

# Understanding The Four-Question Framework for Threat Modeling

---





Shostack + Associates White Paper #5  
November 2024  
by Adam Shostack



# Context

“The Four Question Framework for Threat Modeling” provides a common language for more effective communication across a business and enables organizations to “measure twice and cut once” as you develop products and services. Related to this, even simply naming a task can surface it and help us understand its importance.

The Four Question Framework for Threat Modeling is made up of these four specific questions:

- >  What are we working on?
- >  What can go wrong?
- >  What are we going to do about it?
- >  Did we do a good job?

The Framework organizes most modern work in threat modeling and informs much of security by design. In understanding the framework, it’s also important to recognize what the questions mean and why they are the way they are.

The questions enable an organization to **focus on goals** and use situationally appropriate ways of finding or recording answers. For example, people will frequently use a whiteboard to discuss what we’re working on and then translate those sketches into a more formal data flow diagram.

The questions **provide a language** that can be used at all levels of an organization. I sincerely hope that everyone in your organization, from your CEO to your interns, can express what they’re working on even if their answers to these questions use different words or focus on different scopes.

The questions are **not technology-specific**. You can apply them to anything from the simple act of walking across a busy street to making corporate decisions to planning a major life change.

People commonly make the mistake of rephrasing the questions. They don’t realize that there are reasons to use the specific framework questions. There’s nuance and intent in the questions, which are meant to be answerable in many ways. Rephrasings often lose nuance, flexibility, or both. Further, consistency in how we say things contributes to consistency in how we do them. Crafting these specific questions was an iterative effort between 2006 and 2014.



# Measure twice, cut once

---

The Four Questions serve the goal of delivering better results. Threat modeling is the “measure twice, cut once” of cybersecurity. When you work with physical goods, you measure twice because it’s expensive to cut material too short and have to replace it. It’s even more expensive to pour concrete in the wrong place. You have to either break the concrete or replan the rest of the building. When you’re working with physical materials this is pretty obvious. But when you’re writing software, the cost in both dollars and morale is less obvious. It’s even more costly when the software interacts with other software. When APIs, file formats, log messages, or other elements are changed, then the cost of managing those changes cascades.

So discovering risks early saves you money. Discovering them late means you have fewer choices, and it frequently leads to arguments about their priority and what to do about them. These arguments destroy morale and waste the time of everyone from individual contributors to executives.

## The Four Questions

---

Let’s delve into each question of the Four-Question Framework for Threat Modeling.



### What are we working on?

Organizations have goals. From “put a man on the moon by the end of this decade, and safely return him to Earth” to “make the UI snappier in this sprint.” Many things obscure or complicate those goals. The simple question “What are we working on?” helps us focus. A simple picture of what we’re working on helps disparate teams get on the same page.

When threat modeling, we often use data flow diagrams to answer the question “What are we working on?” to the point where people describe them as threat model diagrams. It’s true but incomplete. Data flow diagrams predate threat modeling, and threat modeling can be done without a flow diagram. People will frequently use state machines, message sequence diagrams, and bizarre amalgamations. All of this is great when we focus on the question “What are we working on?” rather than “Show me your data flow diagram.”

The form “What are we working on?” can easily be extended to “What are we working on right now?” This aligns well with agile development and the approach from Izar Tarandach, “Threat model every story.” When we innocently vary the question to “What are we building?” we move towards a waterfall view and the dangerous implication that we have to analyze the whole of what we’re building, rather than the part that we’re working on right now. That tends to encompass the parts that have already been built. As discussed in the earlier section, “Measure twice, cut once,” changes to either poured concrete or already developed software are expensive. That can lead to threat modeling being a source of endlessly revisiting decisions and slowing progress.

Early versions of the framework used a “you” form (as in “what are you working on”). That was a very consultant-oriented framing and so it changed to “we.” This is more inclusive and creates participation, rather than framing security as “us and them.”<sup>1</sup>

Saying “we” also encourages us to understand the perspective of the team doing the work. For example, mobile apps and their backends are often built by separate teams, and each team should focus on what they’re working on: the app or the backend. Each team is best able to deal with the threats to the parts of the system they’re working on. In an ideal world, they’ll collaborate and discover the threats to the system as a whole.<sup>2</sup> All threat modeling is done from a perspective. Threat modeling is more effective when we’re explicit about our perspective, for example, “we’re considering threats that are in scope for the mobile application team.”

The last reason to ask “What are we *working on*?” is that the things being worked on are ... being worked on. Those systems have “change energy.” Proposed changes as a result of threat modeling are more likely to align with other work being done, rather than apparently being random bugs in software in maintenance mode.



## What can go wrong?

The question of “What can go wrong?” is not quite universal. Movie villains, children, and some politicians<sup>3</sup> seem to have a hard time grasping that not everything will proceed exactly as they have foreseen. The rest of us can anticipate problems, and we do so often. Creating an organizational norm of searching for and addressing problems is healthy.

Executives and technical staff often have different lenses for thinking about what can go wrong and, lacking a common language, can talk past each other in terms of why their answers are important. Executives may not know what “technical details” lead to the problems they worry about, and technical professionals may have a hard time expressing business impact. Other technical professionals may get caught up in the question of “What’s a threat?”<sup>4</sup> But what’s important is: “What can go wrong?”

Traditional approaches to threat modeling often incorporate mnemonics like STRIDE or structures like Kill Chains. I’ve written an entire book that focuses on STRIDE (*Threats: What Every Engineer Should Learn From Star Wars*, Wiley, 2023), and while I think the contents are useful, threat modeling can and should be accessible to those without that knowledge.

And so by phrasing the question as “What can go wrong?” we expand the space of answers, increase participation, and reduce debates over terminology.

---

<sup>1</sup> As organizational psychologist Roger Waters points out, “God only knows, it’s not what we should choose to do.”

<sup>2</sup> Some security practitioners are haunted by the possibility of missing the “compositional threats,” the ones that get exposed by looking at the forest, rather than the trees, or, perhaps preferably, the forest as it relates to the ecosystem, and so it becomes admirable to incorporate larger and larger views of systems which are ever less influenced by those doing the work. And while that can lead to “interesting” or even fundamental challenges, it reduces the proportion of their analysis which is actionable.

<sup>3</sup> This is a non-partisan perspective.

<sup>4</sup> A threat is the promise of future violence, especially if the target doesn’t do something.



## What are we going to do about it?

There's a range of technical and business activities you take in response to a threat. Strategically, the techniques of risk management are all available. You can mitigate, eliminate, accept, or transfer risk. Mitigation is listed first because often we develop new features or deploy controls. The question is not "How are we going to fix (or mitigate) that?" because sometimes we need to engage in risk management or feature redesign.

The phrase "going to do" implies a future action. Threat modeling happens in the context of other work. Once you've found the issues and are tracking them, threat modeling may be done, and you have items for the backlog. We don't treat fixing bugs as part of the cost of quality assurance work, but some people do treat similar outcomes of threat modeling as part of the cost of threat modeling. Considering the actual work to fix things as part of threat modeling contributes to the perception that threat modeling is heavyweight and time-consuming.

Another outcome of threat modeling can be knowing what we're not going to do and why. "Collecting pictures of IDs in the signup process adds a huge burden to secure the data that we never use again. It's not worth it." And if the analysis is done by developers, then they're more likely to incorporate it into their thinking going forward.



## Did we do a good job?

In devising the framework, the question of "Did we do a good job?" started out meaning "validation," the largely mechanical analysis of "Do we have a diagram," "Did we find some threats," and the like. That extended to the questions of "Did we file bugs/tickets?" and then "Did we fix the problems?"

As I've worked with more companies to adopt threat modeling, I've seen the value of giving people a chance to reflect on their work, both as part of structured training and as they start to do the work on real systems. Giving people the opportunity to consider if they've done a good job is an emotionally critical part of enabling change as is ensuring that they see the work as worthwhile. It's important to keep in mind that any new task is frustrating and any change is hard. On your first day at the gym, all you get is soreness. Many people never go back.

When considering whether we did a good job, we can look at specific tasks. For example, did we do a good job at modeling the system, at modeling the system to come to a common understanding, or at documenting our intent in a way that makes customers happy? Is it time to move beyond asking "What can go wrong?" and add structure to how we ask the question here?

As you work to adopt threat modeling, asking simple questions can be revelatory: Would you threat model again? Would you recommend threat modeling to your colleagues?<sup>5</sup> If people say no, understanding why will help you address the concerns. If they say yes, then they're getting value from the work and will likely keep doing it. And if they say they'd do it again but do not recommend it, understanding why may help you improve a process definition.

Overall, the "good job" phrasing encompasses two meanings: "Were we effective?" and "Were we efficient?" Efficiency comes with practice and reflection. Additionally, we often have very high hopes for a new practice. The value we get from the work may well be less than we hope but worthwhile in and of itself. The question of "Did we do a good job?" allows us to continue assessing tradeoffs between efficient and effective.

<sup>5</sup> This is a variant of the question used for Net Promoter Score (NPS). Those familiar with that system often default to using the numerical scoring system, but simply asking the question can lead to a conversation.

My colleagues on the Threat Modeling Manifesto rephrased this question as “Did we do a good *enough* job?” A focus of the manifesto was organizations getting started on threat modeling, and the “good enough” version of the question refocuses attention on an immediate assessment of goodness and for us to look at “good enough” without some arbitrary measure of “good.”

While that “good enough” framing can be beneficial, it does reduce attention on the long-term view of a good job because it’s hard to assess what good enough might be. This illustrates an important tie to the broader Secure By Design initiative from CISA, which has a strong focus on a technical property called memory safety. Memory safety issues are at the heart of many vulnerabilities. CISA’s success will lead to a large reduction in the number and severity of issues found, and so the issues that threat modeling exposes will increase in both number and importance.

As time goes on and opting to not threat model becomes a puzzling, bizarre, or alarming choice, we’ll move from asking “Would you threat model again?” to “Would you use that technique again?”

When we take a long-term view of threat modeling, we can also consider the question of “Did we do a good job?” over time. Are we seeing less re-work, better collaboration, or fewer escalations? Are we seeing fewer issues (or less severe ones) from penetration tests, bug bounties, or incident reports?

## Language

The Four Question Framework enables us to put labels on threat modeling and its component tasks, which helps us have a consistent conversation about that work. That conversation can and should span the organization.

### Naming a practice

Simply naming a practice can surface it and help us understand its importance. For example, Goldratt’s *The Goal* popularized the idea that “work in progress” piles up and focuses attention on delivery and throughput. Naming mentoring as an activity helps formalize and increase access to otherwise implicit career advice. Naming the emotional labor of helping people resolve conflicts and working on team morale and more helps managers note that such essential work is rarely rewarded.

Threat modeling can also be unnamed work done by some people as part of their conception of good engineering. The act of naming and labeling threat modeling enables you to have a conversation about its value, normalize the practice, and even use it to create an organizational language.

### An organizational language

The questions of “What can go wrong?” and “What are we going to do about it?” are crucial to security. And they’re not limited to security. Every project has risks, and every organization needs to deal with them. Within the security world, we have high-precision tools that we’ve developed for consistency. A side effect of that consistency is that our tools can be seen as jargon-heavy, in the weeds, or otherwise unhelpful. The Four Questions enable a conversation that crosses boundaries and spans from leadership to individual contributors.

# Conclusion

The value of measuring twice before we cut physical material is obvious. The value of doing so in software can be less obvious, either because it's not clear how or because the tasks can be expensive. The Four Question Framework has evolved into a leading tool for aligning threat modeling practices by addressing both concerns. The specific phrasing of the questions is the result of conscious evolution. Focusing on goals, rather than techniques, smooths the road to delivering more secure systems.



## ACKNOWLEDGEMENTS

Matt Coles, Avi Douglan, Irene Michlin, Izar Tarendach, and Kim Wuyts all provided useful feedback.

Like many other useful inventions, the Four Question Framework started somewhat accidentally, and only through observation and with honing and careful shepherding did it reach its current form. The process took several years. The "References" section at right includes a non-exhaustive history of the Framework. To put a few of those items in context, many Microsoft employees did important threat modeling work from roughly 1999 to 2012, including Kohnfelder and Garg (1999) and Swiderski and Snyder (2002). There are many articles from those years that include predecessors to the Four Question Framework, including Hernan (2006) and Shostack (2008b). A 2008 MSDN article "Reinvigorate your Threat Modeling Process" shows a five-stage process (Figure 1 of Shostack 2008a), and a conference talk explicitly discusses a "four-step process" (Shostack, 2008c). But the framework is not important enough to be mentioned in a 2010 Blackhat talk or a 2012 3GSE paper. Modern versions first appeared in *Threat Modeling: Designing for Security* and have been largely consistent since then. This list is a sampling of the discussion that happened, and omissions don't reflect any negative judgment.

## REFERENCES

**Cybersecurity Infrastructure and Security Agency.** *Secure by Design Resources*, October 25, 2023, [www.cisa.gov/resources-tools/resources/secure-by-design](http://www.cisa.gov/resources-tools/resources/secure-by-design)

**Eliyahu M. Goldratt and Jeff Cox.** *The goal: a process of ongoing improvement*. Gower, 1989.

**Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack.** *Uncover Security Design Flaws Using The STRIDE Approach*. MSDN Magazine, November, 2006.

**Loren Kohnfelder and Praerit Garg.** "The threats to our products." *Microsoft Interface*, Microsoft Corporation 33 (1999).

**Adam Shostack (2008a).** *Security Briefs: Reinvigorate your Threat Modeling Process*. MSDN Magazine. July 2008.

**Adam Shostack (2008b).** *Experiences Threat Modeling at Microsoft*. MODSEC@ MoDELS 2008 (2008): 35.

**Adam Shostack (2008c).** *SDL Threat Modeling: Past, Present and Future*, Toorcon, April 2008

**Adam Shostack.** *Elevation of Privilege: The easy way to threat model*, Blackhat 2010.

**Adam Shostack.** *Elevation of privilege: Drawing developers into threat modeling*. 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14). 2014.

**Frank Swiderski and Window Snyder.** *Threat modeling*. Microsoft Press, 2004.

**Izar Tarandach.** *Continuous Threat Modeling*, 2019, [github.com/izar/continuous-threat-modeling](https://github.com/izar/continuous-threat-modeling)

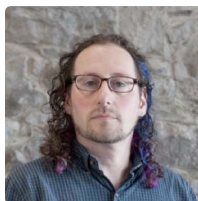


## ABOUT SHOSTACK + ASSOCIATES

Adam Shostack founded the company that bears his name in 2016. Shostack + Associates now focuses on delivering great learning experiences, primarily around threat modeling, including classic training and also helping leaders learn to navigate the complex organizational changes that often surround threat modeling.

### Get In Touch

If threat modeling isn't delivering what you hope for, then it's our hope that this paper will help. If we can help further, please don't hesitate to reach out for a confidential consultation, at [adam@shostack.org](mailto:adam@shostack.org).



## ABOUT ADAM SHOSTACK

Adam is the author of *Threat Modeling: Designing for Security* and *Threats: What Every Engineer Should Learn from Star Wars*. He's a leading expert on threat modeling, a consultant, expert witness, and game designer. He has decades of experience delivering security. 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 hundreds of millions of systems
- > Led the design and delivery of the Microsoft SDL Threat Modeling Tool (v3)
- > Created the *Elevation of Privilege* threat modeling game
- > Co-authored *The New School of Information Security*

Beyond consulting and training, Shostack serves as a member of the Blackhat Review Board, an advisor to a variety of companies and academic institutions, and an Affiliate Professor at the Paul G. Allen School of Computer Science and Engineering at the University of Washington.



## LICENSE

This document is copyright ©2024 Shostack + Associates.

The content is licensed under a Creative Commons Attribution-NonCommercial License 4.0  
<https://creativecommons.org/licenses/by-nc/4.0>.

The Four Question Framework is licensed as CC-BY. (Some lawyers worry that using the *complete* Framework may not be fair use.)