural goals. We discuss the impact of security on other goals such as performance, high availability, robustness, scalability, interoperability, maintainability, portability, ease of use, adaptability, and evolution. We conclude with guidelines for recognizing conflicts in the architecture, setting priorities, and deciding which goal wins.

*Chapter 15, “Enterprise Security Architecture,”* discusses the question, “How do we architect security and security management across applications?” We discuss the assets stored in the enterprise and the notion of database-of-record status. We also discuss common issues with enterprise infrastructure needs for security, such as user management, corporate directories, and legacy systems. We present and defend the thesis that enterprise security architecture is above all a data-management problem and propose a resolution using XML-based standards.

**Part V, Business Cases for Security,** introduces common issues faced by architects making a business case for security for their applications.

*Chapter 16, “Building Business Cases for Security,”* asks why it is hard to build business cases for security. We present the *Saved Losses Model* for justifying security business cases. We assign value to down time, loss of revenue, and reputation and assess the costs of guarding against loss. We discuss the role of an architect in incident prevention, industry information about costs, and the reconstruction of events across complex, distributed environments in a manner that holds water in a court of law. We ask whether security is insurable in the sense that we can buy hacker insurance that works like life insurance or fire insurance and discuss the properties that make something insurable.

*Chapter 17, “Conclusion,”* reviews security architecture lessons that we learned. We present some advice and further resources for architects.

We conclude with a bibliography of resources for architects and a glossary of acronyms.

## Online Information

---

Although we have reviewed the book and attempted to remove any technical errors, some surely remain. Readers with comments, questions, or bug fixes can email me at [book@jay-ramachandran.com](mailto:book@jay-ramachandran.com) or visit my Web site at [www.jay-ramachandran.com](http://www.jay-ramachandran.com) for Web links referred to in the text, updated vendor product information, or other information.

## Conclusion

---

A note before we start: although it might seem that way sometimes, our intent is not to present vendors and their security offerings as in constant conflict with your application and its objectives and needs. Security vendors provide essential services, and no discussion of security will be complete without recognition of their value and the role that their products play.

Security is commonly presented as a conflict between the good and the bad, with our application on one hand and the evil hacker on the other. This dichotomy is analogous to describing the application as a medieval castle and describing its defense: “Put the moat here,” “Make it yea deep,” “Use more than one portcullis,” “Here’s where you boil the oil,” “Here’s how you recognize a ladder propped against the wall,” and so on. This view presents security as an active conflict, and we often use the terms of war to describe details. In this case, we view ourselves as generals in the battle and our opponents as Huns (my apologies if you are a Hun, I’m just trying to make a point here).

Our basic goal is to frame the debate about systems security around a different dichotomy, one that recognizes that the castle also has a role in peacetime, as a market place for the surrounding villages, as the seat of authority in the realm, as a cantonment for troops, and as a place of residence for its inhabitants. Think of the system’s architect as the mayor of the town who has hired a knight to assemble a standing army for its defense. The knight knows warfare, but the mayor has the money. Note that we said that the architect is the mayor and not the king—that would be the customer.



# ACKNOWLEDGMENTS

I thank John Wiley & Sons for the opportunity to write this book, especially my editor, Carol Long. Carol read the proposal on a plane flight back from RSA 2001 and sent me a response the day she received it. From the start, Carol shared my perspective that security as seen by a software architect presents a unique and interesting viewpoint. I thank her for her belief in the idea. I thank my assistant editor, Adaobi Obi, for her careful reviews of the first draft and her many suggestions for improving the presentation. I thank my managing editor, Micheline Frederick, for her many ideas for improving the readability of the manuscript. I would also like to thank Radia Perlman for some valuable advice on the structure of this book at an early stage.

I thank the technical review team of Arun Iyer and Jai Chhugani for their excellent and insightful remarks, their thorough and careful chapter-by-chapter review, and many suggestions that have improved the text immeasurably. I also thank Steve Bellovin and Radia Perlman for reading the final draft of the manuscript. I am solely responsible for any errors and omissions that remain. Please visit my Web site [www.jay-ramachandran.com](http://www.jay-ramachandran.com) for the book for more information on security architecture, including Wiley's official links for the book, errata submissions, or permission requests.

I thank Tim Long, Don Aliberti, Alberto Avritzer, and Arun Iyer for their guidance in the past and for the many ideas and opinions that they offered me on security, architecture, and computer science. I am sure that the four of you will enjoy reading this book, because so much of it is based on stuff I learned from you in the conversations we have had.

I am heavily indebted to and thank the many security gurus, assessors, and developers I have had the pleasure of working with over the years on many systems, feasibility studies, applications, and consulting services. Their remarks and insight pepper this book: Steve Bellovin, Pete Bleznyk, Frank Carey, Juan Castillo, Dick Court, Joe DiBiase, Dave Gross, Daryl Hollar, Phil Hollemeak, Steve Meyer, Betsy Morgan, Shapour Neshatfar, Dino Perone, Bob Reed, Greg Rogers, George Schwerdtman, Gopi Shankavaram, Joyce Weekley, and Vivian Ye.

I thank Jane Bogdan, Dennis Holland, Brenda Liggs, and other members of the research staff at the Middletown Library for their assistance. I would also like to thank the staff of Woodbridge Public Library, my home away from home.

I am especially grateful to the brilliant and dedicated group of people at AT&T who call themselves certified software architects. You made my life as Architecture Review Coordinator so much easier. On my behalf and on behalf of all the projects you have helped, I

thank Janet Aromando, Mike Boaz, Jeff Bramley, Terry Cleary, Dave Cura, Bryon Donahue, Irwin Dunietz, John Eddy, Neal Fildes, Cindy Flaspohler, Tim Frommeyer, Don Gerth, Doug Ginn, Abhay Jain, Steve Meyer, Mike Newbury, Randy Ringeisen, Hans Ros, Ray Sandfoss, George Schwerdtman, Manoucher Shahrava, Mohammed Shakir, David Simen, Anoop Singhal, David Timidaiski, Tim Velten, and Dave Williamson.

Special thanks go to many friends and their families for countless hours over two decades spent debating all things under the sun, some of which related to computing and engineering. I thank Pankaj Agarwal, Alok Baveja, Paolo Bucci, Jai and Veena Chhugani, Anil and Punam Goel, Nilendu and Urmila Gupta, Nirmala Iyer, Aarati Kanekar, Atul and Manu Khare, K. Ananth Krishnan, Asish and Anandi Law, Pushkal Pandey, Sushant and Susan Patnaik, Mahendra Ramachandran, Ming Jye-Sheu, and Manoj and Neeta Tandon for their friendship.

This book would not exist but for my family. I thank my family, Jayashree, Akhila and Madhavan, Bhaskar and Vidyut, and especially Amma, Appa, Aai, and Daiyya for their blessings. Without their confidence, support, and help in so many ways, I could not have attempted let alone completed this task. Hats off to you all. To Ronak and Mallika, for their patience and humor, and last but not least, to Beena, for all the support in the world. You steered the ship through the storm while the first mate was down in the bilge thinking this book up. This book is for you.