Introduction to Software Security

Chapter 2.3: Microsoft Security Design Lifecycle and Threat Modeling Methodology

Elisa Heymann
elisa@cs.wisc.edu

Barton P. Miller
bart@cs.wisc.edu

Loren Kohnfelder
loren.kohnfelder@gmail.com

DRAFT — Revision 3.1, March 2024.

Objectives

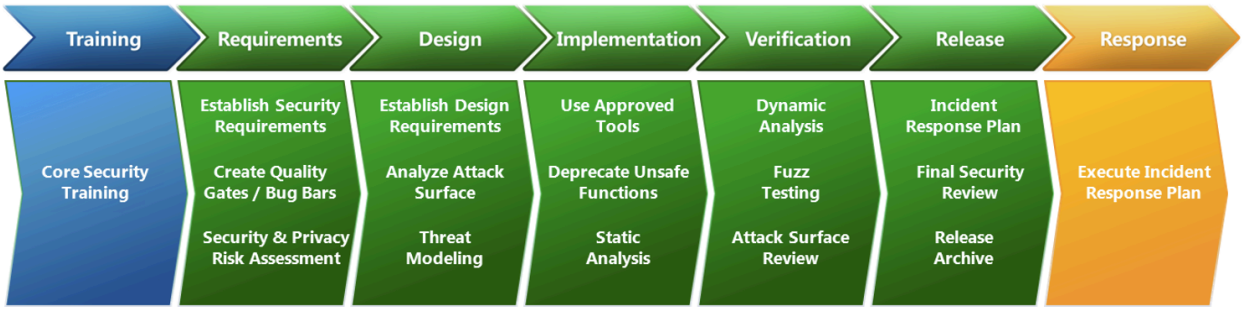
- Learn the STRIDE model of threat classification.
- Develop an understanding of the Microsoft Threat Modeling methodology.
- Understand the Microsoft Security Design Lifecycle and how it relates to STRIDE and threat modeling.

Background

To turn the tide on their early struggles with security issues plaguing their products, Microsoft introduced threat modeling methodologies as a core component of their efforts. They developed this methodology to give their designers and programmers a conceptual and practical tool that would provide a clear understanding, at design time, of the threats that a system could face. More than a decade later, Adam Shostak produced a comprehensive book[1] to describe the details of their methodology and a tool to automate its use.

In January of 2002, Bill Gates sent an email to all Microsoft employees announcing the creation of their Trustworthy Computing (TwC) initiation. One of the early outcomes of this initiative was their Security Development Lifecycle (SDL)[2]. This life cycle includes steps for requirements, design, implementation, verification, and release. Threat Modeling is a key part of the design step in the SDL.

[1] Adam Shostak, **Threat Modeling: Designing for Security**, John Wiley and Sons, Indianapolis, 2014.
[2] "Microsoft Security Development Lifecycle (SDL), Security Engineering, Microsoft Corp. https://www.microsoft.com/en-us/securityengineering/sdl/



A key enabler for promoting the Threat Modeling methodology was the development of the Threat Modeling Tool, which automated much of the process of identifying threats in a design. In 2004, to make their methodology more widely accessible and understood, Microsoft published a book describing the details and application of this methodology[3]. In 2014, Adam Shostak produced a new book trying to make the methodology and the tool more approachable[4].

In this chapter, we explain the threat classification system used by Microsoft, describe the steps performed in their threat modeling methodology, and then introduce their Threat Modeling Tool.

STRIDE

STRIDE is a threat classification model developed as part of Microsoft's Trustworthy Computing (TwC) initiative[5] (one of the authors of this book assembled the letters of the acronym for the original paper[6]). The model is a useful way to categorize common threats to a software system.

Note that STRIDE has broader applicability then the Microsoft Threat Modeling methodology and is sometimes used separately or incorporated into other threat modeling methodologies.When filing a security vulnerability report, noting the STRIDE category is a quick and easy way to characterize the issue for others to quickly see.

The six STRIDE threat categories are as follows, each to be explained in detail with examples following:

- **S**poofing of an identity
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

STRIDE aids threat modeling as a kind of checklist of types of threat to consider, though it is important to bear in mind that it may not necessarily cover every possible threat to every system. For example, when

[3] Frank Swiderski and Window Snyder, **Threat Modeling**, Microsoft Press, Redmond, WA, 2004.
[4] Adam Shostak, **Threat Modeling: Designing for Security**, John Wiley and Sons, Indianapolis, IN, 2014.
[5] Aanchal Gupta, "Celebrating 20 Years of Trustworthy Computing", Microsoft Corp., January 2022. https://www.microsoft.com/en-us/security/blog/2022/01/21/celebrating-20-years-of-trustworthy-computing/
[6] Kohnfelder, Loren; Garg, Praerit: "The threats to our products". *Microsoft Interface* (April 1, 1999).. https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx

looking at a system component block diagram or reviewing the security of a protocol, it helps to write out the six letters to make it easy to remember the corresponding threat categories, and then apply each to the analysis at hand for separate detailed consideration. For example, S: Is there an identity-spoofing threat here and what are the implications of that spoofing? T: What data might be subject to tampering and what are the implications of that tampering? We can ask similar questions for R, I, D, and E.

Some threats will involve more than one category and, generally, as long as you identify one of them and mitigate it, you often get two birds with one stone. For example, if an attacker spoofs the identity of an administrator, they also acquire the associated high level of access, which is an Elevation of Privilege. These generic categories are defined to help, not to make the job harder, but if you can identify these overlapping cases it helps later assessing the effectiveness of mitigations.

An example of the limitations of STRIDE might be in threat modeling a medical device, a particular configuration error might cause a malfunction creating a dangerous condition for the patient that could harm their physical health.

Spoofing

Spoofing of an identity is when a person or program successfully impersonates another, thereby concealing the attacker's true identity and often gaining unauthorized access in the process.

- In our vault example, an attacker impersonates Rich Customer and convinces the bank to allow them to have access to Rich Customer's money.
- Sender email address spoofing.
- Man-in-the-middle intrusion in communication such as impersonating a web service.
- Web page referrer spoofing.

Tampering

Tampering includes unauthorized modification or deletion of data or code that alters the system behavior.

- In our vault example, Evil Attacker manipulates the security cameras so the attacker can access the safe without being seen.
- Installing backdoor access (a secret, unauthorized access path). Note that this might also be thought of as an Elevation of Privilege as the backdoor provides increased access.
- Disabling security monitoring.
- Subverting authorization.
- Bypassing valid license checks.
- Altering control flow.
- Injecting malicious code into a program.
- Changing the balance amount of an account.

When tampering occurs, it can be much more pernicious if it goes unnoticed for a long period of time. In a commonly used devious form of attack, code is left behind (often termed an **implant**) that may be hidden or intentionally left inactive for a period of time, making it difficult to detect. Persistent implants can survive reboots and system updates and can be very difficult to eradicate. Similarly, subtle tampering

of important data can be devastating if undetected, compared to outright destruction of the data which is immediately noticed and soon remedied.

Repudiation

Repudiation is when a user plausibly denies performing a specific action — a fancy word for, “It wasn’t me!” Non-repudiation is the good security property of having strong evidence so, in the case of a dispute, it can be reasonably proven to a third party exactly who did what (and, ideally, when).

- In our vault example, Evil Attacker denies that they took Rich Customer’s money from the vault. They claim that they were at the movies at the time of the crime.
- A user signing a document and then claiming that the action was performed by someone who stole their credentials.
- A user claiming that they never sent an email that they actually sent.
- Denying the contents of a deleted posting online because the screenshot could be easily faked.

Information disclosure

Information disclosure is releasing information to an actor that is not explicitly authorized to have access to that information.

- Evil Attacker quietly snaps a photo of Rich Customer's transaction receipt, and then posts the receipt to Twitter, revealing their account balance.
- An attacker extracting data files from your system (exfiltration).
- Error message revealing too much information, for example, exposing the full path of the program.
- Debug error messages left in the deployed software.
- Using a weak encryption method that is easily cracked, revealing the contents.
- Disclosure of an encryption key, which could disclose the contents of gigabytes of encrypted data that is publicly accessible.

Denial of Service

Denial of service is making a resource, such as a host, server, network or application, unavailable for legitimate users. This category matches loss of availability as described within the C-I-A principles.

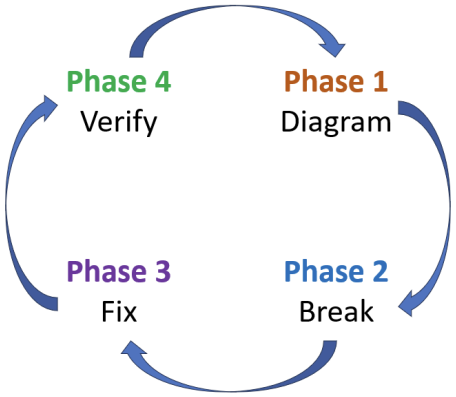
- In our vault example, preventing Rich Customer from taking their money from the vault, by putting a fake “bank closed” sign on the door.
- Saturating an online service with a large number of requests (often initiated by orchestrated access from thousands of malware-ridden home computers).
- Taking control of a large number of computers and using them to bombard a server in what is called a distributed denial of service (DDoS, pronounced “dee-doss”) attack.
- Exploiting a buffer overflow and making the system crash.
- Placing an orange cone on the hood of a self-driving vehicle, preventing it from moving.

Elevation of privilege

Elevation of privilege is getting access to more resources or functionality than a user is normally allowed by giving them more privileges than intended by the developer or system administrator.

- In our vault example, a bank employee steals the bank president's ID card so that they can have access to all the bank records.
- Vertical escalation: gaining root or administrator access.
- Horizontal escalation: accessing information associated with a different user.

The Steps in Microsoft Threat Modeling



1. Diagram[7]

The goal of this phase is to capture all the requirements and characteristics of your software system. You will need to identify the components and resources of the system, trust boundaries and the way they interact. The end product of this phase is a flow diagram of the system, such as you might draw by hand or using a threat modeling tool.

2. Break

The second phase is to apply a threat modeling technique to the information that you gathered for the diagram. This phase could be based on a manual approach to threat modeling or an automated tool-based approach such as Microsoft Threat Modeling (Chapter 2.2.1) or PASTA (Chapter 2.2.3).

3. Fix

For each of the threats that you identified in the second phase, you will need to decide how you will handle that threat. You can respond to a threat in one of three ways:

a. Modify your design so that the threat is no longer applicable. For example, you might discover that you are checking an input constraint outside the trust boundary of your server, so need to do that check inside the boundary (presumably in the host, not in the client).

[7] Microsoft actually calls this the "Design" phase. However, this overloads the use of "design", which is also used for a somewhat different purpose in the SDL steps (as we can see from the diagram at the start of this chapter).

b. Plan a implementation (coding) strategy that will account for the threat and prevent it from happening. For example, if your design has the threat of a directory traversal attack (Chapter 3.3), then you would need to ensure that you do proper checks on user input before using them to construct a path name.
c. Decide that threat is not a serious concern in your design so ignore it. For example, you might discover that some operation can occur without user authentication. However, it may be that this allowed operation is generally considered low risk and could cause minimal harm.

4. Verify

The last phase is to review the fixes from the third phase to ensure that the concerns have been addressed and all security controls are in place. As part of this phase, you will identify the information needed to update the flow diagram to include your latest design decisions. You then repeat the process starting from the first phase.

Microsoft Threat Modeling Tools

There are several tools that help the designer with the non-trivial task of performing Threat Modeling. One of the most mature and widely used is Microsoft's Threat Modeling Tool[8], which (not surprisingly) follows Microsoft's Threat Modeling Methodology.

Here is a list of readings that will introduce you to the tool and how to use it:

- https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool : (1 short page of introduction to MS Threat Modeling).
- https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool-getting-started . Basic tutorial on the MS Threat Modeling Tool.
- https://docs.microsoft.com/en-us/azure/security/azure-security-threat-modeling-tool-mitigations. A description of how to mitigate the threats that you found with the Threat Modeling Tool.

Summary

Microsoft has long been a leader in the area of secure software design. Motivated by security concerns, they developed their Security Design Lifecycle to provide a structured way to produce more secure code and to document this process. As part of SDL, they developed the Threat Modeling methodology, along with a tool, to guide their designers and developers in the secure software process. Drilling down further, they provided the STRIDE system for categorizing threats that a software system might face. Taken together, these provide a good approach to

Exercises

1. For each of the STRIDE categories, list some applicable threats for a real or theoretical software component. Suggested subjects: a game console, a home assistant (like Alexa or Siri), a smart connected kitchen appliance, or something more creative.

[8] J. Geib, B. Santos, D. Coulter, K. Bulloc, J. Howell, M. Baldwin and B. Kess, "Microsoft Threat Modeling Tool", August 2022, https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

2. For a new or existing software project, work through the steps of Microsoft Threat Modeling using the Microsoft tool. Develop the diagram, use the tool to identify potential threats, and then evaluate these to decide what kind of response is needed.
3. Try out the techniques presented in this chapter using the exercise based on Microsoft's Threat Modeling Tool: https://research.cs.wisc.edu/mist/SoftwareSecurityCourse/Exercises/TM.pdf