Six Steps to Effective Threat Modeling
Build trust in your software with threat modeling

Every organization relies on software to innovate and deliver value to customers, so in effect, every business is a software business. The software that organizations develop and use to run their business operations allows them to work faster and more efficiently, but if that software is not developed and deployed securely, it can introduce a considerable amount of risk. And that means software risk is business risk, so it needs to be prioritized and managed proactively.

It's tempting to think that the solution is simply newer, better, or faster security tools, but securing software requires more than just tools. Threat modeling is an important part of a holistic approach to application security (AppSec). It aligns people, processes, and technologies to optimize security in the service of an organization's unique policies and business objectives.

What is threat modeling?

**Threat modeling** is a structured process that

- Identifies security requirements
- Pinpoints security threats and potential vulnerabilities
- Quantifies threat and vulnerability criticality
- Prioritizes remediation methods

Threat modeling is a set of techniques that identifies risks by assessing system architecture and expected behavior. Threat modeling analyzes software design by comparing design views against threat agents to identify security weaknesses. Working with the software design means that the analysis can be conducted without having to delve into the code. Design weaknesses that can be identified during the analysis include missing security controls, weak or inappropriate security controls, and potential vulnerabilities.

Threat modeling breaks down a system's architecture, identifying key structural elements and system assets and documenting risk with enough detail to allow the business to make informed decisions about that risk.

Threat modeling creates artifacts including

- An abstraction of the system
- Profiles of potential attackers, including their goals and methods
- A catalog of threats that could arise

*Threat modeling promotes the idea of thinking like an attacker. It enables organizations to build security considerations into software from the start, rather than addressing security as an afterthought.*

Secure architecture: The bedrock of threat modeling

Threat modeling allows the system to account for risks and attackers. When performed during the architecture design phase, threat modeling enables organizations to avoid or mitigate security issues by changing the design or adding security controls. As part of a holistic AppSec program, addressing these security issues during architecture design saves the escalating expenses of addressing them when they’re discovered later in the development process.

Threat modeling promotes the idea of thinking like an attacker. It enables organizations to build security into software from the start, rather than addressing security as an afterthought. In addition to its use during the architecture design phase, threat modeling is also very useful during other stages of the software development life cycle (SDLC), including secure requirements identification, security test planning, secure code review, and penetration testing. Threat modeling informs these activities and offers invaluable insight into the methods attackers could use to harm a system.

The six primary activities of a threat model

Threat modeling identifies risks and flaws affecting a system. A thorough analysis of the system architecture, business context, and artifacts such as functional specifications and user documentation allows threat analysts to discover important aspects of the system—security-related or not—and synthesize an understanding that may not yet exist within the organization. Conceptually, threat modeling is a simple process. There are six primary activities that constitute 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. This means setting parameters and boundaries for which components should be included in the review. This phase provides an understanding of the overall effort and expected delivery timeline for the assessment.

2. Gather data
Next, the threat modeling team reads and reviews artifacts provided by the development team. These documents include, but are not limited to, architectural descriptions, software design documents, API specifications, end user documentation, source code, and related artifacts. This preliminary analysis is used to identify high-risk areas in the platform, which will form the basis for all future activities in the threat model.

As part of the data gathering phase, the team will select key members of the application and business teams to interview. The interviews will focus on providing a more in-depth understanding of the application to assess areas of risk. These interviews provide the threat analysts with key knowledge about the design and implementation details of the application.

3. Model the system
The threat analysts can now build a simplified diagram of all the in-scope, adjacent, or connecting components (some which may be out of scope), as well as the connections and their protocols.

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.
Diagramming the system structure provides a solid visual understanding of the system. The component diagram on the previous page illustrates the major components of a system and the interactions between those components.

4. Perform threat analysis

When the component diagram is complete, the threat analysts add assets, controls, threat agents, and trust zones to complete the threat model diagram. This threat analysis diagram should enumerate all possible realistic threats to the system, focusing on data, dataflow, platform, and operational security. In addition, it should identify weak or missing controls that may lead to a successful compromise. Once identified, they mark the location of each threat, as the diagram below illustrates.

**Assets**
- A01: Primary DB credentials
- A02: Sensitive application data
- A03: Session tickets
- A04: Application username and password
- A05: Application event tables
- A06: Report data
- A07: Coyote Data REST credentials
- A08: Reports-4-U SFTP credentials
- A09: VisiWidget API
- A10: Active Directory username and password
- A11: Special Flash functionality
- A12: Client certificate for Thick-Client
- A13: Application log file and event data
- A14: Cached reporting data

**Threat Agents**
- TA01: Unauthorized external user
- TA02: Authorized external application user
- TA03: Unauthorized internal user
- TA04: Authorized internal application user
- TA05: Authorized internal system admin
- TA06: Authorized internal DB admin
- TA07: Authorized developer

**Controls**
- C01: Primary DB authentication and schema
- C02: External application user authentication
- C03: One-way SSL
- C04: Mutual SSL
- C05: IP address restrictions
- C06: Token validation (for Reports-4-U)
- C07: Salted SHA-256 hash of user passwords
- C08: Internal user Flash SWF file
- C09: 90-day limit for report data
5. Perform risk analysis

Now that the model has been completed, it is imperative to review the data flow and connections component by component to enumerate a list of realistic threat scenarios. Creating a traceability matrix is a way to record missing or weak controls, so a plan to rank and mitigate them can be defined.

To create a traceability matrix, consider the threat agents and follow the control path. If an asset can be reached without going through a control, there may be a potential risk or design flaw. If an asset can be 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>
<tbody>
<tr>
<td>Skill</td>
<td>Motivation</td>
<td>Attack vector</td>
<td>Discoverability</td>
<td>Probability</td>
<td></td>
<td></td>
</tr>
</tbody>
</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. Completing a thorough traceability matrix requires you to think not like a developer, but like an attacker. One way to do this is to break threats into categories like these.

By thinking through each of these threat agent categories, and tracing them through your design to see where they might slip through a control, you can build a traceability matrix to properly track threat information through the requirements, design, and testing phases of the SDLC.

6. Create reporting

The final stage of the threat model process is to document all models, system descriptions, potential risks, actual risks and their associated likelihood and impact. A consulting service like the ones Synopsys offers will provide this information in a PDF report.

The three most common threat modeling misconceptions

There are three major misconceptions about threat modeling as a security process.

- Threat modeling is limited to the design stage.
- Threat modeling optional exercise for which penetration testing or code review can substitute.
- Threat modeling is too complicated to attempt at all.

Let's dispel these misconceptions.

There are good reasons to conduct a threat model after deployment. Threat modeling does not need to be limited to the design stage of development. Even if the software has already been deployed, threat modeling will help increase security. Understanding security issues in a current deployment can help determine the future security architecture strategy. Monitoring security weaknesses in a current deployment also allows for faster and more effective remediation when threats arise. Without investigating and identifying the potential threats an application faces, it's impossible to ensure that all risks have been addressed.
Penetration testing and code reviews can't substitute for threat modeling. Penetration testing and secure code review are effective for finding bugs in code. However, security assessments such as threat modeling that demand a thorough understanding of secure design principles and the business context of the system being reviewed are better at uncovering design flaws. This means that security assessments are very difficult to automate. Although penetration testing and secure code review are important security activities, it's important to keep in mind that they can only discover bugs in code—they can't expose design flaws or threats that are particular to a business context.

Threat modeling isn't that complicated. Many developers are intimidated by the idea of threat modeling. At first glance, it can seem daunting; however, if tasks are broken up into workable steps, performing a threat model on a simple web application—or even a complex architecture—becomes systematic. The key is to start with basic best practices like the ones we've outlined here.

“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, “The Art of War”

The light at the end of the tunnel

Finding implementation bugs is important, but it’s not enough to secure all 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 circumvent them: Build security in from the beginning and maintain a solid application security initiative in the future.
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, or knowledge that can be applied to a vulnerability and cause a security compromise.

**Attack surface**: The collection of interface points that can 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 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.
Synopsys Software Integrity Group provides integrated solutions that transform the way development teams build and deliver software, accelerating innovation while addressing business risk. Our industry-leading portfolio of software security products and services is the most comprehensive in the world and interoperates with third-party and open source tools, allowing organizations to leverage existing investments to build the security program that’s best for them. Only Synopsys offers everything you need to build trust in your software.

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

Synopsys, Inc.
690 E Middlefield Road
Mountain View, CA 94043 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com