Introduction to Software Security

Chapter 2.2: Overview of Threat Modeling

Elisa Heymann
elisa@cs.wisc.edu

Barton P. Miller
bart@cs.wisc.edu

Loren Kohnfelder
loren.kohnfelder@gmail.com

DRAFT — Revision 3.0, August 2023.

Objectives

- Understand the purpose and goals of threat modeling.
- Learn the key provisions of the Threat Modeling Manifesto.
- Understand why threat modeling is an essential part of securing any software system.

Overview

Threat modeling informally happens frequently in many facets of our lives. Everytime we think “what can go wrong?” and plan accordingly, we are performing Threat Modeling. For example, perhaps you need to catch a flight out of Chicago O’Hare airport early on a Monday evening and you live 150 miles away. You try to anticipate the different issues that might happen on the way to your flight, such as traffic, construction, a flat tire, adverse weather, or long security lines. To mitigate the different threats, you determine that you need to leave for the airport five hours before your flight. That time might give you slack in the case that any of those threats become real, but will not have you waiting at the airport for an excessive amount of time (the very real threat of much wasted time). Note that those five hours would not be enough in the worst case scenario, if all those threats became real such as an accident causing 70 miles of heavy traffic, then having a flat tire, and then more traffic because of road work. However, you determine that the likelihood of all those threats happening on the same trip is low, so your mitigation strategy consists of leaving five hours before your flight. There is no need to attempt to completely neutralize all possible threats, and it is probably impossible to do so perfectly, but it is still valuable to do what you can.

In the systems area, threat modeling consists of identifying the potential threats that can affect your system, with the outcome being a list of the possible threats. Ideally, that list would be prioritized, sorted by the risk associated with the threats. As we saw in the Chapter 1.2 (Basic Concepts and Terminology), risk depends on the likelihood of that threat becoming real times the impact that threat would have if it became real.

Before any human or tool can identify any threats, we need a model of the system. That means representing the processes, resources (such as database and files), networks, and the interactions between them. The next step is to identify threats associated with the design of your system. There are tools that help designers to do that, but tools have limitations. Even with their limitations, tools can be a good starting point. After the threats have been identified, the designer/analyst needs to come up with ways of mitigating the relevant threats.

There is always a trade-off between how secure our system will be and the amount of resources we are willing to commit to security. We could try to mitigate every possible threat we can think could affect our system, no matter how low the associated risk value. However, such an effort would require a huge amount of resources in terms of time and personnel, which means that it would be very expensive. The other extreme would be to not care about mitigating any risks. Neither of those two approaches is sensible. Threats that may result in harming your system if they became real should be mitigated in most cases. It is not a trivial task to decide where to draw the line between those that are likely to become real and those that are not.

One way to strike a balance is by weighing mitigations by how easy and safe they are compared to the effectiveness in prevention that would result. That is, even mitigating a relatively lower risk threat makes good sense if it is an easy code fix that is unlikely to introduce additional bugs and would foreclose the possibility of a particular exploit. By comparison, an elaborate mitigation of a greater risk might be best left unimplemented given the effort involved, especially when it might introduce new bugs and likely degrade performance.

So how do we apply this to the design of real software? The good and bad news is that there have been several threat modeling methodologies proposed to guide you in this process. The good news is that you can find a method that fits your circumstances and style. The bad news is that evaluating the various methods can distract your team from making forward progress on improving your software security.

Some of these methodologies are more mature than others, so are more easily incorporated into your software development lifecycle. And there are a variety of views as to what operations are included in threat modeling, which explains some of the variation.

The Threat Modeling Manifesto

A group of leading security practitioners developed a collection of principles that any threat modeling methodology should embody and published it in their Threat Modeling Manifesto[1].

As part of this effort, Adam Shostake defined threat modeling as answer these four questions about a software system:

[1] Z. Braiterman, A. Shostack, J. Marcil, S. de Vries, I. Michlin, K. Wuyts, R. Hurlbut, B.S.E. Schoenfield, F. Scott, M. Coles, C. Romeo, A. Miller, I. Tarandach, A. Douglen and M. French, “Threat Modeling Manifesto”, https://www.threatmodelingmanifesto.org/

1. *What are we working on?*

   Identify the users, system components (active entities), resources (data entities), trust boundaries, and flow of operations or data. This information is often represented in diagram form. Threat modeling tools can provide a structured way to do this step. For a beginner, such a tool can help guide you and provide support creating the diagram. For more experienced programmers and designers, a threat modeling tool can help standardize the style and presentation. However, such a tool is not a requirement for successfully doing threat modeling.

2. *What can go wrong?*

   From the information the first step, project what potential vulnerabilities (threats) can affect the system. A threat modeling tool can help to automate the creation of a threat list.

3. *What are we going to do about it?*

   For each potential threat, identify a means to avoid or prevent the threat. Such a means might involve mechanisms that you need to include in your software or changes to your design.

4. *Did we do a good enough job?*

   We need to go beyond the tool and the model. Not all reports from a threat modeling tool require action and not all threats will generate reports. Human inspection, team discussion, and peer review can all support a more complete threat modeling process, whether it is based on a tool or not.

The Threat Modeling Manifesto advocates several principles including:

Early and often: Threat modeling should be used from day one. Using threat modeling during the design phase, before the first line of code is written, lowers the overhead of dealing with the threats that are found and places the design on a solid foundation. To quote from NIST SP800-218, "Most aspects of security can be addressed multiple times within an SDLC, but in general, *the earlier in the SDLC that security is addressed, the less effort and cost is ultimately required to achieve the same level of security.*"[2]

Document carefully: You should record each step in the threat model process. This record provides accountability both within your organization and for your stakeholders. It also allows you to track and measure your progress. Widely sharing your threat model can provide transparency and confidence to your users in your security processes. While not essential, a threat modeling tool with diagramming capability can be of great help in this area.

As you read about the particular methodologies, think about how these questions and principles are addressed by each methodology.

[2] M. Souppaya, K. Scarfone and D. Dodson, "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities", NIST Special Publication 800-218, February 2022. https://doi.org/10.6028/NIST.SP.800-218

A Survey of Threat Threat Modeling Methodologies

In the next two chapters, we briefly describe two methodologies, with the goal of familiarizing you with a couple of different approaches. Note that there have been several survey papers published that survey the top N threat modeling methodologies (where N varies from paper to paper)[3,4]. We will not cover all the methodologies mentioned in these surveys as they either overlap heavily with the ones that we do cover, or are either incomplete or not specific enough to be ready to apply to your design. The methodologies that we will describe are Microsoft Threat Modeling and PASTA. Also, because of its early development and continued use in certain communities, we also present the DREAD threat classification. Note that DREAD addresses only a small part of the threat modeling process but was a foundational development.

While there are sometimes passionate arguments between practitioners as to which methodology to use, the key point is to make sure that you are doing **some form** of threat modeling during your design time, even if it is an informal process developed within your team. Making the choice from among the collection of established threat modeling methodologies is made more complicated in that some methodologies are proprietary and based on a company's product, so the rhetoric surrounding these methodologies is often a bit exaggerated.

We want to emphasize that it is better to adopt some systematic strategy for identifying threats than to delay and defer adopting any strategy. The earlier that you incorporate a threat modeling methodology, the more effective and painful it will be to use.

Summary

Threat modeling is a key step in producing secure software. It provides an opportunity to stop and think about security issues, hopefully before you start coding. When learning about threat modeling, we are confronted with a plethora of articles on the topic, often vague or misinformed, often motivated by marketing rather than technical understanding. This chapter presents the basic ideas behind threat modeling as a foundation for the next chapters that cover specifics of a couple of threat modeling methodologies and a threat classification system.

Exercises

1. Pick an everyday task and briefly enumerate potential threats and useful mitigations for it.
2. Research some of the other threat modeling methodologies listed in the referenced surveys to find one that best fits your needs for a current software project (or hypothetical one). What pros and cons did you see and how did those influence your preference?
3. Find a publicly available threat model on the web to read and assess. What insights did you learn, and did you think of any threats that might apply that were missed?

[3] N. Shevchenko, T.A. Chick, P. O'Riordan, T. Scanlon, C. Woody, "Threat Modeling: A Summary of Available Methods", *white paper,* CMU Software Engineering Institute, August 2018. https://resources.sei.cmu.edu/asset_files/WhitePaper/2018_019_001_524597.pdf
[4] C. Gonzalez, "Top 8 Threat Modeling Methodologies and Techniques", exabeam, February 2022. https://www.exabeam.com/information-security/threat-modeling/