Dr. Dobbs Jolt Award Finalist 2014

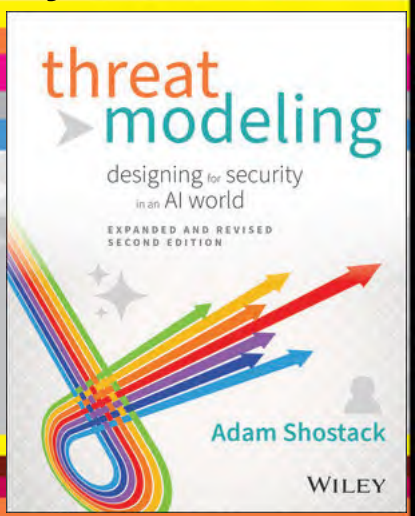
Adam Shostack
Microsoft's Threat Modeling Expert

Chapter 19 "Architecting for Success" from

threat modeling

designing for security

New Edition Coming January 2027



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

CHAPTER

19

Architecting for Success

There is no perfect or true way to threat model; but that is not to say that there are no poor approaches documented, approaches that have never worked for anyone but their author, and it is not to say that you can't compare approaches and decide that some are better or worse. One readily observable indicator is whether the authors describe organizational factors in depth, such as the degree of expertise needed, or inputs and outputs. Another indicator is whether the system has proponents (other than its creators) who make use of it in their own work.

This chapter closes the book by looking at the ways in which the threat modeling practitioner's approach, framing, scope, and related issues can help you design new processes or roll processes out successfully. In other words, it moves from focusing on how threat modeling can go wrong to how to make it work effectively.

This chapter begins with a discussion of flow and the importance of knowing the participants, and then covers boundary objects and how "the best is the enemy of the good." It closes with a discussion of how "the threat model" is evolving and artistry in threat modeling.

Understanding Flow

Flow is the state of full immersion and participation in an activity. It reflects a state of undistracted concentration on a task at hand, and is associated with

effective performance by experts in many fields. In his book *Finding Flow*, Mihaly Csíkszentmihályi (Basic Books, 1997) describes how "the person is fully immersed in what he or she is doing, characterized by a feeling of energized focus, full involvement, and success." Many structured approaches to threat modeling actively inhibit flow in both beginners and experts, and few allow it to emerge. The documented and common elements of flow include the following:

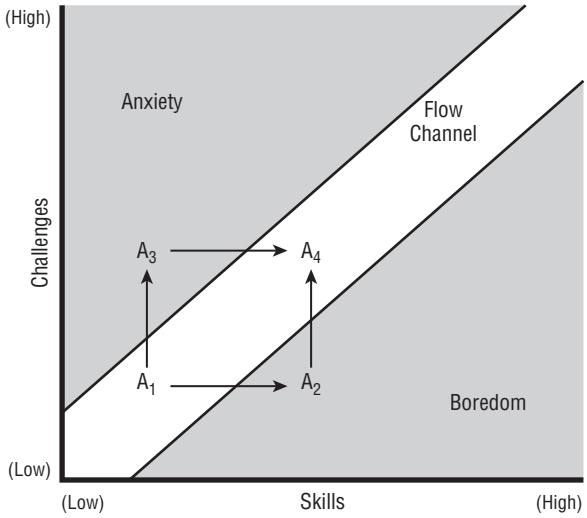
1. The activity is intrinsically rewarding
2. People become absorbed in the activity*
3. A loss of the feeling of self-consciousness*
4. Distorted sense of time
5. A sense of personal control over the situation or activity*
6. Clear goals*
7. Concentrating and focusing
8. Direct and immediate feedback*
9. Balance between ability level and challenge*

Items with an asterisk (*) are those for which I have personally witnessed regular or systematic failures of threat modeling systems to achieve this property.

Flow is the most important test of an approach, methodology, or task for threat modeling. Knowing who will find flow in an approach is a key to architecting for success. If your audience can't find flow, their ability to find threats will be dramatically inhibited. Without flow, threat modeling is a chore, and it is less likely to be a part of an engineering process.

This is not to argue that flow is always a criteria for assessing security processes, especially those beyond threat modeling, but it offers a model for addressing a class of issue. As an example, many approaches to threat modeling have no clearly stated and achievable goal. For example, a goal might be to "find all possible security problems." This is an exceptionally broad goal and one whose achievement is subject to extended argument. Similarly, many processes require diagrams but offer no criteria for what constitutes "sufficient" diagramming. These sorts of problems can be predicted or addressed using flow as a model of people.

One element of flow is the balance of ability level and challenge, which is sometimes represented as a flow channel (see Figure 19-1). The idea is that a person starts an unfamiliar task in state A1. From there, if the challenge is too low relative to their skills, they move to boredom (A2). If the challenge is too high relative to their skills, they move to anxiety (A3). When challenge and skills are balanced, they can learn new skills, take on greater challenges, and experience flow (A4).



From *Flow: The Psychology of Optimal Experience*
by Mihaly Csikszentmihalyi

Figure 19-1: The flow channel

If you are developing a system and find that your participants are either bored or anxious, look for ways to better help them achieve a flow state.

Flow and Threat Modeling

For far too many people, the attempt to threat model leaves them anxious or scared. The balance is skewed toward challenge, and people don't see a way to develop their skills. They're overwhelmed by the details and the requirements. Advocates of threat modeling can unintentionally push people toward anxiousness by overwhelming them with details and possibilities. When someone has no basis for comparison or decision making, they're easily overwhelmed with "you might do it like this or that," or with decision fatigue. (The reason it's so hard to choose between 80 varieties of toothpaste is because most people don't have any idea what makes one toothpaste better than another, and the differences may not matter a lot.)

Following are three key aspects of aligning threat modeling with finding flow:

- **Clear goals:** Many threat modeling processes are focused on a report, or don't have clear internal steps. To the extent that it's possible, each system in this book has clear goals, exit criteria, and self-checks.
- **Direct and immediate feedback:** There are a variety of levels at which feedback is important. Is this a good threat? Have we examined this system "sufficiently"? Microsoft's SDL TM Tool provides instant feedback

on diagrams, and you can expect other tools to provide better feedback on other elements of threat models over time (or, if they don't, encourage their creators to consider it). The tool is disucssed further in Chapter 11, "Threat Modeling Tools."

- **Balance between ability and challenge:** Some of the systems in this book are for experts. Others (especially *Elevation of Privilege*) are for beginners. Selecting a system that's appropriate for the skills you have will help you develop new ones. Jumping to something that's too hard will be frustrating, and too easy to quit.

The ideal state for anyone, in any task they care about, is the flow channel, where ability and challenge are balanced. The structures in *Elevation of Privilege* are useful because they help people get to that balance in a nonthreatening way—but the challenges don't stop there. If you're a threat modeling expert, you can often find flow by looking for a personal challenge in a threat modeling session. The most elegant threat? The most impactful? The threat that will crystalize a requirement or non-requirement? The threat that will highlight the technical debt impact of a design? Perhaps it's not the best threat, but the most threats, or the most ways in which a certain STRIDE type can appear. Maybe it's a beginner who needs effective coaching?

This idea of finding something to improve relates to a body of work on *deliberate practice*. Deliberate practice is a term of art meaning an assigned practice task with a defined focus area (Ericsson, 1993). A good music teacher might assign two new students vastly different pieces to play, with one focusing on rhythm, the other focusing on dexterity. Each piece would be selected against a weakness that the teacher can see. (A popular author contorted this into "10,000 hours of work will make you an expert.") There's good evidence that expertise develops faster with deliberate practice, and threat modeling will improve when we start to develop example systems that help people work through common failings.

In the SDL TM Tool, people model software at the start of threat modeling. They do so in whiteboard-like diagrams to minimize cognitive load (see the section "Beware of Cognitive Load" later this chapter.) They happen at the beginning to enable a positive feedback experience ("You've successfully created a diagram!"). Contrast this with other approaches that begin with something like "Create a list of all the assets in the system." Making a list of assets is not something most developers do, so starting a threat modeling exercise from a diagram can be seen as more likely to lead to a flow experience, and less likely to inhibit one.

The most trenchant critique of the SDL Threat Modeling Tool (which I created) is that it can be tedious. The STRIDE-per-element approach used in the tool can be seen as problematic in that similar issues tend to crop up repeatedly in a threat model. Therefore, as people use the tool, they find themselves repeatedly entering the same threat, copying and pasting, or entering a reference. It's not a great use of people's precious time. I take some small pride in having created

a tool that helps people who previously couldn't perform these tasks reach the point where they can execute the key elements of the tasks and even get bored with them.

But my pride is not what's important. What's important is a tool that challenges people more appropriately, enabling them to do better at threat modeling. Someone once said that a video game is a program with a user interface that's so compelling that you keep using it for fun. A compelling video game offers tools for you to master, trickier environments for you to explore, and bigger monsters to fight so that the challenge presented keeps increasing. Threat modeling tools could do the same.

Stymieing People

One of the opposites of flow is when people feel stymied by what's in front of them. Sometimes that's because the problem at hand is hard. Often, it's because the person doesn't know where to start, because the problem is too big, or they don't know what success looks like. When no one knows what success looks like, one way to overcome that is to look for ways to break the problem into smaller chunks, or look for other ways to get started. (The trope that "you need to crawl before you can walk" often comes up in these circumstances.)

A variant of this is the claim that threat modeling doesn't find anything interesting (see, for example, Osterman, 2007). Sometimes, threat modeling can involve walking through a lot of possibilities but finding few interesting threats. This doesn't feel good; but if you have skilled practitioners performing the threat modeling and still not finding anything, you can rest easier. You have assurance that the designers probably did a good job of threat modeling, either explicitly or implicitly.

Beware of Cognitive Load

Cognitive load is the amount of information you're asking someone to keep in their working memory at one time. When there's too much, people are forced to fall back to cheat sheets, work much more slowly, or go back to their e-mail. Cognitive load can be another inhibitor of flow.

Whiteboard diagrams are familiar to almost everyone in software. Data flow diagrams and swim lane diagrams are almost as familiar. Their cognitive load is small. UML and other diagrams are both more complex and less familiar, so for many, the load is greater. Therefore, you should generally prefer data flow and swim lane diagrams, so that the energy can be spent on the unique security tasks in threat modeling. If everyone on a team is deeply familiar with UML, then it may be a better choice, as UML is more expressive than DFDs. See the section "Boundary Objects," later in this chapter.

Formal systems can offer an interesting example of cognitive loading. When people encounter a new notational system for the first time, or the first time in a long time, they have to think about what something like ∩ means (hint, it's set union or intersection), or how to pronounce φ ("phi"). The energy spent on such subtasks distracts from the nominal goal. Of course, there are good reasons to use such notations, including precision. There is no universal "good decision" regarding something like notation. There are simply decisions that provide various participants with different value in different situations. One such value would be that the expected participants can discover and address threats. You should focus on that.

Avoid Creator Blindness

Another important aspect to understanding people in a threat modeling session is what I call *creator blindness*. This is the set of cognitive factors that make it difficult to proofread your own work, or judge an artistic work in progress. When the technology you're creating is a program of any complexity, making it work properly, meeting all the stated and unstated requirements, will expand to take up all available room. The cognitive load involved in thinking about the technology and how to attack it is enough to overwhelm nearly anyone. This is part of why Eric Raymond claims that "All bugs are shallow with enough eyes" (Raymond, 2001). Other people will see your bugs faster than you will. Security professionals will see security bugs faster than others, but trying to threat model your own creation is a tough challenge. If you need to analyze your own technology, use the structures in Part II of this book to help draw you into looking at it in a new light.

The problem of being unable to assess your own system also shows up in the design of threat modeling approaches. The person who designs a new threat modeling approach is emotionally involved with the pride of creation, which hinders their ability to assess their own work. The use of structured experiments, as described at the end of the previous chapter, can help with this variant of creator blindness, as can the tools in this chapter.

Assets and Attackers

This book has been critical of asset-centered modeling, and it has been critical to attacker-centric modeling. Not to beat a dead horse, this section examines them again in light of factors such as flow and boundary objects. Asset-centric approaches require developers to focus on something other than the software they're building. That takes them into an unfamiliar space. The jargon "everything in the diagram is an asset" leads to extra cognitive load from using a different word (asset) than is natural (such as computer).

Attacker-centric approaches also stymie flow. The initial requirement to create a set of attackers means putting energy into a subdeliverable, although Appendix C, "Attacker Lists" can reduce or replace that work. Flow is also stymied by the demand to "think like an attacker" or even "this attacker." As discussed in the previous chapter, asking someone to "think like an attacker" can be a tall order (like "think like a professional chef"), and many people will be stymied and have no idea where to start.

Knowing the Participants

The better you understand the people who will be doing threat modeling, the better you can select or design a system that will work for them, and teach them how to perform. Are you designing a system for experts or newcomers? Those new to threat modeling will appreciate structure, while experts will find that it chafes. Are you teaching people who work on a particular technology? If your audience is the Microsoft Windows product group, you might find that a set of unix examples fails to resonate or communicate the details that matter to you.

You can divide participants into two major groups: those who want to develop deep expertise in threat modeling and those who don't. Both are reasonable desires. Many people are far more skilled in database design than I am; but I know enough to know what I don't know, and enough that I can sit in a design meeting and not feel like a waste of a chair. I used to refer to people who are not expert in threat modeling as "non-experts" until I realized how deeply situated in my worldview that is, and how dismissive it can sound. It's important to respect people's desires regarding the skills they want to develop (although it is reasonable to respectfully try to influence those choices and desires). I sometimes fall into the term non-expert, but I prefer to use "expert in other things."

Those who don't want to become threat modeling experts can still be asked to develop a basic degree of familiarity. As discussed in Chapter 17, "Bringing Threat Modeling to Your Organization," much like version control, you can reasonably expect a skilled software engineer to have some familiarity with concepts such as checking code into a branch, and managing branches. Similarly, you can look forward to a world in which every reasonably skilled software engineer will have some familiarity with threat modeling and why it's worthwhile, and in which a shop that doesn't threat model is viewed with as much disdain as one that does version control by copying source onto a random USB drive once a week and throwing them in a drawer.

Threat modeling experts will have to engage differently in such a world, and it will be a better world for many people. A typical engagement will have much deeper questions, and that puts a different burden on the threat modeling experts. Less frequently will a threat modeling expert be able to enter a room and

have something useful to say after a simple glance at a diagram. The "obvious" threats will be found and addressed more frequently by experts in other things.

Security experts will get value from experts in other things when those folks are not leaning on the security experts all the time. They can review instead of create, which frees up time to focus on the trickier, more challenging, deeper threats. At the same time, it can be scary to have experts in other things learn some of the magic tricks that you perform. It threatens the way security professionals self-identify and how they present themselves. Some security professionals will be scared of the change, and not want to move it ahead. Those who do drive change to their organization will find themselves increasingly valued by management and future employers.

These comments on participants are quite general. Understanding your own participants and especially the technical leaders will be a key part of making a process that works. Not understanding the participants is a sure route to failure. Understanding your participants and explicitly documenting the salient attributes of your environment are important components of a successful threat modeling process, both within your organization and when others try to learn from it.

Boundary Objects

The concept of a *boundary object* has been quite useful to me in designing threat modeling approaches that work for a wide set of participants. It is perhaps most easily understood through an example: A set of experts were working on a museum exhibit about birds. Some experts focused on the learning objectives and how the exhibit as a whole would come together. Other experts focused on the individual displays. The bird watchers wanted to focus on comparing the birds, such as markings on the chest, the different beaks. The evolutionary biologists wanted to show the birds in their niches, and how they could have evolved into them. The exhibit wasn't coming together. Each group had its own jargon, its own perspective, its own way of thinking about what made a given bird interesting or not interesting. Long story short, what helped was when the participants had the bird under discussion in the room with them, and could point to salient features. The bird acted as a boundary object, something they could all focus on in their discussions (Star, 1989). Ironically and unfortunately, there is no easy introduction to this field. That is, it lacks a boundary object.

In crafting a threat modeling approach that will include a variety of participants, boundary objects can help. The Microsoft SDL TM Tool includes two by-design boundary objects: diagrams and bugs. At the beginning of this chapter, you learned how creating those diagrams at the start of the process reduces friction to a flow state, and how the choice of diagram type interacts with cognitive load. There's another important aspect to the diagram, which is

that it acts as a boundary object. Both software developers and security geeks can point at the diagram and use it as a focal point for conversation. Their individual jargon, perspectives, and ways of thinking meet when someone goes to the whiteboard, jabs at a line, and says, for example, "Right here, what data goes from foo to bar?" That data flow is the boundary object.

The second boundary object in the tool is the bug. The bug acts as an object that both developers and threat modeling experts can point to when discussing an issue. This is why the SDL TM Tool includes buttons that file bugs with a single click, and threats are portrayed as less complete if a bug hasn't been filed. Using boundary objects at the beginning and end of a process is a good design pattern for any system passed between people with different skill sets. There may be other boundary objects, and looking for them as you work with different communities is a great practice in architecting for success.

The Best Is the Enemy of the Good

There are a number of scenarios in which aspirations of security perfection lead to either not shipping or shipping without security. Aspiring to the best security is admirable, as is delivering good security. Finding the right balance between the good and the best is tricky in many areas of life. Getting the wrong balance can have all sorts of downsides.

There are some threat-modeling-specific risks to be aware of. One is that effort focused on defending against a more powerful adversary might distract from effort to defend against attacks by other adversaries. For example, effort to make it harder to exploit vulnerabilities by corrupting memory and exploit vulnerabilities is deeply technically challenging. It may be that work to prevent the more common social engineering attacks is set aside to focus on the interesting technical challenge. Paul Syverson has shown that in comparing privacy technologies of low- and high-latency mixes, some simple attacks work better on the higher security, high-latency mixes. In each of these cases, the trade-offs are not simple ones (Syverson, 2011).

Another risk is that your security may be different with respect to different adversaries with different capabilities. It's not always the case that adversaries are a strict superset of one another. For example, two attackers might have different risk tolerance. A technically weaker attacker—say, a criminal running malware—might be more willing to take risks than a technically more powerful attacker such as an intelligence agency. There is also a risk that a threat is anchored in defenders' minds. That is, they become overly-focused on a threat to the exclusion of others. For example, privacy threats from smart meters are complex to defend against, and easily understood. There is a set of threats to privacy that are easier to execute and easier to defend against, but the more subtle surveillance threats get more attention (Danezis, 2011). A real-world

example of this can be found in the Predator drone system. The story is well summarized in a *Wired* article:

> *The original Predator, just 27 feet long, was little more than a scaled-up model plane with an 85-horsepower engine. It had a payload of just half a ton for all its fuel, cameras and radios. And encryption systems can be heavy. (Big crypto boxes are a major reason the Army's futuristic universal radio ended up being too bulky for combat, for example.) With the early Predator models, the Air Force made the conscious decision to leave off the crypto. The flying branch was well aware of the risk. "Depending on the theater of operation and hostile electronic combat systems present, the threat to the UAVs could range from negligible with only a potential of signal intercept for detection purpose, to an active jamming effort made against an operating, unencrypted UAV" the Air Force reported in 1996. "The link characteristics of the baseline Predator system could be vulnerable to corruption of down links data or hostile data insertions."*
>
> "Most U.S. Drones Openly Broadcast Secret Video Feeds" (Shachtman, 2012)

The threats are easily found via STRIDE or other threat modeling, and include signal intercept (information disclosure of location), active jamming (DoS), corruption of downlinks (tampering, information disclosure), and hostile data insertions (EoP). The standard approach to mitigating network threats is the use of cryptography. Thus, the mitigations are also well understood, with the possible exception of location tracking, for which military aircraft typically use spread spectrum or narrow-beam communications.

However, the mitigation description in the article is slightly inaccurate. It is more clearly stated as "NSA-approved encryption systems can be heavy." The NSA wants a variety of shielding and self-destruct mechanisms to address threats of electromagnetic emissions and disclosure of cryptosystems or keys. These are reasonable threats to mitigate in the NSA's book, but as a result of requiring these as a "package deal," the version 1 system used no crypto. Application compatibility concerns have kept crypto out ever since. This is a great example of allowing the best to be the enemy of the good. It might have been possible to deploy with publicly available cryptographic systems without additional layers of protection in the first releases, and then add more protections later. (There are possible concerns about key management, which are likely solvable with public key cryptography. Again, if you allow the best to be the enemy of the good, you end up with control channels in the clear.)

Closing Perspectives

As this book draws to a close, there are two topics that remain. The first is the claim that the threat model has changed, and what you can do with such a claim.

The second is the matter of artistry in threat modeling. It will likely surprise no one that I am fond of threat modeling, and see room for artistry.

"The Threat Model Has Changed"

If I had a dollar for every time I'd heard that "the threat model has changed" in the last few months, I certainly wouldn't tell readers of this book where I was keeping them. It's worth breaking this claim down a little, and understanding what it means.

So first, what's "the threat model?" This phrase often means some set of threats that everyone needs to worry about. And over the last 50 years, that's been revolutionized at least twice. One revolution was the rise of personal computers, when anyone could obtain a computer that was under their control and use it as a base of operations. The second was the rise of interconnected networks. Most important was the Internet, but the rise of networks in general made it possible for remote attackers to come after you. There's a good case that the extension of the Internet and inexpensive computers to the poorest parts of the world represents a third such revolution, in which our assumptions about the economics of attacks have been shattered.

There are other, smaller, but still important changes recently. These include the rise of criminal markets, the normalization of the Internet as a battlefield, and the rise of online activism. There has been a rise of online marketplaces where specialists in techniques such as exploit development, phishing, or draining bank accounts can buy and sell their skills. This rise in the efficiency of attackers and their ability to collaborate is clearly important, although how much the change in attacker economics will change the threat model is still playing out. Another important change is the apparent willingness of governments to invest in the continuation of politics by "cyber" means. Some label this war, others espionage. I think it is probably something new, and the ways in which organizations can find themselves under sustained, persistent attack by paid attackers is a change to the threat model. The third change is the rise of online activists. Although the term "hacktivism" was coined in 1996, the eagerness of people around the world to take up attack tools and apply them seems new and different.

There is another sense in which the phrase "the threat model has changed" is used. It also means the threats which matter enough to influence requirements has changed. This is how many people are using it in the wake of Edward Snowden's revelations about Internet spying. We have long known (in general terms) that the Internet is either heavily monitored or highly susceptible to such monitoring, and that knowledge was insufficient to motivate action. Many of those asserting that the threat model has changed are really asserting that the valid requirements have changed.

Not all of the revelations are things we knew. For example, efforts by US agencies to introduce weakened cryptosystems into American standards represent

new threats. (This is simplified for the sake of an example.) Nonetheless, many of the revelations are changing the requirements end of the threat model, rather than revealing new threats.

Lastly, there is a tension between two perspectives. The first is that "all this has happened before, and all this will happen again." In many ways, the attacks change slowly, and new threats are often slight variations on the old threats. The other perspective is that attacks always get better. Both are right. What is perhaps most important is when someone tells you that the threat model has changed, you understand the ways in which it has changed, and what that may require or enable you to better secure.

On Artistry

"If you have to ask what jazz is, you'll never know."

Louis Armstrong

Many security experts learned to threat model the way blues or jazz musicians would learn to play: apprenticed to a master. This has many advantages, and a huge downside. The downside first: It's hard to decide if the heroin addiction is part of what you need to learn. More to the point, it's hard to know if the masters are doing it the way they do it because that's the best way, or because that's how they learned it.

An early reviewer commented that this book turns threat modeling into a mechanical exercise, and takes the art out of it. For the early parts of this book, that is correct, intentional, and essential. Truly excellent threat modeling is not something that everyone can achieve. Anyone can pick up a camera and take a picture. It's easy to produce a .jpg file that captures the light in a scene. With an understanding of technique, deliberate practice, and critiques someone can regularly produce a decent picture. With all that and talent, they can produce great pictures. This book introduces the practitioner to the techniques and provides guidance and structure; but all the books in the world don't obviate the need to practice and learn. Systems to be threat modeled abound. Find a teacher or even a partner student and take them on. Competency requires a mastery of the "tools" that can only result from using them. It requires critiques and asking how you can do better. It requires someone telling you where your work is lacking. With decent tools and feedback, anyone can become competent.

It's perfectly reasonable to want to aim higher than that, and many of the people who read the closing chapters of a book on threat modeling are likely aiming higher—and to them I want to say that the mechanical aspects are necessary but not sufficient.

To produce art requires practice and experience. It may be that you're the Salvador Dali of threat modeling. That's awesome. As anyone who's ever really studied a Dali painting knows, Dali had incredible technique to back up his

talent. He didn't just say, "Hey, this would be a fun image." He made the clock melt on canvas. Anyone can put a urinal in a room and call it art. Developing a deep understanding of what makes a threat model good and what helps an organization deliver good threat models is part of greatness, and technique provides a needed foundation for your artistry.

Another interesting question is should an organization aim for process or artistry? In a fascinating Harvard Business Review article ("When Should a Process Be Art?"), Joseph Hall and M. Eric Johnson take aim at this question. They say when a process has variable inputs and customers value distinctive outputs, an artistic process might be a good answer. In their model, an artistic process is one where highly skilled professionals exercise judgment. They give the example of Steinway pianos, each made with wood whose variation impacts the instrument. Threat modeling certainly has varied inputs of requirements and software, and the output that's most helpful can be different. So it may well be a good candidate for an artistic process if your organization can support one. The essence is to figure out where artistry is appropriate, to create processes to support and judge the artistic work, and to periodically re-evaluate the balance between art and science. See (Hall, 2009) for more.

Summary

There is a set of prescriptive tools that you can use to design better processes. The first is attention to flow, that state of full engagement that can lead to outstanding results. There is a set of conditions for flow, and each can be considered by a system designer. Doing so requires that you understand who will be executing the tasks using the approach you're designing. People with different backgrounds will have different skills, and thus respond differently to the same challenge. They will also see objects differently. Artifacts such as diagrams and bugs can work as boundary objects, enabling people with different skills and from different disciplines to meet on common ground.

With these tools, and with all the good aspirations in the world, it can be tempting to aim for perfect security. Perfect security is a worthwhile goal, but so is shipping. Unfortunately, sometimes the two goals are at odds, and a team will need to make trade-offs. It's important to not allow the "best," or your highest aspirations for a system, to become the enemies of shipping with good security.

Finding a balance between all these factors is where threat modeling moves from practical and prescriptive toward artistry. This book has focused on the practical and prescriptive, because those have been lacking. The tasks involved in threat modeling have often been too hard. There is a great deal of artistry to be found in finding the best threats, the most clever redesigns, or the most elegant mitigations.

Now Threat Model

So now we arrive at the end of this book, and the start of something better. This book has presented the state of threat modeling as 2013 draws to a close. I hope it has come together in a form that you can use either to begin threat modeling or to improve your threat modeling. Parts II and III should provide enough prescriptive advice and detailed, actionable information that you feel comfortable assembling the "Lego blocks" into something that works for your organization. Part IV on specifics should get you through those gnarly spots. This closing chapter should help you understand what makes a threat modeling approach succeed or fail. I encourage you to now put the knowledge you've gained to good use. Go threat model, and make things more secure.