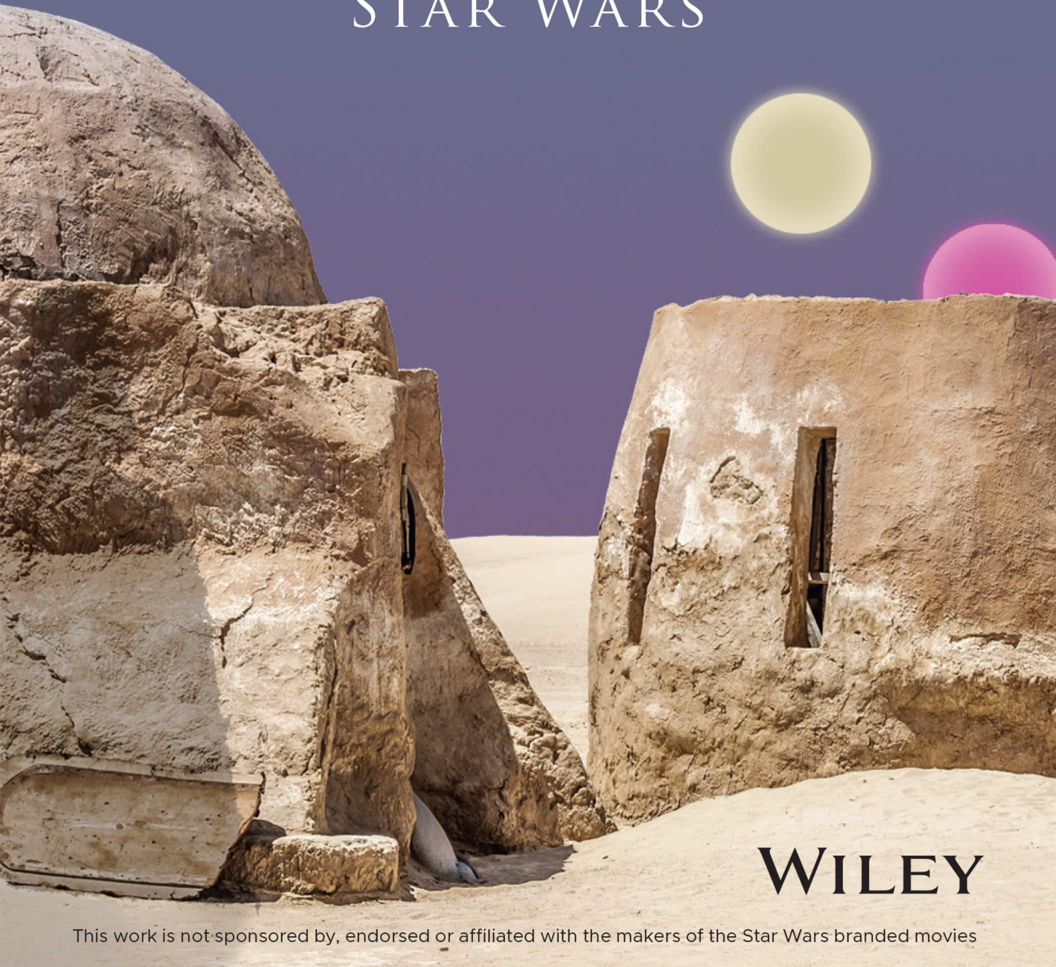
ADAM SHOSTACK

THREATS

What Every Engineer Should Learn From Star Wars



WILEY

THREATS

What Every Engineer Should Learn from Star Wars

Adam Shostack

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada and the United Kingdom.

ISBN: 978-1-119-89516-9
ISBN: 978-1-119-89517-6 (ebk.)
ISBN: 978-1-119-89769-9 (ebk.)

Library of Congress Control Number: 2022947804

Cover image: © Perszing1982/Getty Images
Cover design: Wiley

The Empire doesn't consider a small,
one-man fighter to be any threat
or they'd have a tighter defense.
—General Dodonna

About the Author

Adam Shostack is a leading expert on threat modeling. He's also a consultant, expert witness, and game designer. He has decades of experience delivering secure products and systems. His experience ranges across the business world from founding startups to nearly a decade at Microsoft.

His accomplishments include:

- Helped create the CVE; now an emeritus member of the Advisory Board
- Fixed Autorun for Windows XP and Vista
- Led the design and delivery of the Microsoft SDL Threat Modeling Tool (v3)
- Created the *Elevation of Privilege* threat modeling game and helped create *Control-Alt-Hack*
- Wrote *Threat Modeling: Designing for Security*, and coauthored *The New School of Information Security*

Beyond consulting and training, Shostack serves as an advisor to many companies and academic institutions. He is an Affiliate Professor at the Paul G. Allen School of Computer Science and Engineering at the University of Washington.

You can learn more at `shostack.org`.

Acknowledgments

This book would be different from cover to cover if not for the stunning and deep worldbuilding by George Lucas, and the crew and cast of Star Wars who first brought it to life. That first crew not sweating the details could have robbed us all of so much.

This book has been a five-year mission to clearly explain security to engineers. In 2017, a gentleman in Cupertino asked me the simple question, "Where do I go to learn more about these threats?" I didn't write down his name, but if you see this, thank you for the question, and sorry it took so long to answer.

In my explorations, I've spoken with hundreds of people about the frame of "what every engineer needs to know." I want to gratefully acknowledge all their contributions, and whatever errors remain are mine.

My amazing team of teachers at Shostack + Associates (Valery Berestetsky, Jamie Dicken, and Caroline Emmott) also read the entire draft (sometimes more than once) and helped ensure the many lessons from our students are included and the many questions they ask are answered.

While I worked on this book, I also worked on a set of courses with LinkedIn Learning. My team there, including Alyssa Pratt, Rae Hoyt, and Andrew Probert, were warm, supportive, and coaching throughout. Working with them on a set of STRIDE courses while I worked on this book improved the separate content in each.

Loren Kohnfelder did more than create the STRIDE mnemonic (with Praerit Garg) that forms the foundation of this book. He also read most of this book in early draft form, and I treasured our long conversations about the specifics and also the structure.

Dean Tribble and Jonathan Shapiro reminded me of Mark Miller's work on Authority as a way out of the messy thinking around privileges and permissions. That challenge could have ended the mission. Similarly, the chapter on parsing became much clearer and richer when I read the work of the LangSec community. Jeff Williams provided thoughtful perspective on early drafts. Izar Tarandach gave the text a close read, even though he has a somewhat competing book. Jim Bell explained how spacecraft on Mars track time.

Some of these improvements to early drafts were to text that's evolved so much that the improvement is no longer visible. I am extra grateful to those people because they helped the book in a way that's less visible but no less vital: Michael Roytman suggested date formats for canonicalization pendulums, and those are way easier to parse than my URL examples. Kim Wuyts helped with the definition of predictability and its privacy impacts.

Thanks also to Jon Callas, Chris Eng, Mark French, Tom Galhager, Shawn Hernan, Jeff Jarmoc, Arron Johnson, Christoph Klaaßen, Lucky Munro, Daniel Ostermeier, Ian Poynter, John Poulin, Morgan Roman, Anant Shrivasta, Peleus Uhley, Tarah Wheeler, and Charles Wilson.

Cypherpunk and lawyer Doug Barnes was the friend who wrote "'and then the cops show up' is a rotten step to include in your protocols," quoted in the repudiation chapter. The sentence and paragraph were complex and so I acknowledge him here.

Others have used Star Wars to illustrate: *The Ultimate Star Wars and Philosophy* (Wiley, 2015), *Star Wars Psychology* (Sterling, 2015), and *The World According to Star Wars* (Sunstein, 2016). I learned from each, and my use of Star Wars is more focused and nuanced for having them to guide the way. Kellman Meghu pointed out that until

Darth Vader shows up, the Empire is running a pretty solid breach response executive meeting, and I haven't found a way to work that into the text. And while mentioning Star Wars content, I especially want to thank those readers who are less familiar with it for pointing out where I went too deep. While they have not failed me, they remain safely anonymous.

Last, but not least, I want to thank my team at Wiley: Jim Minatel for asking provocative questions again and again as I figured out why this is more than a chapter of another book and then advising me how to write a book that draws on a beloved universe. Kelly Talbot has been a fantastic project editor, embracing the humor and challenges. Kim Wimpsett edited carefully where it was needed and fixed that which shall not be named. And speaking of not named: there were many people involved who I never got to meet. Much like the anonymous Rebels so frequently in the background, you helped make this great.

—Adam Shostack

Contents

Preface

How does R2-D2 know who Ben Kenobi is? How does he decide to play the recording of Princess Leia for Ben, but not Luke? How does Princess Leia tell R2 her intentions? These three questions touch on fundamental issues of security: authentication, authorization, and usability. (Star Wars geeks have an answer to the first from the prequels, but Leia does not know that answer.) What's more, the way the world of Star Wars engages with technology and computers gives us a familiar base from which to learn about how technology works in our world.

I was a Star Wars fan before I ever wrote a line of code and long before I broke my first system. As I became an expert in computer security, it became clear to me that we in the field are much better at code than with stories, and while it's tempting to say "That is why you fail," telling better stories is not our only hope. As I reflected on Star Wars, I realized that as the crawl fades, the camera descends onto Princess Leia's ship being pursued over…a stolen data tape! I realized Star Wars is not only the story of Luke's hero's journey and growth into adulthood but also a story of information disclosure and its consequences. Over the last decade, I've used Star Wars to tell the story of computer security because the epic stories give us reference points and illustrations of important issues.

In this book, almost every reference is to the original trilogy. There is material I could use in *Rogue One*, in the prequels and sequels, and in the TV shows, the books, and more. But I'll assume that most readers have only watched and rewatched the core three: *Star Wars*, *Empire*, and *Jedi*.

Like the Force is a property of all living things, security is a property of all technological systems. And like the Force has a light side and a dark side, security has defenses and attacks. This book is focused primarily on the attacks, the threats, the problems. You need to understand the threats to select appropriate defenses. It's dramatic to watch the Emperor unleash purple force lightning on Luke Skywalker, but better training could have alerted Luke to the threat and how to defend against it. Neither a firewall nor a checklist will block force lightning.

If you want to make a home secure, you need to think through the many things that might go wrong. Some are natural (floods), some can be natural or manmade (fire), and some (theft) are the acts of intelligent beings.

We have implicit models of what a home is, the types of homes, and the common types of problems. Those problems vary somewhat: the central plain of the United States has tornados, southeast Asia has monsoons, and the Middle East has sandstorms. But you can go to your insurer and get a list. (It's split across "optional coverage" and "exclusions.")

Our implicit models of how technical systems are set up are weaker. Technology has more variety and more rapid change than our homes or office buildings. Three-tier architectures are unlike microservices. Microservice cloud deployments differ from the virtual machines deployed a mere decade ago.

Technological builders and defenders have a disadvantage: it's hard to get away from thinking about making the system work. We all know it's hard to make a system work. That there are trade-offs and compromises made to get the code to work, to get it to customers, and even to deploy it.

Security's traditional response to this has been an exhortation to "Think like an attacker!" That's hard. In response I encourage people to think like a professional chef, planning for hundreds of diners to sit down tonight. Most of us wouldn't know where to start. But we don't have to "think like an attacker" to get different perspectives on the systems we're working on.

Security also differs from other potential problems because we have intelligent attackers who can learn and adapt. If we think about a person wanting to steal your stereo, then they can come through the front door in many ways: they can kick it in, jimmy the latch, pick the lock, or steal a key. Some attacks exploit vulnerabilities: the lock has a weak front plate that's easily drilled because of a defect in the casting. Some exploit design flaws: the lock has only a few pins, making it easy to pick. They can bypass our defense, walking over to a window. And in technological systems, the range of attacks seems endless and perhaps unknowable—a problem this book solves.

One of the reasons that we fail at making secure systems is attackers have a great many advantages. They can study their target, plan their attacks, and launch them only when they feel confident. They can do what they will to take control of a system, make it misbehave, or embarrass its creator. And some of what attackers do is really very clever, and all of it is unexpected. That's tremendously important. In a great little video *The Death Star Architect Speaks Out*, the architect says, "The shot was literally not possible unless you had magic powers. Maybe if someone would have told me to account for space wizards when designing the exhaust ports, we would still have a Death Star!"

He has a good point. Too often, security researchers get publicity for taking a completed system and pointing out flaws…like engineers not knowing a torpedo could fly through miles of piping without turbulence deflecting it. It can feel like security experts are judging you and answering every question by rolling their eyes and saying "Search your feelings." This book focuses on the important threats.

Security expert and author Bruce Schneier once wrote, "When I visited the National Security Agency, I asked to see the 'big book of attacks.' They told me there's no single place where it's all written down." This book aims to fix that. It's important because understanding "the attacks" is easier if there is a defined set of "the attacks." This is not an attempt to categorize every attack or to be comprehensive. That last is probably surprising and may even worry you. But the reality is that security issues are violations of security requirements.

The requirements for different systems are different. Should I include violations of the requirements for nuclear bombs or currency printing? ("Fewer than two people can activate the system" or "Another customer can obtain the same paper stock.") Completeness would obscure the more common attacks and make it hard to quickly reference threats that may inspire and enable you to reason by analogy and discover attacks on your own systems. Understanding the threats is the crux, and until now they've been hard to understand.

Someone else wrote "All interesting systems surprise their creator." That's the property that takes them from useful to interesting. And security issues are often issues of surprise. They rely not only on mistakes in what's there, but in the failure of architects to develop defenses.

Human attention is a harsh master. It is hard to perceive what is missing. My intent in cataloging common issues is to say: these matter. These must be considered and, by collecting, organizing, and presenting them, provide some clarity about what is in the set of things "you must consider." If what's in this book is ignored, maybe it's reasonable to claim that is a failure of the engineer. That's not to say "You can ignore anything else." Just as a pilot must land the plane beyond the checklist or a surgeon must treat the patient, what must be addressed is not limited to what's in the pages of this book.

Human attention is really a harsh master. Daniel Kahneman is a Nobel Prize–winning founder of behavioral economics. In his lovely book, *Thinking Fast and Slow*, he uses only a single acronym: WYSIATI. What You See Is All There Is. The importance of what's in front of you is so great, it crowds out our efforts to "remain aware" and to "keep in mind." Yet as an engineer you must do exactly that—keep in mind reliability, performance, usability, maintainability, and a great many other properties. We have many tools for managing such things, including automation, checklists, and the judgment of diverse teams.

For this book, I am making a design trade-off and assuming that defenses are known and understood, or at least understandable once you understand the threats. So I focus on the threats and touch briefly on the defenses. That's a conscious trade-off to make the book shorter and more approachable.

Introduction

My students teach me so much. As I hear the questions they ask and read the assignments they submit, I learn where they face challenges in securing their systems. I learned about threats over a decades-long career, from a few wise teachers and from many mistakes. As I mention in the acknowledgments, this book really was catalyzed by a simple question: "Where do I go to learn about the threats?"

A bit like "There's good in him, I've felt it," I've felt that question in so many conversations. The word *security* subsumes a great deal of complexity and nuance. I was going to say we tend to learn about threats by osmosis, but that's not true. We tend to learn about threats when something blows up. Even when that something is smaller than a Death Star, the lessons are often traumatic, sometimes career-changing. Tragedy is a bad teacher.

If we want to be systematic in our search for threats to our products, we must be structured in how we learn and teach about those threats.

Who This Book Is For

This book is for every engineer.

It will be most useful to those who build or operate complex software-rich systems. There are hard trade-offs in engineering, which are made harder when security goals are obscure or vague. The book is focused on systems that incorporate code, but these days, what doesn't? Engineers who work in more traditional parts of the field (aerospace, chemical, civil, mechanical) are finding that these more elegant systems from a more mechanical time are being supplanted. Your systems must now interface with code, and you must address its security properties.

Over the last few decades, the job of software development and systems operation has changed. We've learned that our hopes of retrofitting properties from accessibility to reliability to usability have cost us dearly and that we need to incorporate each from the start. We are learning that security is much the same way. Choices made during system development have consequences. We see the need to address security earlier and more holistically.

This book is also for security professionals and enthusiasts. There are many pathways into many fields focused on security and hacking. Few of them provide a broad framework that will serve to organize the flood of information about threats, vulnerabilities, and exploits that you'll encounter. My hope is that this book serves all of them.

This book is for every engineer, even if they're not a science-fiction fan, and if you are, whatever world you love. As I spoke about this book, Star Trek fans came out of the nebulas to ask "Why?" And I love Trek. I love the optimistic view of the future, how the series reveres competence and science, and the writing and character development. I turned in the manuscript with the dedication: "to boldly secure what no one has secured before," as an attempt at a loving homage. My team told me that it was too jarring for the opening, and they were right.

What You'll Gain from This Book

Security.

More specifically, you'll gain the understanding of security in ways that enables you to build and operate systems that perform despite the efforts of adversaries. Much like understanding force (the mass times acceleration kind) allows us to think about many different parts of the world and bring it to bear on our projects, this book provides you with an enduring framework to anticipate threats.

It’s traditional to include a breathless list of security flaws here, in the hopes of motivating readers. It hasn’t seemed to work, so I’m not going to bother. In 2023, the issue with security is no longer why. It’s what and how.

A Few Words for the Nonengineer

This book is written for engineers: people who build or operate complex technical systems, especially the algorithms, chips, sensors, and actuator parts of those systems. It’s written to be as clear as reasonably possible, and if you’re a nontechnologist looking for advice, I want to include the three things you should do.

First, turn on automatic update on everything, most especially devices, operating systems, and web browsers. The updates that engineers ship often address security problems that can be exploited automatically. If your vendor mixes functionality changes with security fixes, complain loudly. But this step is a crucial defense against those exploitable problems.

Second, use a password manager, and have it create long, random passwords for you. One of the ways security fails is when websites leak your email address and password. Attackers gather and trade those lists, and they test the combinations on every website they can. They also test variants. They know that my Amazon password might be “adamamazon” or “amazonas1?” and computers are very good at testing those sorts of combinations, along with amazon-feb and the others you’ve thought of. Use a long, random password. If I expect I’ll need to type it in regularly, I’ll use the feature that gives me three or four random words as a password. By the way, I use 1Password from Agilebits as my password manager and recommend it. (We have no business relationship.)

Third, trust your feelings. If you feel a website isn’t safe, leave. Find the company by searching or with a bookmark. If you think an email is suspicious, call the person or entity that claimed to have sent it. Use the number on the back of your card to call your bank. In each of these cases, you're taking control, and you're using resources that an attacker can't influence.

Maybe an attacker can replace the card in your wallet, and if you have attackers like that, seek professional help. I'm not saying that sarcastically. If you're up against a spy agency who will spend the time and energy to create a card and put it in your wallet, this advice isn't going to save you.

Two more optional steps if you want extra safety. First, craft special email addresses for special relationships. Set up something like hiufdsuapre8wafdsjkf@gmail.com and use it for either one bank or all your banks. This protects you if an attacker takes over your main email account, and it helps you sort out phishing emails. If you only use that for your bank, then any mail from "your bank" in your main account is automatically suspicious. See above about trusting your feelings.

Lastly, I use a different browser and browser profile for online banking. Browser software is pretty solid these days, but with all the attacks, I feel more comfortable having a low-use browser for that. (These days, one is Firefox, the other is Chrome. At other times, it was two different Firefox profiles, with two dramatically different visual themes.)

That's it. That's my advice. Thank you for buying this book. You're welcome to read it or pass it on to a technologist or budding technologist who you know. Either way, I'm going to assume a technical reader, so we start speaking the binary languages that underpin both a galaxy far, far away and our own world.

Let me draw your attention to a principle that underlies the advice: isolation. A password manager isolates sites from each other, as does using two email accounts or two browsers. Leaving a site or calling your bank leaves the locus of an attack. That isolation, separating parts of a system from each other, is also the reason we have different computer accounts, firewalls, and a host of other defensive techniques.

Of course, each layer of isolation comes at a cost of convenience. Not allowing software to seamlessly work together means you have to do the things that make them work together, because that way attackers have to trick you into doing those things.

This advice is sadly not the advice you'll get from everyone. We lack information on the root causes and history of incidents that would help us prioritize, which is a problem I write about elsewhere and don't dwell on much in this book.[1]

Security Terminology

This book is about *threats*. We all know a threat when we hear one—"Give me your money, or else!" "I have altered the terms of the deal. Pray I do not alter them…any further." I use threat to mean a future problem and one that can often be averted if we take preventative action.

Security folks use the word *threat* in a variety of ways. We call an attacker a threat, or sometimes a threat agent. The anti-malware part of the industry calls each virus or bit of malware a threat.

Carrying out a threat is an attack. Each of the threat, its manifestation, and its impact can be a concern. The law considers a credible threat as assault; the act of hitting someone is the battery in "assault and battery." These can result in injury. In cybersecurity, we often worry about both the threat and its result. If someone breaks in by spoofing a legitimate user, they can quickly chain other threats, such as tampering or information disclosure. Especially as you are learning, being specific about the relationship between mechanism and impact can be helpful. A *risk* is the quantified refinement of a threat, and those quantifications often involve probability of success and the magnitude of the impact in dollars or lives.

[1]You can find more on that at shostack.org/resources/lessons.

An attacker uses an *exploit* to take advantage of a vulnerability. An exploit (as a noun) is a bit of software that allows its user to do something that the system owner would like to prevent. To exploit (as a verb) is to use that software against a target. A *vulnerability* is either a specific code issue (a bug), a flaw where design requirements have been overlooked, or the result of a trade-off made by designers or operators. Sometimes specificity helps with clarity; other times it descends into pedantry.

The word *trust* is used a lot in computer security and can be trusted to trip up the unwary. In normal English, trust means "a firm belief in someone's reliability, honesty, or ability." Trustworthy means someone who lives up to that trust. In computer security, trusted means something with the ability to break your security. Cambridge University professor Ross Anderson provides an example: "The spy caught selling secrets was trusted but not trustworthy." Others have pointed out that the word is often used in a passive or Orwellian voice. A "trusted system" fails to specify who trusts it. The Galactic Empire often labeled systems as "trusted" to bypass any discussion of their impact on the people it touches.

Aphorisms

There are a few bits of pithy wisdom I'd like to share because they can broadly inform your work as it touches on security.

"Attacks only get better; they never get worse." Bruce Schneier attributes this as a saying at the American National Security Agency (NSA). While defenses do get deployed, the lesson of an attack is never lost. The tools developed to execute it don't go away. They're honed and refined.

"Theories of security come from theories of insecurity," said Rick Proto of the NSA. Those attacks get better, and the collection of attacks inform how we think about what security is.

"All models are wrong; some models are useful." British statistician George Box said this.

"Computer security is perverse. When you want something to be difficult, it's easy, and when you want it to be easy, it's hard." (Me.) Consider file deletion. When you want to make a file really disappear, it's difficult, and when you want to recover it, it is surprisingly hard. It's hard to really make a file disappear because deletion usually just removes the pointers within the file system. If you try to overwrite the bits on a magnetic disk, it turns out that the physical records on the disk vary in size and so can be read after overwriting. And flash drives make it tough to ever write to the same locations. Similarly, randomness is easy to find when you want predictability. Computers seem unpredictable and heisenbugs are common, but just try writing a safe random number generator.

"Attackers will spend their budget how they want, not how you hope." (Me again.) You may hope that attackers will behave in very specific ways, but then they wouldn't be attackers.

"Security is a systems property." It's unclear who first said this. This is a true claim, and what it means is that system security is often limited by weak links. This book helps you remove the obviously weak links.

"Shipping is a feature." This is a common saying at Microsoft. All the new features that have been built do no good until they're being used by your customers, so delaying to add a few more is often unwise. Similarly, delaying delivery in the hopes of achieving perfect security means no one can use your new features. I'm making that same call in shipping this book now: I hope its virtues outweigh its flaws.

"The devil is in the details." Whoever said this wasn't thinking of security, but they could well have been. A great many things turn out to be less secure as one delves in, and security experts have great respect for talented reverse engineers who pry systems apart to understand their inner workings and in doing so discover unexpected properties of the system.

How This Book Is Organized

This book starts with STRIDE, a classic way of thinking about threats. STRIDE stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Expansion of Authority. STRIDE is a mnemonic that helps us remember six major groups of threats, covered in the first six chapters. Those are followed by chapters on predictability, parsing, and kill chains.

Most chapters in this book follow the same general plan: start with an explanation of the threat, then how it manifests in specific technologies, the mechanisms that attackers use, and finally a short section on defenses.

There are many organizational choices to make writing a book like this. I grappled with the different ways computing now works and the way various threats impact them. Those ways include the Internet of Things, mobile, the cloud, and AI/ML. The specifics in these sections are in addition to the broader points made in the chapter, not a replacement for them—the fact that a computer has the shape of an internet of things teddy bear doesn't mean the rest of the chapter doesn't apply. A few of these sections in other chapters have additional sections because the nature of the threat has interesting properties in a specific scenario that's worth discussing.

The sole emergent technology not treated in this way is quantum computing. Most of the STRIDE threats will work on the systems that surround a quantum core, and probably work on that core. For example, the power draw of the mirrors in quantum cryptography leads to important information disclosure attacks. (Quantum crypto uses spin information to distribute cryptographic key information in ways that are hard to eavesdrop on, often relying on fiber between sites. It is very different from the use of quantum mechanics for computing.) The primary early impact of quantum computing seems to be breaking most classical asymmetric cryptography by discovering the keys, an information disclosure threat. If you're curious about quantum, *Law and Policy for the Quantum Age* (Hoofnagle and Garfinkel, 2021) is an excellent primer.

Another crucial organizational choice is to revisit threats. I've learned from teaching the first time someone encounters some information, it may not sink in. Coming back to it from a different angle often helps.

Style and Conventions

Many organizations and products are named. Product names are used to make examples concrete, and no malice is meant toward the creator or trademark owner. The passe convention of including a "for example" with each one wastes the time of most readers for the possible benefit of a few particularly literal-minded ones, who might be confused however many clarifiers are included.

A Few Words from a Jedi Master

Yoda:…a Jedi's strength flows from the Force. But beware of the Dark Side. Anger, fear, aggression; the Dark Side of the Force are they. Easily they flow, quick to join you in a fight. If once you start down the dark path, forever will it dominate your destiny, consume you it will, as it did Obi-Wan's apprentice.

Luke: Vader…Is the Dark Side stronger?

Yoda: No, no, no. Quicker, easier, more seductive.

Luke: But how am I to know the good side from the bad?

Yoda: You will know…when you are calm, at peace, passive. A Jedi uses the Force for knowledge and defense, *never* for attack.

The dark path is the path of ignoring security. Easily, the code flows. But once you start down that path, forever will it dominate your destiny. The easy choice is to ignore security and focus entirely on features that are more visible to customers. Modern languages make complete static analysis feasible by constraining some of the seductive power of pointers. There's a cost: the dark side of C is faster code, but forever will it dominate your security advisories. And 20 years ago, when security mattered less, that was a choice many companies made, often thoughtlessly. It was the choice that Microsoft made in its heyday.

But Yoda was right: "Consume you it will." I worked at Microsoft for most of a decade, and I have tremendous respect for my colleagues who have been bolting security onto Office and Windows and replacing parts of their guts. They have achieved far more than I would have thought possible. But the very different innards of IoS and ChromeOS allow those competitors to move faster today.

Lastly, there is a security career path open to you, a path of attack. It's flashy. It's powerful: "Let me show you how I can pwn your system." And if you want to follow that path, my only request is that you do so ethically, using your skills and knowledge to conduct authorized attacks to build stronger defenses. My own path started with vulnerability discovery but lately has been focused on delivering stronger systems. It's a harder path, but the impact long term can be much greater.

1 Spoofing and Authenticity

Shortly after we first meet Luke Skywalker, he is cleaning his newly acquired droids, and R2-D2 teases him with part of a message that is only supposed to play for Obi-Wan Kenobi. How does R2-D2 know who Obi-Wan Kenobi is? How does he decide to play the recording of Princess Leia for Obi-Wan, but not Luke? As I mention in the book's introduction, these questions are multifaceted. Let's go deeper into questions of names and authenticity.

As we look at this interaction, I'll treat droids as computers. And so we can ask questions like "How does a computer identify a human?" This is one of several crucial types of authentication. We can also ask how a human identifies a computer, or one computer identifies another. Star Wars is full of problems that stem from challenges with how humans identify other humans. In the prequels, why don't the members of the Jedi Council realize that the Chancellor is also the Sith Lord Darth Sidious?

Authentic means something is "the genuine article" or "the real thing." R2-D2 only wants to play the video for the real, authentic Obi-Wan, not anyone who walks up and asks for it. To do that, we need identifiers and authenticators. *Spoofing threats* are violations of authenticity; you get someone or something that is not what you're expecting. The Death Star fails to authenticate R2-D2 when he plugs in, a common flaw in the world of Star Wars. In our world, spoofed authentication codes are a common problem: we call them stolen passwords. But it's not just fake people; it's also fake websites for phishing and other scams.

Identifiers and Authentication

Authenticity first requires an identifier: a statement of who you are. This might be a name (Han Solo) or a role (Stormtrooper). Either might be true or false, and given the risk of impersonation, confusion, or lies, we look for authentication factors, such as an ID, a password, or a uniform, as we evaluate if the identifier is authentic and grant (or deny) authorization. This chapter will start with identifiers both human and technical. We'll naturally touch on authentication as we go through the various specific ways that human and technical identities are spoofed and then learn about it more in depth later in the chapter. There are many forms of authentication to consider, depending on if the authentication is *by* a person or a computer and *to* a person or a computer. From there, we'll look at spoofing in different scenarios, the mechanisms used, and the defenses. This chapter is longer than many that follow because spoofing manifests very differently when a person or a computer is impersonated, and the ways of checking are different when performed by people or computers.

As shown in Figure 1.1, the means of authentication differ, based on what sort of entity is trying to prove its identity and what sort of entity is checking.

Frankly, some of the methods shown in Figure 1.1 are not very reliable. For example, the computer in front of you is authenticated by its physical location: you trust it with your password because you're typing on it. Sometimes that weak authentication is OK, and other times the party checking the other entity wants the authentication to be stronger. See Figure 1.2.

		Authenticating to	
		Human	Computer
Authentication by	Human	Recognition ID badge	Password
	Computer	Physical location Brand	Certificate

FIGURE 1.1 Ways of authenticating

		Authenticating to	
		Human	Computer
Authentication by	Human	Person we know: easy Organization: hard	Hard Remote: very hard
	Computer	Physically present: medium Remote: very hard	Certificates: easy

FIGURE 1.2 Difficulty of authenticating

Technical Identifiers

There are many types of technical identifiers including identifiers for services, machines, files, processes, and users. Some are designed for humans, such as `threatsbook.com`, others are designed for computers, such as 172.18.19.20. Of course, tools exist to map between them. These matter because each mapping is a looming opportunity for errors to creep in or for threats to impact your system.

In fact, any time there's a mapping from a real object to a representation of that object, there can be confusion. Calls like `listen(socket)` and `open(file)` are fraught with threats as you map from a filename to a file descriptor.

Machine or service identity involves a name, such as `rebelbase.threatsbook.com`. Computer namespaces are usually unique for some scope. Ideally, `rebelbase.threatsbook.com` will be globally unique, but there can be many computers with a DNS name of `rebelbase.local`—one on Yavin, one on Hoth, one on your local network.

The service might be being spoofed with a lookalike name, `rebe1base`. If it's a website and you give it a username and password, its operators may log in to the real `rebelbase`, thus spoofing you. Almost all connections that are vulnerable to spoofing have vulnerabilities in both directions, with the impact of an attack falling on different parties.

Computers will often have DNS names like `rebelbase.threatsbook.com`. That address can refer to more than one physical machine, but these are not the only names a physical computer can have. It can have a UNC (Windows) name and other names. The name may refer to more than one machine, via, say, DNS round-robin, and there may be layers of mapping with cnames and other systems for indirection.

Similarly, files have both names and technical identifiers, such as inode numbers, which the filesystem uses for efficiency.

The identity of a process can often be represented by a file or port, or even an executable name. For example, someone might expect that the thing listening on port 25 is a mail server, or the first process named Chrome is our web browser. Processes can change their names in the process table, and malicious code will often try to change process names to masquerade as something harmless.

Lastly, users have various sorts of usernames, display names, and other identifiers, and understanding those brings in enough complexity that it's worth considering the very broad range of human identifiers and then how computers represent them.