

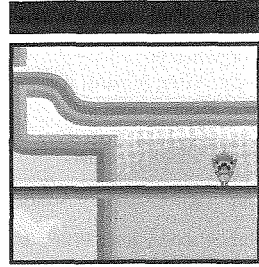
Dr. Dobbs Jolt Award Finalist

Adam Shostack
Microsoft's Threat Modeling Expert

threat modeling

designing for security





Threat Modeling

Designing for Security

Adam Shostack

WILEY

IriusRisk

Ex. 1012 - Page 2

Threat Modeling: Designing for Security

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

Copyright © 2014 by Adam Shostack

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

SKY10024132_011421

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) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

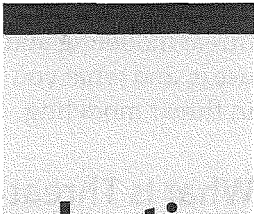
Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2013954095

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.



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

IriusRisk

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 on Risk

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.

3. Address threats using the approaches in Part III.
4. Validate your work for completeness and effectiveness (also Part III).

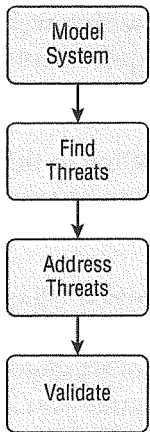


Figure I-1: The Four-Step Framework

This framework was designed to align with software development and operational deployment. It has proven itself as a way to structure threat modeling. It also makes it easier to experiment without replacing the entire framework. From here until you reach Part V, almost everything you encounter is selected because it plugs into this four-step framework.

This book is roughly organized according to the framework:

Part I “Getting Started” is about getting started. The opening part of the book (especially Chapter 1) is designed for those without much security expertise. The later parts of the book build on the security knowledge you’ll gain from this material (or combined with your own experience). You’ll gain an understanding of threat modeling, and a recommended approach for those who are new to the discipline. You’ll also learn various ways to model your software, along with why that’s a better place to start than other options, such as attackers or assets.

Part II “Finding Threats” is about finding threats. It presents a collection of techniques and tools you can use to find threats. It surveys and analyzes the different ways people approach information technology threat modeling, enabling you to examine the pros and cons of the various techniques that people bring to bear. They’re grouped in a way that enables you to either read them from start to finish or jump in at a particular point where you need help.

Part III “Managing and Addressing Threats” is about managing and addressing threats. It includes processing threats and how to manage them, the tactics and technologies you can use to address them, and how to make risk tradeoffs

you might need to make. It also covers validating that your threats are addressed, and tools you can use to help you threat model.

Part IV “Threat Modeling in Technologies and Tricky Areas” is about threat modeling in specific technologies and tricky areas where a great deal of threat modeling and analysis work has already been done. It includes chapters on web and cloud, accounts and identity, and cryptography, as well as a requirements “cookbook” that you can use to jump-start your own security requirements analysis.

Part V “Taking it to the Next Level” is about taking threat modeling to the next level. It targets the experienced threat modeler, security expert, or process designer who is thinking about how to build and customize threat modeling processes for a given organization.

Appendices include information to help you apply what you’ve learned. They include sets of common answers to “what’s your threat model,” and “what are our assets”; as well as threat trees that can help you find threats, lists of attackers and attacker personas; and details about the *Elevation of Privilege* game you’ll use in Chapter 1; and lastly, a set of detailed example threat models. These are followed by a glossary, bibliography, and index.

Website: This book’s website, www.threatmodelingbook.com will contain a PDF of some of the figures in the book, and likely an errata list to mitigate the errors that inevitably threaten to creep in.

What This Book Is Not

Many security books today promise to teach you to hack. Their intent is to teach you what sort of attacks need to be defended against. The idea is that if you have an empirically grounded set of attacks, you can start with that to create your defense. This is *not* one of those books, because despite millions of such books being sold, vulnerable systems are still being built and deployed. Besides, there are solid, carefully considered defenses against many sorts of attacks. It may be useful to know how to execute an attack, but it’s more important to know where each attack might be executed, and how to effectively defend against it. This book will teach you that.

This book is not focused on a particular technology, platform, or API set. Platforms and APIs influence may offer security features you can use, or mitigate some threats for you. The threats and mitigations associated with a platform change from release to release, and this book aims to be a useful reference volume on your shelf for longer than the release of any particular technology.

This book is not a magic pill that will make you a master of threat modeling. It is a resource to help you understand what you need to know. Practice will

help you get better, and deliberative practice with feedback and hard problems will make you a master.

New Lessons on Threat Modeling

Most experienced security professionals have developed an approach to threat modeling that works for them. If you've been threat modeling for years, this book will give you an understanding of other approaches you can apply. This book also give you a structured understanding of a set of methods and how they inter-relate. Lastly, there are some deeper lessons which are worth bringing to your attention, rather than leaving them for you to extract.

There's More Than One Way to Threat Model

If you ask a programmer "What's the right programming language for my new project?" you can expect a lot of clarifying questions. There is no one ideal programming language. There are certainly languages that are better or worse at certain types of tasks. For example, it's easier to write text manipulation code in Perl than assembler. Python is easier to read and maintain than assembler, but when you need to write ultra-fast code for a device driver, C or assembler might be a better choice. In the same way, there are better and worse ways to threat model, which depend greatly on your situation: who will be involved, what skills they have, and so on.

So you can think of threat modeling like programming. Within programming there are languages, paradigms (such as waterfall or agile), and practices (pair programming or frequent deployment). The same is true of threat modeling. Most past writing on threat modeling has presented "the" way to do it. This book will help you see how "there's more than one way to do it" is not just the Perl motto, but also applies to threat modeling.

The Right Way Is the Way That Finds Good Threats

The right way to threat model is the way that empowers a project team to find more good threats against a system than other techniques that could be employed with the resources available. (A "good threat" is a threat that illuminates work that needs to be done.) That's as true for a project team of one as it is for a project team of thousands. That's also true across all levels of resources, such as time, expertise, and tooling. The right techniques empower a team to really find and address threats (and gain assurance that they have done so).

There are lots of people who will tell you that they know the one true way. (That's true in fields far removed from threat modeling.) Avoid a religious war and find a way that works for you.

Threat Modeling Is Like Version Control

Threat modeling is sometimes seen as a specialist skill that only a few people can do well. That perspective holds us back, because threat modeling is more like version control than a specialist skill. This is not intended to denigrate or minimize threat modeling; rather, no professional developer would think of building software of any complexity without a version control system of some form. Threat modeling should aspire to be that fundamental.

You expect every professional software developer to know the basics of a version control system or two, and similarly, many systems administrators will use version control to manage configuration files. Many organizations get by with a simple version control approach, and never need an expert. If you work at a large organization, you might have someone who manages the build tree full time. Threat modeling is similar. With the lessons in this book, it will become reasonable to expect professionals in software and operations to have basic experience threat modeling.

Threat Modeling Is Also Like Playing a Violin

When you learn to play the violin, you don't start with the most beautiful violin music ever written. You learn to play scales, a few easy pieces, and then progress to trickier and trickier music.

Similarly, when you start threat modeling, you need to practice to learn the skills, and it may involve challenges or frustration as you learn. You need to understand threat modeling as a set of skills, which you can apply in a variety of ways, and which take time to develop. You'll get better if you practice. If you expect to compete with an expert overnight, you might be disappointed. Similarly, if you threat model only every few years, you should expect to be rusty, and it will take you time to rebuild the muscles you need.

Technique versus Repertoire

Continuing with the metaphor, the most talented violinist doesn't learn to play a single piece, but they develop a repertoire, a set of knowledge that's relevant to their field.

As you get started threat modeling, you'll need to develop both techniques and a repertoire—a set of threat examples that you can build from to imagine how new systems might be attacked. Attack lists or libraries can act as a partial

substitute for the mental repertoire of known threats an expert knows about. Reading about security issues in similar products can also help you develop a repertoire of threats. Over time, this can feed into how you think about new and different threats. Learning to think about threats is easier with training wheels.

“Think Like an Attacker” Considered Harmful

A great deal of writing on threat modeling exhorts people to “think like an attacker.” For many people, that’s as hard as thinking like a professional chef. Even if you’re a great home cook, a restaurant-managing chef has to wrestle with problems that a home cook does not. For example, how many chickens should you buy to meet the needs of a restaurant with 78 seats, each of which will turn twice an evening? The advice to think like an attacker doesn’t help most people threat model.

Worse, you may end up with implicit or incorrect assumptions about how an attacker will think and what they will choose to do. Such models of an attacker’s mindset may lead you to focus on the wrong threats. You don’t need to focus on the attacker to find threats, but personification may help you find resources to address them.

The Interplay of Attacks, Mitigations, & Requirements

Threat modeling is all about making more secure software. As you use models of software and threats to find potential problems, you’ll discover that some threats are hard or impossible to address, and you’ll adjust requirements to match. This interplay is a rarely discussed key to useful threat modeling.

Sometimes it’s a matter of wanting to defend against administrators, other times it’s a matter of what your customers will bear. In the wake of the 9/11 hijackings, the US government reputedly gave serious consideration to banning laptops from airplanes. (A battery and a mass of explosives reportedly look the same on the x-ray machines.) Business customers, who buy last minute expensive tickets and keep the airlines aloft, threatened to revolt. So the government implemented other measures, whose effectiveness might be judged with some of the tools in this book.

This interplay leads to the conclusion that there are threats that cannot be effectively mitigated. That’s a painful thought for many security professionals. (But as the Man in Black said, “Life is pain, Highness! Anyone who says differently is selling something.”) When you find threats that violate your requirements and cannot be mitigated, it generally makes sense to adjust your requirements. Sometimes it’s possible to either mitigate the threat operationally, or defer a decision to the person using the system.

With that, it’s time to dive in and threat model!