Dr. Dobbs Jolt Award Finalist 2014

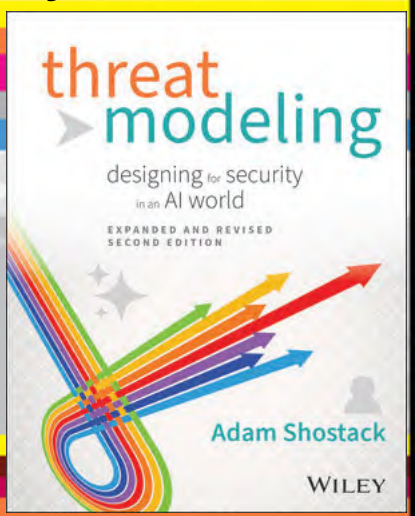
Adam Shostack
Microsoft's Threat Modeling Expert

Chapter 11 "Threat Modeling Tools" 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

11

Threat Modeling Tools

This chapter covers tools to help you threat model. Tooling can help threat modeling in a number of ways. It can help you create better models, or create models more fluidly. Tools can help you remember to engage in various steps, or provide assistance performing those steps. Tools can help create a more legible or even beautiful threat model document. Tools can help you check your threat model for completeness. Finally, tools can help you create actionable output from a threat model.

Tools can also act as a constraint. You may find yourself stymied by usability issues, such as fields you're unsure how to fill out. Or you might find that a tool cramps your style. Some trade-offs are unavoidable as tools are created, so the chapter starts with general tools that are useful in threat modeling, and then progresses to more specialized tools.

A few disclosures: I do not have personal experience with each tool described here, and some of the tools I created myself. (Those are treated at greater length, because there's less risk of me insulting the authors.)

This chapter starts by describing some generally useful tools and how to apply them to threat modeling. You'll then learn about the open-source tools that are available, followed by commercial tools. The chapter closes with a few words about tools that don't yet exist.

Generally Useful Tools

This section discusses tools that are not specialized for threat modeling but can be tremendously useful. It covers a few of the more useful tools to encourage you to think about the tools you already use and with which you are familiar.

Whiteboards

I can hardly imagine threat modeling without a whiteboard. No technology I've used has the immediacy, flexibility, and visibility to a group than a whiteboard when iteratively drawing system architecture. Whiteboards also have the advantage of transience—drawing on paper just isn't the same. On a whiteboard, no one tries to correct details such as a line not being connected properly, so the discussion can be focused on how the system actually works.

For distributed teams, a webcam focused on a whiteboard may work, or you may have "virtual whiteboarding" technologies that work for you.

Office Suites

Microsoft Office contains a number of tools that are very useful in threat modeling. Word is a great tool for recording threats in free-form. What to record is dependent on the approach you've chosen. Excel can be used for issue tracking and status. Visio is great for turning whiteboards into more precise documents. Of course, Office is one of several suites with word processing, spreadsheet, and drawing functionality. The only caveats would be the limitations of the tools. The document tool should be more than text—a feature such as embedded images is extremely useful. Similarly, use a vector drawing tool that enables you to move symbols as symbols. Automatic connector management is also super-useful, and of course this feature is not unique to Visio.

To state the obvious, Microsoft Word, Excel, and Visio are commercially licensed tools.

Bug-Tracking Systems

Whatever bug-tracking system you use should also be used to track threats. A good bug from threat modeling can take many forms. The form you use will influence how you title and discuss bugs, and there is no universally right way to approach it. (The right way is the way that works best for you and your organization.) The title could express any of the following:

- **The threat itself**: Here the bug title is of a form such as "an attacker can threaten the component" or "the component is vulnerable to threat."

For example, "the front end is vulnerable to spoofing because we use reusable passwords."

- **The mitigation:** Here, the bug title is of a form such as "the component needs mitigation." For example, "the front end needs to run only over SSH." In the text of the bug, you should also explain the threat.
- **The need to test a mitigation:** This is what you can title a bug if someone says, "Oh, the front end isn't vulnerable to that." Rather than absorb time in the meeting to discuss or check the threat, file a bug, "Test front end vulnerability to *threat*" and ensure that there are good tests for the bug.
- **The need to validate an assumption:** These bugs are filed to ensure that someone follows up on an assumption you discover while threat modeling, and on which you depend for a security property. The bug should have a title such as "security depends on assumption A" or "security property X of component Y depends on assumption Z." For example, "Security depends on the assumption that no one would ever find the key in the fake rock that looks exactly like the rocks at our last house."
- **Other tracking items:** You should treat the preceding items as suggestions, not a form into which all bugs need to fit. If you find something worth tracking, file a bug.

When tracking security bugs from threat modeling, there are a few fields that can make running queries and analysis more reliable. These include whether the bug is a security bug, whether it's "stop ship," and how the bug was found (for example, threat modeling, fuzzing, code review, customer report). You can also check the tables in Chapter 7, "Processing and Managing Threats," and Chapter 9, "Trade-Offs When Addressing Threats," for useful fields. For example, you might want a risk management approach field whose values could be avoid, address, accept, or transfer.

The right fields to use will depend in large part on the queries you want to run, which of course depend on the questions you want to ask. Some questions you might want to ask include the following:

- Do we have any open security bugs?
- Do we have any open threat modeling bugs?
- Do we have any high-severity threat modeling bugs left to fix?
- How much risk are we transferring to end-users in the security operations guide or via warning dialogs?
- What department head has signed off on the largest business risk? Which department head has signed off on the most risks?

Open-Source Tools

A variety of open-source tools for threat modeling are available. The open source tools illustrate some of the challenges in creating a high-quality threat modeling tool.

TRIKE

There are two tools named TRIKE. The first was a standalone desktop tool, written in Smalltalk. That tool is no longer being maintained, and TRIKE is now implemented in a spreadsheet. According to documentation, it works best in Excel 2011 for the Macintosh (Trike, 2013). TRIKE is sometimes referred to as "OctoTrike."

TRIKE does not fit cleanly into the four-stage framework defined in this book. The TRIKE spreadsheet contains 19 pages, which are grouped as follows: one overview, seven main threat pages (actors, data model, intended actions, connections, protocols, threats, and security objectives), four record-keeping pages (use case index, use case details, document index, and development team) and seven reference sheets (actor types, data types, action, network layers, meaningful threats, intended response, and guide words). As of this writing, the help spreadsheet appears to be a reference document, not an introduction of the system.

SeaMonster

SeaMonster is an Eclipse-based attack tree and misuse case tool that was developed by students at the Norwegian University of Science and Technology. It appears to be abandoned since 2010 (SeaMonster, 2013). The code is still available.

Elevation of Privilege

Elevation of Privilege (the game) is designed to be the easy way to get started threat modeling. It works by inviting individuals to participate in a game. The game consists of 74 physical playing cards in six suits, named for the STRIDE threats, with most suits having cards 2 through Ace. Two suits have fewer cards in order to avoid redundant threats, and it was challenging to find broadly applicable threat instances that were easily explained on a card. Each card has a specific instance of a STRIDE threat. For example, the 6 of Tampering reads "An attacker can write to a data store your code relies on." Another example card is shown in Figure 11-1.

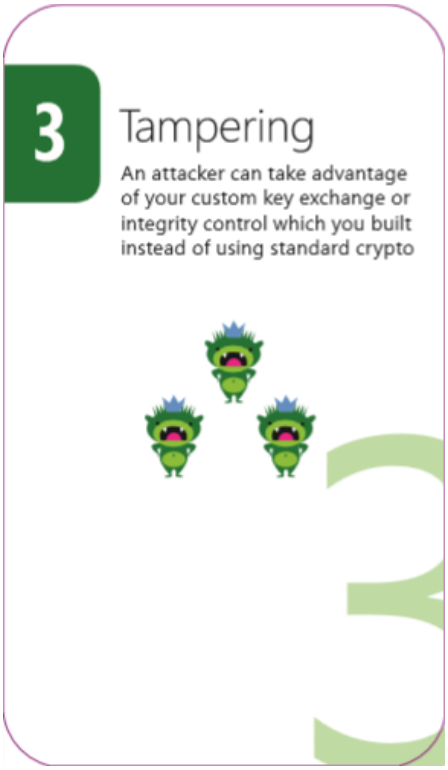


Figure 11-1: An *Elevation of Privilege* card

The *Elevation of Privilege* files can be downloaded from `shostack.org/games/elevation-of-privilege`. Before starting *Elevation of Privilege*, participants or the game organizer create a diagram of a system being modeled. People then come together for a game. The organizer explains the rules, and may ask people to "put their skepticism on hold." The game starts by dealing out the deck, and is then structured into turns. The first card played is always the 3 of Tampering. Play proceeds around the table in hands.

Each hand starts with a player selecting a suit to lead and playing in that suit. Each player plays by selecting a card and connecting it to the diagram. The player must play in the suit that was led if they have a card in that suit. If they don't, they may play any card. When play has gone once around the table, the hand ends. The player who played the highest card wins the hand. The highest card is either in the suit that was led, or, if a card in the Elevation of Privilege suit card was played, the highest card played from the EoP played wins the hand. (All Elevation of Privilege threat cards are higher ranked than the suit that was led,

and only Elevation of Privilege cards can win when someone leads in another suit.) Players get a point for connecting the threat on their card to the diagram with a "buggable threat," and a point for winning the hand by playing the highest card either in the suit that was led or in EoP. Any EoP card trumps the suit that was led. To encourage creativity, each ace card says "You've invented a new threat," and the threats are enumerated on cards included in the pack. The game ends either when time allocated has elapsed or when all the cards have been played. The winner is the player with the most points.

A buggable threat is one a team identifies and is willing to file a bug for. It's a simple and implicit element of most software development. Some teams may find it more helpful to ask "would we add that to the backlog?" However you want to approach it (in the context of the game), you want an understandable and shared bar to test threats, so that you focus on finding the good ones.

Games are less threatening than "serious" work, and they provide structure and hints to the beginner, enabling new players to find a threat based on the cards in their hands. The game is also intended to help players find a *flow state*, a concept that is covered in depth in Chapter 19, "Architecting for Success." EoP is covered in greater depth in my paper *Elevation of Privilege*: *Drawing Developers into Threat Modeling* [Shostack, 2012].

Microsoft makes the files (source and PDF) available under a Creative Commons BY-3.0 license, allowing you to take it, modify it, make derivative works, and even sell them.

Commercial Tools

Here are a few commercially licensed threat modeling tools. I mention a few commercial tools as examples, but *caveat emptor*.

ThreatModeler

ThreatModeler from MyAppSecurity.com is a defense-oriented tool based on data elements, roles, and components. It uses a set of attack libraries, including the MITRE CAPEC (see Chapter 4, "Attack Trees"), the WASC threat classification, and others. The tool generates attack trees with the component as the root, requirements that can be violated as a first level of subnode, and then threats and attacks as the next layers. According to the documentation, ThreatModeler is intended to be used by architects, developers, security professionals, QA professionals, or senior executives. ThreatModeler requires Windows.

Corporate Threat Modeller

Corporate Threat Modeller from SensePost is a tool built to support a methodology designed after an analysis of the strengths and weaknesses of a number

of threat modeling approaches. Those approaches included threat trees and OCTAVE, a US-CERT-originated system for threat modeling a business (White, 2010). The analysis also looked at Microsoft's SDL Threat Modeling Tool v3, and the Microsoft "IT Infrastructure Threat Modeling Guide," (McRee, 2009) which shows how to use STRIDE-per-element to threat model IT infrastructure.

The Corporate Threat Modeler was explicitly designed for consultants. Insofar as it was developed with an explicitly stated target user (not "everyone"), it is one of the most interesting tools on the market. The approach starts with an architectural overview, and then applies a somewhat complex risk equation. The tool is free to download.

Secur*IT*ree

Secur*IT*ree is threat risk software from Amenaza Technologies, which launched in 2007 to positive reviews (SC Magazine, 2007). The product seems like a well thought through tool for constructing, managing and interpreting threat trees. It contains not only the ability to manage trees, but a set of ways to filter those trees. Each node in the tree has behavioral/capability indicators: a cost to execute, noticability, and a technical ability. It also has impact/attacker benefit indicators of attacker gain and victim loss, along with stored notes for a node or subtree. You can filter the tree based on a given attacker ability. Secur*IT*ree comes with a library of threat trees, which is likely to help its customers get to the interesting part of the threat modeling work faster. Secur*IT*ree also includes some excellent screencast-delivered training (Ingoldsby, 2009). Secur*IT*ree runs on Windows, Mac, and Linux.

Little-JIL

If you're making use of threat trees at a research institution, the Little-JIL software may be helpful. "Little-JIL is a graphical language for defining processes that coordinate the activities of autonomous agents and their use of resources during the performance of a task." It has been used for creating an elections process model and a set of fault trees for that model (Simidchieva, 2010). The full fault trees are available as a graphML model. The software used to create and process the models may be freely used at research institutions (Laser, undated).

Microsoft's SDL Threat Modeling Tool

Microsoft has shipped at least four families of threat modeling tools. They are the *Elevation of Privilege* card game, the SDL Threat Modeling Tool v3, the Threat Analysis and Modeling Tool, and the Threat Modeling Tool v1 and 2. I was the project lead for *Elevation of Privilege* and the SDL Threat Modeling Tool v3 and 3.1. The currently available SDL Threat Modeling Tool is (or has been) available free from Microsoft.

The SDL Threat Modeling Tool v3 was designed in reaction to the complexities and usability issues encountered when engineers who were not threat modeling experts tried to use the older tools. It was the first tool designed around the four-stage framework. The tool has four major screens, designed around the tasks that naturally fit together: Draw Diagrams, Analyze Model, Describe Environment, and Generate Reports. The Draw Diagrams screen, shown in Figure 11-2, includes both the capability to draw diagrams with a constrained Visio stencil set and a diagram validation section with heuristics. The Analyze Model screen, shown in Figure 11-3, is automatically filled out with threats according to STRIDE-per-element.

Each threat instance contains a set of guiding questions to help engineers think through the threat, and an area to record the threat, mitigation, to track whether work on the threat is complete, and to file a bug. The Describe Environment screen is something of a catch-all to track assumptions, external notes and the context of the threat model. The Reports screen includes an all-up report, an open issues report, a list of bugs, and a diagrams-only report intended to facilitate printing. The tool also contains a manual, a sample threat model (for the tool itself), and a getting started guide, all accessible via the Help menu.

As shown in Figure 11-2, the main Draw Diagrams screen contains the following, clockwise from upper left (excluding the menu):

- The diagrams control, enabling you to create sub-diagrams
- The Visio shapes you can use as diagram elements
- The "default" diagram (discussed in the next paragraph)
- The numbered Screens control
- Diagram validation (feedback)
- A help pane

The default diagram is present because human factor testing has shown that less experienced threat modelers are sometimes stymied by a blank screen. Providing them with a starting diagram serves two purposes. One, it demonstrates what is expected in that space. Two, rather than needing to create a diagram, a novice can modify what's there, which is an easier task.

One other feature worth mentioning from Figure 11-2 is the help field. Generally, help is a menu option that software engineers ignore, because they believe they're too smart to need to read what they expect will be a badly written help file. Therefore, the tool has basic help onscreen.

The Analyze Model screen shown in Figure 11-3 has two panes. The left pane is a list of diagram elements and the threats associated with them, presented as a tree with a single level of branches. The right pane is titled with the element name ("Results") and a description of the element. Under that is a reminder of the STRIDE-per-element threats to which it is subject, and an option to not

generate threat placeholders, with a reason box. A large portion of the screen is devoted to onscreen guidance: "Some questions to ask about this threat type." The guidance is specific to threat and element category (process, data store, data flow, or external entity).

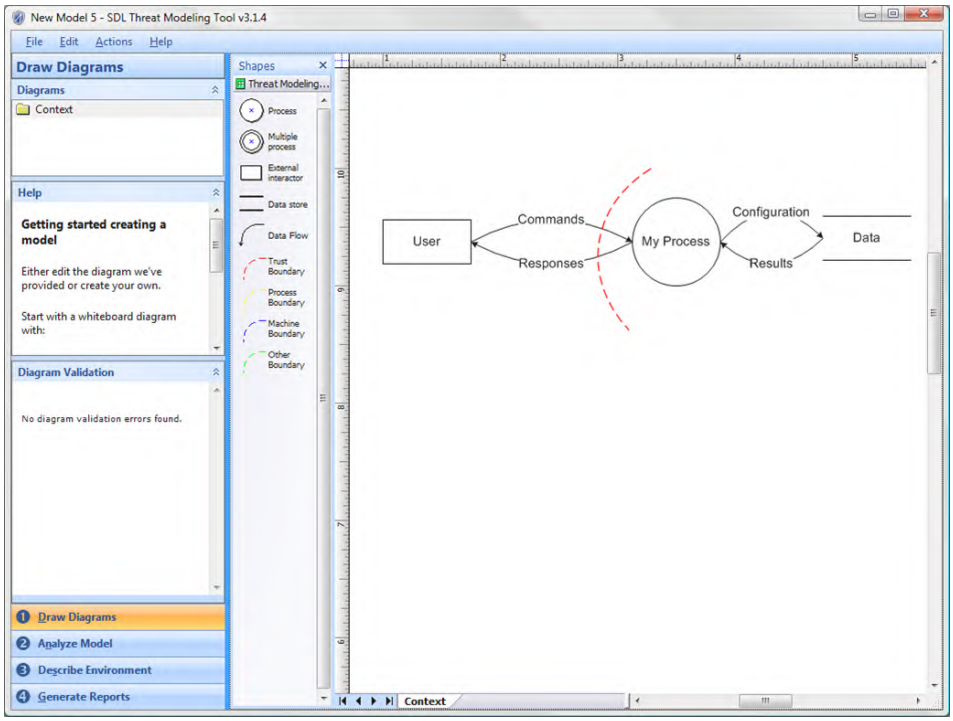


Figure 11-2: The SDL Threat Modeling Tool "Draw Diagrams" screen

There's also a command link to "Certify that there are no threats of this type." The word *certify* was carefully chosen to convey gravity. The last element on the screen is where the threat modeler describes the threat impact, and how it will be mitigated. Most of that is done in two large text entry boxes, but there is also a Finished check box, a "file bug" command link, and a completion bar. The completion bar (shown empty, under the word *completion*) fills out in four segments to encourage text entry in the Impact and Solution fields, as well as checking "finished" and filing a bug. There is also an Add Threat command link in case someone discovers an additional tampering threat against the results data flow.

The bug filing is intentionally abstracted into an API, and the tool ships with sample code to connect to a variety of bug tracking systems, or allow you to connect to whatever you use. When developing the tool, we intentionally spent time to create an API and to ship it under the Microsoft Public License (an Open

Source Initiative Approved License), because bugs are a critical part of ensuring that the threat model leads to something actionable.

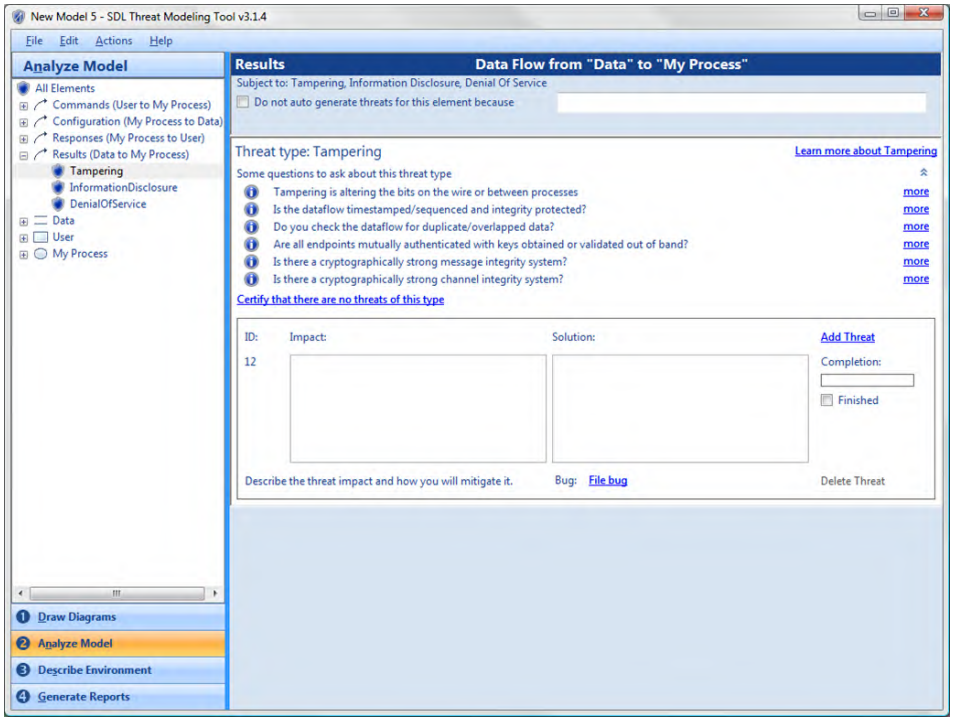


Figure 11-3: The SDL TM Tool Analyze Model screen

NOTE **Within the Draw Diagrams screen, the term "validation" causes consternation when diagrams don't conform. For example, one heuristic is to show where all data comes from. That can be addressed by including three extra elements: an installer external entity, a data flow, and a trust boundary. Doing so is not obvious, and requires roughly 10 steps. In a future version, it would be great to see diagram feedback that includes advice on how to address each. Also, boxed trust boundaries would be one of several improvements that could be made to the shapes in the default set. Others include rounded rectangles for processes, less curvy lines, and better positioning of text.**

The SDL TM Tool v3.1 series is a no-cost download from Microsoft (`www.microsoft.com/security/sdl/adopt/threatmodeling.aspx`) and it requires Visio 2007 or 2010 to use. The tool is compatible with the Visio 2010

evaluation version. Newer versions of the tool may become available (after this book goes to press) with different dependencies.

Tools That Don't Exist Yet

There are two categories of features that people often ask for that are worth a brief discussion: automated model creation and automated threat identification. A great many people want tools that can take a piece of software that's already been written and extract a data flow or other architectural diagram. This is an attractive goal, and one that might be feasible for programs written in strongly typed languages. Marwan Abi-Antoun has done some work showing how to extract data flow diagrams for Java (Abi-Antoun, 2009). (The system with open code and published DFDs he found to use for testing was only 3,000 lines of code.) If technology to do this is further developed, it will present great value to threat modeling, but also create a temptation to not perform any threat modeling or analysis until late in a project. Threat modeling after the code has been completed limits options for addressing issues. This is discussed further in Chapter 7, "Processing and Managing Threats" and Chapter 17, "Bringing Threat Modeling to Your Organization."

Similarly, tools that can take a diagram or other model and produce lists of threats would be lovely. A Spanish graduate student, Guifré Ruiz, and colleagues have created a first version of such a tool (Ruiz, 2012). However, these tools carry a risk that security analysis will focus only on known threats from an attack library. Such tools cannot (currently) analogize from closely related threats the way an experienced person can. Threat analysis that could reliably extend that knowledge to prevent new systems from making mistakes others have made would be a useful step forward. As more such tools are developed, it will be important to consider the balance between human and automated security design analysis. After all, to the extent that you need software engineers to create new functionality, that new functionality and the new combinations that result may expose new threats. It's not impossible to imagine a tool that would find threats against code not yet written, but it's hard to imagine one that would do so as comprehensively as an expert threat modeler.

Summary

A wide variety of tools are available for threat modeling. General-purpose tools such as whiteboards and bug-tracking systems can be very helpful, and tools such as word processors, spreadsheets, and diagramming tools can be used to

help you threat model. Also available are a variety of specialized threat modeling tools. Microsoft has shipped several of these free, including *Elevation of Privilege* and the SDL Threat Modeling Tool, and you can find other commercial and open-source tools that may aid your efforts. There is also demand for tools that can automate model creation or threat identification, although such tools may come at a high price if they appear to find threats while missing new threats or are used too late in the development process.