The 6 Most Common Threat Modeling Misconceptions

Plus tips to get your bearings straight
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The passageway to threat modeling</td>
<td>3</td>
</tr>
<tr>
<td>Secure architecture: The bedrock of threat modeling</td>
<td>4</td>
</tr>
<tr>
<td>The 5 pillars of a successful threat model</td>
<td>5</td>
</tr>
<tr>
<td>The 6 most common threat modeling misconceptions</td>
<td>8</td>
</tr>
<tr>
<td>Tips to take control of your risk management process</td>
<td>12</td>
</tr>
<tr>
<td>The light at the end of the tunnel</td>
<td>14</td>
</tr>
<tr>
<td>Appendix (Includes threat modeling vocabulary)</td>
<td>15</td>
</tr>
</tbody>
</table>
Before we set off on our journey, let’s establish the basics. There is often confusion around the definition of “threat,” which can be defined in one of two ways:

1. A potential event that will have an unwelcome consequence.
2. An individual, organization, or system from which an attack can originate.

The process of threat modeling breaks down a system’s architecture, identifying key structural elements and system assets to highlight acceptable risks and associated attacks against the system.

Key structural elements and system assets at the heart of a threat model include:

- Valuable resources to be protected
- System entry and output points
- Data and control flows
- Security mechanisms
- Potential attackers

All of these components, resulting from risk analysis, form a system’s threat model.
Threat modeling has a special relationship with secure architecture that allows the system which is set for design to account for risks and attackers. During architecture, this relationship enables the avoidance (by architecting differently) or mitigation (by adding security controls) of important security issues. It’s best to discover these security issues during architecture since they often become increasingly expensive to address when discovered during the development process.

Threat modeling promotes the idea of thinking like an attacker. It enables organizations to build software with security considerations, rather than addressing security as an afterthought. Despite the special relationship it has with secure architecture, threat modeling is also very useful as an input to activities that take place during other stages of the software development life cycle (SDLC), including secure requirements identification, secure code review, and penetration testing. Threat modeling informs these activities and offers invaluable insight into the methods attackers could use to harm the system.
Threat modeling identifies risks and flaws affecting a system. A thorough analysis of the software architecture, business context, and other artifacts (i.e. functional specifications, user documentation) allows practitioners of the threat modeling process to discover important aspects of the system—security-related or not—and synthesize an understanding of the system that may not yet exist within the organization. Conceptually, threat modeling is a simple process. There are five primary activities that need to be performed when creating or updating a threat model:

1. Define scope and depth of analysis.

The first step in shaping a threat model is to define the scope and depth. Once a reasonable scope is determined with stakeholders, it needs to be broken down in terms of individual development teams so they are able to develop a model of the software.

2. Gain an understanding of what is being threat modeled.

Diagramming the system structure is critical. This creates a solid visual understanding of the system. The component diagram below illustrates the major components of the system and the interactions between those components.

Components identified in this diagram include the application server, a legacy system, data warehouse, thick client, primary database server, and a single sign-on server.
3. Model the attack possibilities.

Identify the assets, security controls, and threat agents. Once identified, mark their locations as the diagram below illustrates.

Now that you have created a threat model, it’s time to interpret it. This interpretation will allow you to generate a traceability matrix to drive other security activities (including secure code reviews and penetration tests) in a risk-based fashion.
4. Interpret the threat model.

Produce a list of potential attacks by asking questions like:

- Is there any path where a threat agent can reach an asset without going through a control?
- For any security control along each of those paths:
  - What must a threat agent do to defeat the control?
  - Can a threat agent defeat the control currently?

5. Create a traceability matrix to record missing or weak controls.

A traceability matrix examines a threat agent. This agent may attempt to compromise an asset by conducting an attack somewhere along the attack surface. Consider what the attack goal could be and how it could impact the target. To mitigate an attack, a control is put in place to establish an acceptable level of risk. Repeat this process for all threat agent/asset combinations.

To create a traceability matrix, consider the threat agents and follow the control path. If you reach the asset without going through a control, you may have discovered a potential attack. If you went through a control, consider whether the control would halt a threat agent and whether that threat agent may have methods to bypass the security control. This is where a knowledge of actual attacks is essential.

<table>
<thead>
<tr>
<th>THREAT AGENT</th>
<th>ASSET</th>
<th>ATTACK</th>
<th>ATTACK SURFACE</th>
<th>ATTACK GOAL</th>
<th>IMPACT</th>
<th>CONTROL (MITIGATION)</th>
</tr>
</thead>
</table>

Although the threat modeling process is simple, it takes time and repetition to become proficient at it. A highly detailed traceability matrix can ensure a thorough and complete threat model. Consider the following additional column categories to achieve this holistic approach:

- Skill
- Motivation
- Attack vector
- Discoverability
- Probability

With experience, you’ll be able to develop a simplified traceability matrix. This allows you to properly track threat information through the requirements, design, and testing phases of the SDLC without as much documentation.
Now that we've established what a threat model is and how to create one, let's examine six common misconceptions about threat modeling and how to set the record straight.

**MISCONCEPTION #1**

*We already conduct penetration tests and code reviews. We’re covered.*

Penetration testing and secure code review can uncover a variety of security issues in an application; however, there are gaps that simply can't be found with these analysis techniques. To identify these gaps, we must first understand the difference between a bug and a flaw.

### The defect universe: bugs vs. flaws

Many software practitioners commonly refer to software defects as bugs. It’s important to distinguish between design-level and implementation-level issues since they involve very different techniques and levels of effort to detect and remediate.

A **bug** is an implementation-level software problem or defect in code.

A **flaw** is a design-level or architectural defect in specifications or software that is more difficult to resolve once the software is partially or completely implemented.

<table>
<thead>
<tr>
<th>Security Defects Uncovered in Practice</th>
<th>IMPLEMENTATION FLAWS</th>
<th>ARCHITECTURAL BUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50%</td>
<td>50%</td>
</tr>
</tbody>
</table>

Many software practitioners commonly refer to software defects as bugs. It’s important to distinguish between design-level and implementation-level issues since they involve very different techniques and levels of effort to detect and remediate.
There is approximately a 50/50 distribution between bugs and flaws. Here are a few examples of each:

**IMPLEMENTATION BUGS:**
- SQL injection
- XML/XPath/* injection
- Cross-site scripting
- Hardcoded passwords
- Unsafe system calls
- Predictable identifiers
- Hardcoding secrets in code

**DESIGN FLAWS:**
- Misuse of cryptography
- Broad trust between components
- Client-side trust
- Broken or illogical access control (RBAC over tiers)
- Missing defense for replay attacks
- Insecure auditing

Penetration testing and secure code review are two activities that are effective for finding bugs. Flaws, on the other hand, are usually uncovered by assessment activities that demand a thorough understanding of secure design principles and business context of the system being reviewed. Both of these are also typically very difficult to automate. When performing penetration testing and secure code review, recognize that you’re only addressing half of the software security defects affecting your software.

**MISCONCEPTION #2**

**We already deployed our system. There’s no reason to conduct a threat model.**

It’s a common misconception that threat modeling is only necessary for new applications. If a threat model doesn’t exist for an application that has been deployed in production:

- You have no information about your production security posture.
- You have no information about deployed defenses and attack surfaces.
- Future deployments can’t defend against existing limitations and vulnerabilities.
- Future deployment can’t take advantage of existing defenses.

In other words, you’re conducting security blindly, if at all. Understanding the issues that are currently deployed influences your future security architecture strategy. Monitoring weaknesses allows your team to react faster and more effectively. Threat modeling is essential to any secure software initiative. Without an understanding of the potential threats that an application faces, it is impossible to ensure that the most pertinent risks are being properly addressed.
MISCONCEPTION #3

We carried out a threat model when the software was built. There’s no reason to do it again.

It is important to know if anything changed in the system since the last threat model. For instance, has a feature been added, removed, or changed? Even if nothing has changed in your software, it is possible, and quite likely, that something has changed in the software you use (frameworks, operating systems, and internal or open source libraries). Even if nothing has changed, new attack techniques may affect your threat model.

MISCONCEPTION #4

We’ve considered threat modeling and feel that it is way too complicated.

Many are intimidated by the idea of undertaking the threat modeling process. At first glance, it can seem daunting. However, if you break up the tasks into the five workable steps, performing a threat model on a simple web application, and even a complex system architecture, becomes systematic. The key is to start off with the basics. Create threat models for simple web applications. Once you’re comfortable with this process, move to more complex systems such as mobile platforms, embedded software, and cloud-based technologies. The good news is that threat modeling is architecture agnostic. However, to be most effective, the individual or team conducting these threat models should have a thorough knowledge of the architecture they’re working with, along with its associated threat landscape.

It is also important to note that a threat model is never complete. It’s a living document that evolves with the application, across releases, and should be mindful of changes in the external business, technical, and threat landscapes.

MISCONCEPTION #5

We don’t have software security experts, so we can’t do threat modeling.

Threat modeling is a lot like cooking. Chefs aren’t the only people around who can cook. At the same time, you probably won’t be preparing an elegant feast on your first day in the kitchen. You need to learn to boil water
first. Threat modeling, like cooking, takes practice. While threat modeling takes time and repetition to become proficient, there are also options available for firms without software security teams or in-house experts. At Cigital, we model threats specific to your business and shine the light on the types of attacks you are most likely to face.

MISCONCEPTION #6

We’re threat modeling at all the right times, so we don’t need additional security activities.

Threat modeling identifies security weaknesses in software. If a weakness is exploitable by a threat agent, it’s a vulnerability. While threat modeling identifies weaknesses, it doesn't evaluate exploitability. Thus, the weaknesses found through threat modeling may or may not be actual vulnerabilities. Subsequent activities such as penetration testing and code reviews can evaluate this exploitability of the weaknesses found during threat modeling.

To give you an example of a weakness, consider an application that doesn’t perform output encoding of untrusted data. The application may or may not contain actual injection vulnerabilities, depending on the quality of input validation controls. Threat modeling simply identifies the missing output encoding control as a weakness. Ensuing activities then determine whether the input validation controls are sufficient to protect against the types of injection vulnerabilities relevant to the application.

Threat modeling can help drive:

- Secure code review which aims to find implementation errors that are relevant to system architecture.
- Penetration testing which verifies the resilience of the system against relevant attacks.
- Security requirement identification which specifies the software’s behavior in response to potential risk and threat agents.
4 ways to use threat modeling to avoid sink holes in your risk management process

1. Think beyond canned attacks.

You should definitely consider standard canned sets of attacks, but you should also be aware that some of these attacks may not actually pose a risk to your system. Standard attacks are useful to model risk against your software, and threat modeling allows you to identify the attacks that are unique to how your system is built. One such example is an internal authorized user attacking an insecure interface used by partner systems.

It’s important to approach threat modeling as a living resource that evolves with the associated application. The real power of threat modeling is that it makes you think about your system’s specific characteristics, helping you to gain visibility around weaknesses that pose significant impact to your entire organization.
2. Identify where threat agents exist relative to the architecture.

“If you know your enemy and you know yourself, you need not fear the results of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.”

-Sun Tzu, Art of War

This quote may have been written to discuss military strategy, but it also holds true when discussing threat potential in applications. Threat modeling helps you identify and model relevant threats which aren't obvious. Modeling the location of the threat agent, their motivation, skills, and capabilities is critical to identify where potential attackers are positioned in relation to your system's architecture.

Once these threats have been listed, the potential risk associated with an action can be weighed against other threats; allowing you to focus on the actions deemed most critical to your organization. This both ensures that time is invested in developing and implementing controls to alleviate the highest amount of risk, and that those controls that are in place will be most effective against the types of threats encountered.

3. Identify top-N lists, attackers, and doomsday scenarios

Before starting any assessment activity, know the top risks and attacks your application faces. Many organizations prescribe a set of common web application risks, such as the OWASP Top Ten. However, these are rarely an ideal match with the issues that your applications and overall organization face.

Doomsday scenarios express extreme situations that threaten your organization. Build your software to account for these scenarios and mitigate the risks. Regularly create and update your threat models to keep your frameworks ahead of top attacks, doomsday scenarios, and any internal or external attackers relevant to your applications.

4. Identify components that need additional protection

Highlighting assets, threat agents, and controls through threat modeling allows you to determine which components are likely targets for attackers and offers insight into how they could be attacked. This intelligence effectively arms you with the knowledge to address weaknesses by adding additional security controls or completely avoiding the risks by removing unrequired functionality.
The light at the end of the tunnel

Finding implementation bugs is important, but it’s not enough to secure your applications. Threat modeling is a systematic way to discover design-level flaws in the architecture of new and existing applications. Think like the bad guys to build security in from the beginning and maintain a solid application security initiative in the future.
Threat modeling glossary

Asset – A valuable resource that needs to be protected by an organization or system. In software security, an asset can be either a data asset or a functional asset.

Attack – A combination of technology, technique, and knowledge that can be applied to a vulnerability and cause a security compromise.

Attack surface – The collection of interface points that may be attacked and potentially penetrated by a threat agent (i.e., attacker) to obtain unauthorized access to an asset. An attack surface contains the sum of entry points, but we are most interested in those that are exploitable (i.e., vulnerable) to obtain such access via attack vectors (i.e., paths to exploit the vulnerabilities).

Control – The mechanism currently in place within a piece of software that protects an asset.

Mitigation - Reducing the severity or impact of an issue or vulnerability discovered in a security test, often through compensating controls such as log monitoring, application firewalls, and temporarily removing access or functionality.

Risk – (1) The probability that an undesirable event will actually occur. (2) A measure of the potential impact given an undesirable event occurring.

Threat agent – Anything (person, code, etc.) capable of acting against an asset in a manner that can result in harm to the system or its owner.

Threat model – A type of security analysis that documents threat agents, skill, motivation, attack surface, attack vectors, discoverability, probability, impact, and mitigation information for a system to facilitate risk analysis. Identified risks are used as input to change design and to improve downstream security activities like penetration testing and secure code review.

Traceability matrix – The mapping of factors including threat agents, skill, motivation, attack surface, attack vectors, discoverability, probability, impact, and mitigation, providing a more formal correlation of these attributes to enhance the outcome of the threat modeling exercise.

Written by Meera Subbarao.
Synopsys offers the most comprehensive portfolio of solutions to build security and quality into your software development and supply chain. We go beyond traditional testing services to help our clients identify, remediate, and prevent vulnerabilities in the applications that power their business.

Our holistic approach to application security offers a balance of managed and professional services and products tailored to fit your specific needs. We don’t stop when the test is over. Our experts also provide remediation guidance, program design services, and training that empower you to build and maintain secure applications.

For more information go to www.synopsys.com/software.