Dr. Dobbs Jolt Award Finalist 2014

Adam Shostack
Microsoft's Threat Modeling Expert

threat modeling

designing for security

WILEY

Threat Modeling

Designing for Security

Adam Shostack

WILEY

Threat Modeling: Designing for Security

Published by
John Wiley & Sons, Inc.
10475 Crosspoint
BoulevardIndianapolis, IN 46256
www.wiley.com

Published by John Wiley & Sons, Inc., Indianapolis, Indiana
Published simultaneously in Canada

ISBN: 978-1-118-80999-0
ISBN: 978-1-118-82269-2 (ebk)
ISBN: 978-1-118-81005-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Library of Congress Control Number: 2013954095

For all those striving to deliver more secure systems

Credits

Executive Editor
Carol Long

Project Editors
Victoria Swider
Tom Dinse

Technical Editor
Chris Wysopal

Production Editor
Christine Mugnolo

Copy Editor
Luann Rouff

Editorial Manager
Mary Beth Wakefield

Freelancer Editorial Manager
Rosemarie Graham

Associate Director of Marketing
David Mayhew

Marketing Manager
Ashley Zurcher

Business Manager
Amy Knies

Vice President and Executive Group Publisher
Richard Swadley

Associate Publisher
Jim Minatel

Project Coordinator, Cover
Todd Klemme

Technical Proofreader
Russ McRee

Proofreader
Nancy Carrasco

Indexer
Robert Swanson

Cover Image
Courtesy of Microsoft

Cover Designer
Wiley

About the Author

Adam Shostack is currently a program manager at Microsoft. His security roles there have included security development processes, usable security, and attack modeling. His attack-modeling work led to security updates for Autorun being delivered to hundreds of millions of computers. He shipped the SDL Threat Modeling Tool and the *Elevation of Privilege* threat modeling game. While doing security development process work, he delivered threat modeling training across Microsoft and its partners and customers.

Prior to Microsoft, he has been an executive at a number of successful information security and privacy startups. He helped found the CVE, the Privacy Enhancing Technologies Symposium and the International Financial Cryptography Association. He has been a consultant to banks, hospitals and startups and established software companies. For the first several years of his career, he was a systems manager for a medical research lab. Shostack is a prolific author, blogger, and public speaker. With Andrew Stewart, he co-authored *The New School of Information Security* (Addison-Wesley, 2008).

About the Technical Editor

Chris Wysopal, Veracode's CTO and Co-Founder, is responsible for the company's software security analysis capabilities. In 2008 he was named one of InfoWorld's Top 25 CTO's and one of the 100 most influential people in IT by eWeek. One of the original vulnerability researchers and a member of L0pht Heavy Industries, he has testified on Capitol Hill in the US on the subjects of government computer security and how vulnerabilities are discovered in software. He is an author of L0phtCrack and netcat for Windows. He is the lead author of *The Art of Software Security Testing* (Addison-Wesley, 2006).

Acknowledgments

First and foremost, I'd like to thank countless engineers at Microsoft and elsewhere who have given me feedback about their experiences threat modeling. I wouldn't have had the opportunity to have so many open and direct conversations without the support of Eric Bidstrup and Steve Lipner, who on my first day at Microsoft told me to go "wallow in the problem for a while." I don't think either expected "a while" to be quite so long. Nearly eight years later with countless deliverables along the way, this book is my most complete answer to the question they asked me: "How can we get better threat models?"

Ellen Cram Kowalczyk helped me make the book a reality in the Microsoft context, gave great feedback on both details and aspects that were missing, and also provided a lot of the history of threat modeling from the first security pushes through the formation of the SDL, and she was a great manager and mentor. Ellen and Steve Lipner were also invaluable in helping me obtain permission to use Microsoft documents.

The *Elevation of Privilege* game that opens this book owes much to Jacqueline Beauchere, who saw promise in an ugly prototype called "Threat Spades," and invested in making it beautiful and widely available.

The SDL Threat Modeling Tool might not exist if Chris Peterson hadn't given me a chance to build a threat modeling tool for the Windows team to use. Ivan Medvedev, Patrick McCuller, Meng Li, and Larry Osterman built the first version of that tool. I'd like to thank the many engineers in Windows, and later across Microsoft, who provided bug reports and suggestions for improvements in the beta days, and acknowledge all those who just flamed at us, reminding us of the importance of getting threat modeling right. Without that tool, my experience and breadth in threat modeling would be far poorer.

Larry Osterman, Douglas MacIver, Eric Douglas, Michael Howard, and Bob Fruth gave me hours of their time and experience in understanding threat

modeling at Microsoft. Window Snyder's perspective as I started the Microsoft job has been invaluable over the years. Knowing when you're done . . . well, this book is nearly done.

Rob Reeder was a great guide to the field of usable security, and Chapter 15 would look very different if not for our years of collaboration. I can't discuss usable security without thanking Lorrie Cranor for her help on that topic; but also for the chance to keynote the *Symposium on Usable Privacy and Security*, which led me to think about usable engineering advice, a perspective that is now suffused throughout this book.

Andy Steingrubl, Don Ankney, and Russ McRee all taught me important lessons related to operational threat modeling, and how the trade-offs change as you change context. Guys, thank you for beating on me—those lessons now permeate many chapters. Alec Yasinac, Harold Pardue, and Jeff Landry were generous with their time discussing their attack tree experience, and Chapters 4 and 17 are better for those conversations. Joseph Lorenzo Hall was also a gem in helping with attack trees. Wendy Nather argued strongly that assets and attackers are great ways to make threats real, and thus help overcome resistance to fixing them. Rob Sama checked the Acme financials example from a CPA's perspective, correcting many of my errors. Dave Awksmith graciously allowed me to include his threat personas as a complete appendix. Jason Nehrboss gave me some of the best feedback I've ever received on very early chapters.

I'd also like to acknowledge Jacob Appelbaum, Crispin Cowan, Dana Epp (for years of help, on both the book and tools), Jeremi Gosney, Yoshi Kohno, David LeBlanc, Marsh Ray, Nick Mathewson, Tamara McBride, Russ McRee, Talhah Mir, David Mortman, Alec Muffet, Ben Rothke, Andrew Stewart, and Bryan Sullivan for helpful feedback on drafts and/or ideas that made it into the book in a wide variety of ways.

Of course, none of those acknowledged in this section are responsible for the errors which doubtless crept in or remain.

Writing this book "by myself" (an odd phrase given everyone I'm acknowledging) makes me miss working with Andrew Stewart, my partner in writing on *The New School of Information Security*. Especially since people sometimes attribute that book to me, I want to be public about how much I missed his collaboration in this project.

This book wouldn't be in the form it is were it not for Bruce Schneier's willingness to make an introduction to Carol Long, and Carol's willingness to pick up the book. It wasn't always easy to read the feedback and suggested changes from my excellent project editor, Victoria Swider, but this thing is better where I did. Tom Dinse stepped in as the project ended and masterfully took control of a very large number of open tasks, bringing them to resolution on a tight schedule.

Lastly, and most importantly, thank you to Terri, for all your help, support, and love, and for putting up with "it's almost done" for a very, very long time.

—Adam Shostack

Contents

Introduction		**xxi**
Part I	**Getting Started**	**1**
Chapter 1	**Dive In and Threat Model!**	**3**
	Learning to Threat Model	4
	What Are You Building?	5
	What Can Go Wrong?	7
	Addressing Each Threat	12
	Checking Your Work	24
	Threat Modeling on Your Own	26
	Checklists for Diving In and Threat Modeling	27
	Summary	28
Chapter 2	**Strategies for Threat Modeling**	**29**
	"What's Your Threat Model?"	30
	Brainstorming Your Threats	31
	Brainstorming Variants	32
	Literature Review	33
	Perspective on Brainstorming	34
	Structured Approaches to Threat Modeling	34
	Focusing on Assets	36
	Focusing on Attackers	40
	Focusing on Software	41
	Models of Software	43
	Types of Diagrams	44
	Trust Boundaries	50
	What to Include in a Diagram	52
	Complex Diagrams	52
	Labels in Diagrams	53

Color in Diagrams 53
Entry Points 53
Validating Diagrams 54
Summary 56

Part II Finding Threats 59

Chapter 3 STRIDE 61
Understanding STRIDE and Why It's Useful 62
Spoofing Threats 64
Spoofing a Process or File on the Same Machine 65
Spoofing a Machine 66
Spoofing a Person 66
Tampering Threats 67
Tampering with a File 68
Tampering with Memory 68
Tampering with a Network 68
Repudiation Threats 68
Attacking the Logs 69
Repudiating an Action 70
Information Disclosure Threats 70
Information Disclosure from a Process 71
Information Disclosure from a Data Store 71
Information Disclosure from a Data Flow 72
Denial-of-Service Threats 72
Elevation of Privilege Threats 73
Elevate Privileges by Corrupting a Process 74
Elevate Privileges through Authorization Failures 74
Extended Example: STRIDE Threats against Acme-DB 74
STRIDE Variants 78
STRIDE-per-Element 78
STRIDE-per-Interaction 80
DESIST 85
Exit Criteria 85
Summary 85

Chapter 4 Attack Trees 87
Working with Attack Trees 87
Using Attack Trees to Find Threats 88
Creating New Attack Trees 88
Representing a Tree 91
Human-Viewable Representations 91
Structured Representations 94
Example Attack Tree 94
Real Attack Trees 96
Fraud Attack Tree 96
Election Operations Assessment Threat Trees 96
Mind Maps 98

Perspective on Attack Trees 98
Summary 100

Chapter 5 Attack Libraries 101
Properties of Attack Libraries 101
Libraries and Checklists 103
Libraries and Literature Reviews 103
CAPEC 104
Exit Criteria 106
Perspective on CAPEC 106
OWASP Top Ten 108
Summary 108

Chapter 6 Privacy Tools 111
Solove's Taxonomy of Privacy 112
Privacy Considerations for Internet Protocols 114
Privacy Impact Assessments (PIA) 114
The Nymity Slider and the Privacy Ratchet 115
Contextual Integrity 117
Contextual Integrity Decision Heuristic 118
Augmented Contextual Integrity Heuristic 119
Perspective on Contextual Integrity 119
LINDDUN 120
Summary 121

Part III Managing and Addressing Threats 123

Chapter 7 Processing and Managing Threats 125
Starting the Threat Modeling Project 126
When to Threat Model 126
What to Start and (Plan to) End With 128
Where to Start 128
Digging Deeper into Mitigations 130
The Order of Mitigation 131
Playing Chess 131
Prioritizing 132
Running from the Bear 132
Tracking with Tables and Lists 133
Tracking Threats 133
Making Assumptions 135
External Security Notes 136
Scenario-Specific Elements of Threat Modeling 138
Customer/Vendor Trust Boundary 139
New Technologies 139
Threat Modeling an API 141
Summary 143

Chapter 8 Defensive Tactics and Technologies 145
Tactics and Technologies for Mitigating Threats 145
Authentication: Mitigating Spoofing 146
Integrity: Mitigating Tampering 148
Non-Repudiation: Mitigating Repudiation 150
Confidentiality: Mitigating Information Disclosure 153
Availability: Mitigating Denial of Service 155
Authorization: Mitigating Elevation of Privilege 157
Tactic and Technology Traps 159
Addressing Threats with Patterns 159
Standard Deployments 160
Addressing CAPEC Threats 160
Mitigating Privacy Threats 160
Minimization 160
Cryptography 161
Compliance and Policy 164
Summary 164

Chapter 9 Trade-Offs When Addressing Threats 167
Classic Strategies for Risk Management 168
Avoiding Risks 168
Addressing Risks 168
Accepting Risks 169
Transferring Risks 169
Ignoring Risks 169
Selecting Mitigations for Risk Management 170
Changing the Design 170
Applying Standard Mitigation Technologies 174
Designing a Custom Mitigation 176
Fuzzing Is Not a Mitigation 177
Threat-Specific Prioritization Approaches 178
Simple Approaches 178
Threat-Ranking with a Bug Bar 180
Cost Estimation Approaches 181
Mitigation via Risk Acceptance 184
Mitigation via Business Acceptance 184
Mitigation via User Acceptance 185
Arms Races in Mitigation Strategies 185
Summary 186

Chapter 10 Validating That Threats Are Addressed 189
Testing Threat Mitigations 190
Test Process Integration 190
How to Test a Mitigation 191
Penetration Testing 191

Checking Code You Acquire 192
Constructing a Software Model 193
Using the Software Model 194
QA'ing Threat Modeling 195
Model/Reality Conformance 195
Task and Process Completion 196
Bug Checking 196
Process Aspects of Addressing Threats 197
Threat Modeling Empowers Testing; Testing Empowers Threat Modeling 197
Validation/Transformation 197
Document Assumptions as You Go 198
Tables and Lists 198
Summary 202

Chapter 11 Threat Modeling Tools 203
Generally Useful Tools 204
Whiteboards 204
Office Suites 204
Bug-Tracking Systems 204
Open-Source Tools 206
TRIKE 206
SeaMonster 206
Elevation of Privilege 206
Commercial Tools 208
ThreatModeler 208
Corporate Threat Modeller 208
Secur*IT*ree 209
Little-JIL 209
Microsoft's SDL Threat Modeling Tool 209
Tools That Don't Exist Yet 213
Summary 213

Part IV Threat Modeling in Technologies and Tricky Areas 215

Chapter 12 Requirements Cookbook 217
Why a "Cookbook"? 218
The Interplay of Requirements, Threats, and Mitigations 219
Business Requirements 220
Outshining the Competition 220
Industry Requirements 220
Scenario-Driven Requirements 221
Prevent/Detect/Respond as a Frame for Requirements 221
Prevention 221
Detection 225

Response 225
People/Process/Technology as a Frame for Requirements 227
People 227
Process 228
Technology 228
Development Requirements vs. Acquisition Requirements 228
Compliance-Driven Requirements 229
Cloud Security Alliance 229
NIST Publication 200 230
PCI-DSS 231
Privacy Requirements 231
Fair Information Practices 232
Privacy by Design 232
The Seven Laws of Identity 233
Microsoft Privacy Standards for Development 234
The STRIDE Requirements 234
Authentication 235
Integrity 236
Non-Repudiation 237
Confidentiality 238
Availability 238
Authorization 239
Non-Requirements 240
Operational Non-Requirements 240
Warnings and Prompts 241
Microsoft's "10 Immutable Laws" 241
Summary 242

Chapter 13 Web and Cloud Threats 243
Web Threats 243
Website Threats 244
Web Browser and Plugin Threats 244
Cloud Tenant Threats 246
Insider Threats 246
Co-Tenant Threats 247
Threats to Compliance 247
Legal Threats 248
Threats to Forensic Response 248
Miscellaneous Threats 248
Cloud Provider Threats 249
Threats Directly from Tenants 249
Threats Caused by Tenant Behavior 250
Mobile Threats 250
Summary 251

Chapter 14 Accounts and Identity 253
Account Life Cycles 254
Account Creation 254
Account Maintenance 257
Account Termination 258
Account Life-Cycle Checklist 258
Authentication 259
Login 260
Login Failures 262
Threats to "What You Have" 263
Threats to "What You Are" 264
Threats to "What You Know" 267
Authentication Checklist 271
Account Recovery 271
Time and Account Recovery 272
E-mail for Account Recovery 273
Knowledge-Based Authentication 274
Social Authentication 278
Attacker-Driven Analysis of Account Recovery 280
Multi-Channel Authentication 281
Account Recovery Checklist 281
Names, IDs, and SSNs 282
Names 282
Identity Documents 285
Social Security Numbers and Other National Identity Numbers 286
Identity Theft 289
Names, IDs, and SSNs Checklist 290
Summary 290

Chapter 15 Human Factors and Usability 293
Models of People 294
Applying Behaviorist Models of People 295
Cognitive Science Models of People 297
Heuristic Models of People 302
Models of Software Scenarios 304
Modeling the Software 304
Diagramming for Modeling the Software 307
Modeling Electronic Social Engineering Attacks 309
Threat Elicitation Techniques 311
Brainstorming 311
The Ceremony Approach to Threat Modeling 311
Ceremony Analysis Heuristics 312
Integrating Usability into the Four-Stage Framework 315

Tools and Techniques for Addressing Human Factors 316
Myths That Inhibit Human Factors Work 317
Design Patterns for Good Decisions 317
Design Patterns for a Kind Learning Environment 320
User Interface Tools and Techniques 322
Configuration 322
Explicit Warnings 323
Patterns That Grab Attention 325
Testing for Human Factors 327
Benign and Malicious Scenarios 328
Ecological Validity 328
Perspective on Usability and Ceremonies 329
Summary 331

Chapter 16 Threats to Cryptosystems 333
Cryptographic Primitives 334
Basic Primitives 334
Privacy Primitives 339
Modern Cryptographic Primitives 339
Classic Threat Actors 341
Attacks against Cryptosystems 342
Building with Crypto 346
Making Choices 346
Preparing for Upgrades 346
Key Management 346
Authenticating before Decrypting 348
Things to Remember about Crypto 348
Use a Cryptosystem Designed by Professionals 348
Use Cryptographic Code Built and Tested by Professionals 348
Cryptography Is Not Magic Security Dust 349
Assume It Will All Become Public 349
You Still Need to Manage Keys 349
Secret Systems: Kerckhoffs and His Principles 349
Summary 351

Part V Taking It to the Next Level 353

Chapter 17 Bringing Threat Modeling to Your Organization 355
How To Introduce Threat Modeling 356
Convincing Individual Contributors 357
Convincing Management 358
Who Does What? 359
Threat Modeling and Project Management 359

Prerequisites 360
Deliverables 360
Individual Roles and Responsibilities 362
Group Interaction 363
Diversity in Threat Modeling Teams 367
Threat Modeling within a Development Life Cycle 367
Development Process Issues 368
Organizational Issues 373
Customizing a Process for Your Organization 378
Overcoming Objections to Threat Modeling 379
Resource Objections 379
Value Objections 380
Objections to the Plan 381
Summary 383

Chapter 18 Experimental Approaches 385
Looking in the Seams 386
Operational Threat Models 387
FlipIT 388
Kill Chains 388
The "Broad Street" Taxonomy 392
Adversarial Machine Learning 398
Threat Modeling a Business 399
Threats to Threat Modeling Approaches 400
Dangerous Deliverables 400
Enumerate All Assumptions 400
Dangerous Approaches 402
How to Experiment 404
Define a Problem 404
Find Aspects to Measure and Measure Them 404
Study Your Results 405
Summary 405

Chapter 19 Architecting for Success 407
Understanding Flow 407
Flow and Threat Modeling 409
Stymieing People 411
Beware of Cognitive Load 411
Avoid Creator Blindness 412
Assets and Attackers 412
Knowing the Participants 413
Boundary Objects 414
The Best Is the Enemy of the Good 415
Closing Perspectives 416
"The Threat Model Has Changed" 417

On Artistry 418
Summary 419
Now Threat Model 420

Appendix A Helpful Tools 421
Common Answers to "What's Your Threat Model?" 421
Network Attackers 421
Physical Attackers 422
Attacks against People 423
Supply Chain Attackers 423
Privacy Attackers 424
Non-Sentient "Attackers" 424
The Internet Threat Model 424
Assets 425
Computers as Assets 425
People as Assets 426
Processes as Assets 426
Intangible Assets 427
Stepping-Stone Assets 427

Appendix B Threat Trees 429
STRIDE Threat Trees 430
Spoofing an External Entity (Client/ Person/Account) 432
Spoofing a Process 438
Spoofing of a Data Flow 439
Tampering with a Process 442
Tampering with a Data Flow 444
Tampering with a Data Store 446
Repudiation against a Process (or by an External Entity) 450
Repudiation, Data Store 452
Information Disclosure from a Process 454
Information Disclosure from a Data Flow 456
Information Disclosure from a Data Store 459
Denial of Service against a Process 462
Denial of Service against a Data Flow 463
Denial of Service against a Data Store 466
Elevation of Privilege against a Process 468
Other Threat Trees 470
Running Code 471
Attack via a "Social" Program 474
Attack with Tricky Filenames 476

Appendix C Attacker Lists 477
Attacker Lists 478
Barnard's List 478
Verizon's Lists 478
OWASP 478
Intel TARA 479

Personas and Archetypes 480
Aucsmith's Attacker Personas 481
Background and Definitions 481
Personas 484
David "Ne0phyate" Bradley – Vandal 484
JoLynn "NightLily" Dobney – Trespasser 486
Sean "Keech" Purcell – Defacer 488
Bryan "CrossFyre" Walton – Author 490
Lorrin Smith-Bates – Insider 492
Douglas Hite – Thief 494
Mr. Smith – Terrorist 496
Mr. Jones – Spy 498

Appendix D *Elevation of Privilege:* The Cards **501**
Spoofing 501
Tampering 503
Repudiation 504
Information Disclosure 506
Denial of Service 507
Elevation of Privilege (EoP) 508

Appendix E Case Studies **511**
The Acme Database 512
Security Requirements 512
Software Model 512
Threats and Mitigations 513
Acme's Operational Network 519
Security Requirements 519
Operational Network 520
Threats to the Network 521
Phones and One-Time Token Authenticators 525
The Scenario 526
The Threats 527
Possible Redesigns 528
Sample for You to Model 528
Background 529
The iNTegrity Data Flow Diagrams 530
Exercises 531

Glossary **533**

Bibliography **543**

Index **567**

Introduction

All models are wrong, some models are useful.

— George Box

This book describes the useful models you can employ to address or mitigate these potential threats. People who build software, systems, or things with software need to address the many predictable threats their systems can face.

Threat modeling is a fancy name for something we all do instinctively. If I asked you to threat model your house, you might start by thinking about the precious things within it: your family, heirlooms, photos, or perhaps your collection of signed movie posters. You might start thinking about the ways someone might break in, such as unlocked doors or open windows. And you might start thinking about the sorts of people who might break in, including neighborhood kids, professional burglars, drug addicts, perhaps a stalker, or someone trying to steal your Picasso original.

Each of these examples has an analog in the software world, but for now, the important thing is not how you guard against each threat, but that you're able to relate to this way of thinking. If you were asked to help assess a friend's house, you could probably help, but you might lack confidence in how complete your analysis is. If you were asked to secure an office complex, you might have a still harder time, and securing a military base or a prison seems even more difficult. In those cases, your instincts are insufficient, and you'd need tools to help tackle the questions. This book will give you the tools to think about threat modeling technology in structured and effective ways.

In this introduction, you'll learn about what threat modeling is and why individuals, teams, and organizations threat model. Those reasons include finding security issues early, improving your understanding of security requirements, and being able to engineer and deliver better products. This introduction has

five main sections describing what the book is about, including a definition of threat modeling and reasons it's important; who should read this book; how to use it, and what you can expect to gain from the various parts, and new lessons in threat modeling.

What Is Threat Modeling?

Everyone threat models. Many people do it out of frustration in line at the airport, sneaking out of the house or into a bar. At the airport, you might idly consider how to sneak something through security, even if you have no intent to do so. Sneaking in or out of someplace, you worry about who might catch you. When you speed down the highway, you work with an implicit threat model where the main threat is the police, who you probably think are lurking behind a billboard or overpass. Threats of road obstructions, deer, or rain might play into your model as well.

When you threat model, you usually use two types of models. There's a model of what you're building, and there's a model of the threats (what can go wrong). What you're building with software might be a website, a downloadable program or app, or it might be delivered in a hardware package. It might be a distributed system, or some of the "things" that will be part of the "Internet of things." You model so that you can look at the forest, not the trees. A good model helps you address classes or groups of attacks, and deliver a more secure product.

The English word *threat* has many meanings. It can be used to describe a person, such as "Osama bin Laden was a threat to America," or people, such as "the insider threat." It can be used to describe an event, such as "There is a threat of a hurricane coming through this weekend," and it can be used to describe a weakness or possibility of attack, such as "What are you doing about confidentiality threats?" It is also used to describe viruses and malware such as "This threat incorporates three different methods for spreading." It can be used to describe behavior such as "There's a threat of operator error."

Similarly, the term *threat modeling* has many meanings, and the term *threat model* is used in many distinct and perhaps incompatible ways, including:

- As a verb—for example, "Have you threat modeled?" That is, have you gone through an analysis process to figure out what might go wrong with the thing you're building?
- As a noun, to ask what threat model is being used. For example, "Our threat model is someone in possession of the machine," or "Our threat model is a skilled and determined remote attacker."
- It can mean building up a set of idealized attackers.
- It can mean abstracting threats into classes such as tampering.

There are doubtless other definitions. All of these are useful in various scenarios and thus correct, and there are few less fruitful ways to spend your time

than debating them. Arguing over definitions is a strange game, and the only way to win is not to play. This book takes a big tent approach to threat modeling and includes a wide range of techniques you can apply early to make what you're designing or building more secure. It will also address the reality that some techniques are more effective than others, and that some techniques are more likely to work for people with particular skills or experience.

Threat modeling is the key to a focused defense. Without threat models, you can never stop playing whack-a-mole.

In short, threat modeling is the use of abstractions to aid in thinking about risks.

Reasons to Threat Model

In today's fast-paced world, there is a tendency to streamline development activity, and there are important reasons to threat model, which are covered in this section. Those include finding security bugs early, understanding your security requirements, and engineering and delivering better products.

Find Security Bugs Early

If you think about building a house, decisions you make early will have dramatic effects on security. Wooden walls and lots of ground-level windows expose you to more risks than brick construction and few windows. Either may be a reasonable choice, depending on where you're building and other factors. Once you've chosen, changes will be expensive. Sure, you can put bars over your windows, but wouldn't it be better to use a more appropriate design from the start? The same sorts of tradeoffs can apply in technology. Threat modeling will help you find design issues even before you've written a line of code, and that's the best time to find those issues.

Understand Your Security Requirements

Good threat models can help you ask "Is that really a requirement?" For example, does the system need to be secure against someone in physical possession of the device? Apple has said yes for the iPhone, which is different from the traditional world of the PC. As you find threats and triage what you're going to do with them, you clarify your requirements. With more clear requirements, you can devote your energy to a consistent set of security features and properties.

There is an important interplay between requirements, threats, and mitigations. As you model threats, you'll find that some threats don't line up with your business requirements, and as such may not be worth addressing. Alternately, your requirements may not be complete. With other threats, you'll find that addressing them is too complex or expensive. You'll need to make a call between

addressing them partially in the current version or accepting (and communicating) that you can't address those threats.

Engineer and Deliver Better Products

By considering your requirements and design early in the process, you can dramatically lower the odds that you'll be re-designing, re-factoring, or facing a constant stream of security bugs. That will let you deliver a better product on a more predictable schedule. All the effort that would go to those can be put into building a better, faster, cheaper or more secure product. You can focus on whatever properties your customers want.

Address Issues Other Techniques Won't

The last reason to threat model is that threat modeling will lead you to categories of issues that other tools won't find. Some of these issues will be errors of omission, such as a failure to authenticate a connection. That's not something that a code analysis tool will find. Other issues will be unique to your design. To the extent that you have a set of smart developers building something new, you might have new ways threats can manifest. Models of what goes wrong, by abstracting away details, will help you see analogies and similarities to problems that have been discovered in other systems.

A corollary of this is that threat modeling should not focus on issues that your other safety and security engineering is likely to find (except insofar as finding them early lets you avoid re-engineering). So if, for example, you're building a product with a database, threat modeling might touch quickly on SQL injection attacks, and the variety of trust boundaries that might be injectable. However, you may know that you'll encounter those. Your threat modeling should focus on issues that other techniques can't find.

Who Should Read This book?

This book is written for those who create or operate complex technology. That's primarily software engineers and systems administrators, but it also includes a variety of related roles, including analysts or architects. There's also a lot of information in here for security professionals, so this book should be useful to them and those who work with them. Different parts of the book are designed for different people—in general, the early chapters are for generalists (or specialists in something other than security), while the end of the book speaks more to security specialists.

You don't need to be a security expert, professional, or even enthusiast to get substantial benefit from this book. I assume that you understand that there are people out there whose interests and desires don't line up with yours. For example, maybe they'd like to take money from you, or they may have other goals, like puffing themselves up at your expense or using your computer to attack other people.

This book is written in plain language for anyone who can write or spec a program, but sometimes a little jargon helps in precision, conciseness, or clarity, so there's a glossary.

What You Will Gain from This Book

When you read this book cover to cover, you will gain a rich knowledge of threat modeling techniques. You'll learn to apply those techniques to your projects so you can build software that's more secure from the get-go, and deploy it more securely. You'll learn to how to make security tradeoffs in ways that are considered, measured, and appropriate. You will learn a set of tools and when to bring them to bear. You will discover a set of glamorous distractions. Those distractions might seem like wonderful, sexy ideas, but they hide an ugly interior. You'll learn why they prevent you from effectively threat modeling, and how to avoid them.

You'll also learn to focus on the actionable outputs of threat modeling, and I'll generally call those "bugs." There are arguments that it's helpful to consider code issues as bugs, and design issues as flaws. In my book, those arguments are a distraction; you should threat model to find issues that you can address, and arguing about labels probably doesn't help you address them.

Lessons for Different Readers

This book is designed to be useful to a wide variety of people working in technology. That includes a continuum from those who develop software to those who combine it into systems that meet operational or business goals to those who focus on making it more secure.

For convenience, this book pretends there is a bright dividing line between development and operations. The distinction is used as a way of understanding who has what capabilities, choices, and responsibilities. For example, it is "easy" for a developer to change what is logged, or to implement a different authentication system. Both of these may be hard for operations. Similarly, it's "easy" for operations to ensure that logs are maintained, or to ensure that a computer is in a locked cage. As this book was written, there's also an important

model of "devops" emerging. The lessons for developers and operations can likely be applied with minor adjustments. This book also pretends that security expertise is separate from either development or operations expertise, again, simply as a convenience.

Naturally, this means that the same parts of the book will bring different lessons for different people. The breakdown below gives a focused value proposition for each audience.

Software Developers and Testers

Software developers—those whose day jobs are focused on creating software—include software engineers, quality assurance, and a variety of program or project managers. If you're in that group, you will learn to find and address design issues early in the software process. This book will enable you to deliver more secure software that better meets customer requirements and expectations. You'll learn a simple, effective and fun approach to threat modeling, as well as different ways to model your software or find threats. You'll learn how to track threats with bugs that fit into your development process. You'll learn to use threats to help make your requirements more crisp, and vice versa. You'll learn about areas such as authentication, cryptography, and usability where the interplay of mitigations and attacks has a long history, so you can understand how the recommended approaches have developed to their current state. You'll learn about how to bring threat modeling into your development process. And a whole lot more!

Systems Architecture, Operations, and Management

For those whose day jobs involve bringing together software components, weaving them together into systems to deliver value, you'll learn to find and address threats as you design your systems, select your components, and get them ready for deployment. This book will enable you to deliver more secure systems that better meet business, customer, and compliance requirements. You'll learn a simple, effective, and fun approach to threat modeling, as well as different ways to model the systems you're building or have built. You'll learn how to find security and privacy threats against those systems. You'll learn about the building blocks which are available for you to operationally address those threats. You'll learn how to make tradeoffs between the threats you face, and how to ensure that those threats are addressed. You'll learn about specific threats to categories of technology, such as web and cloud systems, and about threats to accounts, both of which are deeply important to those in operations. It will cover issues of usability, and perhaps even change your perspective on

how to influence the security behavior of people within your organization and/or your customers. You will learn about cryptographic building blocks, which you may be using to protect systems. And a whole lot more!

Security Professionals

If you work in security, you will learn two major things from this book: First, you'll learn structured approaches to threat modeling that will enhance your productivity, and as you do, you'll learn why many of the "obvious" parts of threat modeling are not as obvious, or as right, as you may have believed. Second, you'll learn about bringing security into the development, operational and release processes that your organization uses.

Even if you are an expert, this book can help you threat model better. Here, I speak from experience. As I was writing the case study appendix, I found myself turning to both the tree in Appendix B and the requirements chapter, and finding threats that didn't spring to mind from just considering the models of software.

TO MY COLLEAGUES IN INFORMATION SECURITY

I want to be frank. This book is not about how to design abstractly perfect software. It is a practical, grounded book that acknowledges that most software is built in some business or organizational reality that requires tradeoffs. To the dismay of purists, software where tradeoffs were made runs the world these days, and I'd like to make such software more secure by making those tradeoffs better. That involves a great many elements, two of which are making security more consistent and more accessible to our colleagues in other specialties.

This perspective is grounded in my time as a systems administrator, deploying security technologies, and observing the issues people encountered. It is grounded in my time as a startup executive, learning to see security as a property of a system which serves a business goal. It is grounded in my responsibility for threat modeling as part of Microsoft's Security Development Lifecycle. In that last role, I spoke with thousands of people at Microsoft, its partners, and its customers about our approaches. These individuals ranged from newly hired developers to those with decades of experience in security, and included chief security officers and Microsoft's Trustworthy Computing Academic Advisory Board. I learned that there are an awful lot of opinions about what works, and far fewer about what does not. This book aims to convince my fellow security professionals that pragmatism in what we ask of development and operations helps us deliver more secure software over time. This perspective may be a challenge for some security professionals. They should focus on Parts II, IV, and V, and perhaps give consideration to the question of the best as the enemy of the good.

How To Use This Book

You should start at the very beginning. It's a very good place to start, even if you already know how to threat model, because it lays out a framework that will help you understand the rest of the book.

The Four-Step Framework

This book introduces the idea that you should see threat modeling as composed of steps which accomplish subgoals, rather than as a single activity. The essential questions which you ask to accomplish those subgoals are:

1. What are you building?
2. What can go wrong with it once it's built?
3. What should you do about those things that can go wrong?
4. Did you do a decent job of analysis?

The methods you use in each step of the framework can be thought of like Lego blocks. When working with Legos, you can snap in other Lego blocks. In Chapter 1, you'll use a data flow diagram to model what you're building, STRIDE to help you think about what can go wrong and what you should do about it, and a checklist to see if you did a decent job of analysis. In Chapter 2, you'll see how diagrams are the most helpful way to think about what you're building. Different diagram types are like different building blocks to help you model what you're building. In Chapter 3, you'll go deep into STRIDE (a model of threats), while in Chapter 4, you'll learn to use attack trees instead of STRIDE, while leaving everything else the same. STRIDE and attack trees are different building blocks for considering what can go wrong once you've built your new technology.

Not every approach can snap with every other approach. It takes crazy glue to make an Erector set and Lincoln logs stick together. Attempts to glue threat modeling approaches together has made for some confusing advice. For example, trying to consider how terrorists would attack your assets doesn't really lead to a lot of actionable issues. And even with building blocks that snap together, you can make something elegant, or something confusing or bizarre.

So to consider this as a framework, what are the building blocks? The four-step framework is shown graphically in Figure I-1.

The steps are:

1. Model the system you're building, deploying, or changing.
2. Find threats using that model and the approaches in Part II.