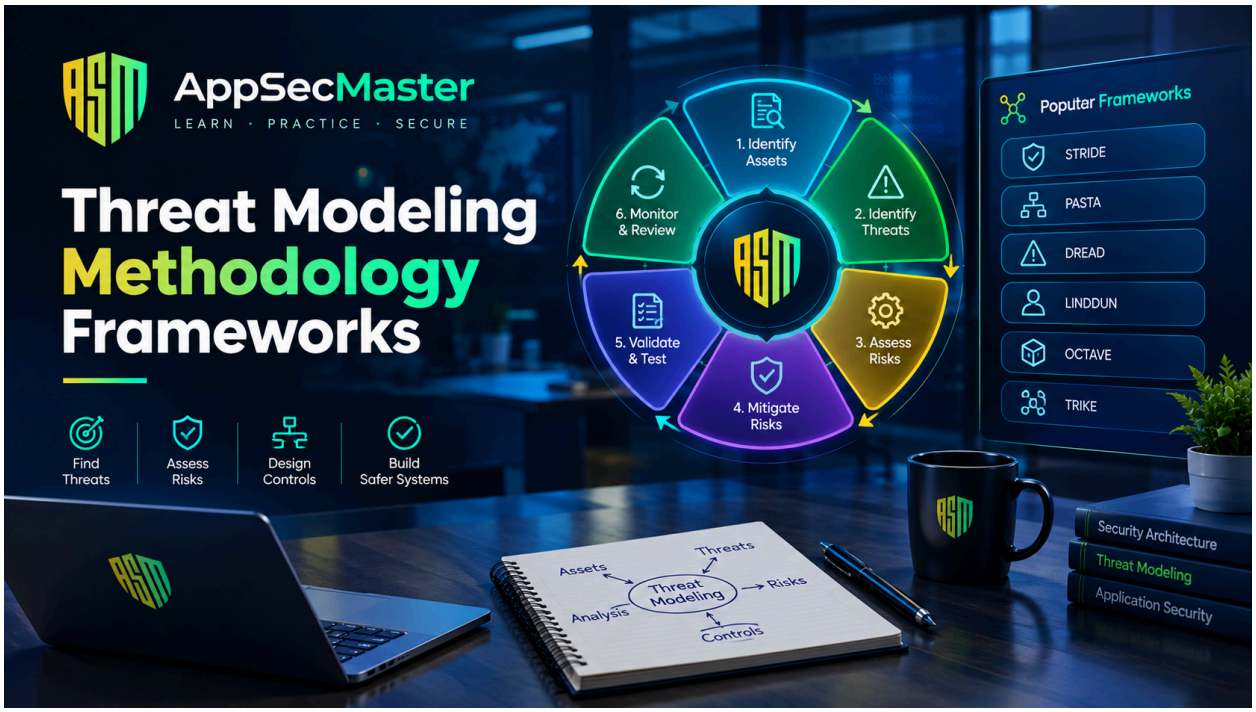
Threat Modeling Methodology Frameworks Compared

A threat modeling methodology is a repeatable process for finding and prioritizing design level security risks. STRIDE is the best general purpose starting point for application teams, attack trees suit goaldriven and business logic analysis, PASTA fits riskcentric programs and LINDDUN targets privacy. Choose by your goal, your team's maturity and the time available, then embed the chosen method in your secure development lifecycle at design review time.

Ask five security engineers to describe the threat modeling methodology and you will hear five different answers. That is not because the field is confused. It is because threat modeling is a family of techniques that answer related but different questions and different organizations need different answers at different moments.If you are brand new to the topic, start with our primer on threat modeling fundamentals; if you prefer to learn by doing, the companion resource on threat modeling practice gives you exercises to apply the methods described here.



What a Threat Modeling Methodology Actually Contains

Frameworks are often confused with methodologies. A framework or technique, such as STRIDE, helps you generate ideas about what might go wrong. A methodology is the full process wrapped around it. A complete methodology answers six questions

1. Scope what system or change is being analyzed and at what level of detail?
2. Representation: how is the system modeled, for example with data flow diagrams or sequence diagrams?
3. Threat discovery: which technique finds threats?
4. Risk evaluation: how are threats ranked?
5. Response: how are mitigation decisions made, recorded and assigned?
6. Validation: how do you check that the model was good and the controls work?

When a team says its threat modeling did not work, the breakdown is usually in one of the last three areas. They found threats but never ranked them, ranked them but never assigned owners, or assigned owners but never verified the fixes. A good methodology is as much about workflow as about security knowledge.

The Main Frameworks and Techniques

STRIDE

STRIDE was developed at Microsoft and remains the most widely taught technique. It categorizes threats as Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege. Each category corresponds to a violated security property authentication, integrity, nonrepudiation, confidentiality, availability and authorization.STRIDE works on a data flow diagram. For each element and flow, you ask which categories apply. Its strengths are coverage and simplicity. Its weakness is that it can generate long lists of generic threats and it is less natural for abuse of legitimate functionality. Our article on web application security threats shows how the STRIDE categories map onto the most common application flaws.

DREAD

DREAD is a rating approach rather than a discovery technique. It scores Damage potential, Reproducibility, Exploitability, Affected users and Discoverability. Averaging gives a single figure. It is simple and teaches reasoning about risk factors, but the scores are subjective, "discoverability" is controversial because it encourages security through obscurity and the same threat can receive different scores from different reviewers. Many mature programs have replaced it with qualitative ratings or industry scoring standards, but it remains common in training because it is easy to understand.

Attack Trees

Attack trees, popularized by Bruce Schneier, represent an attacker's goal as the root of a tree. Child nodes are subgoals or methods, combined with OR (alternatives) and AND (all required). You can annotate leaves with cost, skill or probability and then calculate the cheapest path for

an attacker. Trees excel at business logic abuse, fraud and multistage attacks. They are less suited to systematically covering every component of a large system.

PASTA

The Process for Attack Simulation and Threat Analysis is a sevenstage, riskcentric methodology. It starts with business objectives and compliance requirements, moves through technical scope, application decomposition, threat analysis, vulnerability analysis and attack modeling and ends with risk and impact analysis. PASTA is heavyweight, but it connects technical threats to business impact in a way that executives and risk managers value. It suits regulated or high value systems more than quick feature reviews.

LINDDUN

LINDDUN is a privacy focused technique whose name lists its categories Linking, Identifying, Nonrepudiation, Detecting, Data disclosure, Unawareness and Noncompliance. Where STRIDE asks how a system could be attacked, LINDDUN asks how it could harm the privacy of the people whose data it handles. It is a strong complement to STRIDE for systems that process personal data.

VAST

Visual, Agile and Simple Threat modeling was designed to scale across large organizations and integrate with agile development. It distinguishes application threat models from operational ones and relies on tooling and automation. It appears mostly in commercial platforms.

OWASP Threat Modeling Guidance

The OWASP community publishes practical material on threat modeling, including the Threat Modeling Cookbook and the Threat Dragon tool. The guidance is technique agnostic; it encourages teams to adopt the four question approach and to choose methods that fit their needs. It is an excellent free reference when you are writing your own internal process and it pairs well with the OWASP Top 10 families for deriving typical threat categories.

Trike and Other Approaches

Trike frames threat modeling as an auditstyle, requirements driven exercise with an explicit actor and asset matrix. Other approaches exist, including security cards and killchain based models. These are rarely a team's first choice but can add useful perspective.Add the device as an untrusted environment under the attacker's control, as described in our reference on mobile app vulnerabilities.

How to Choose a Methodology

No single method wins everywhere. Use the following decision factors.

Your goal. If you want to review a new feature quickly, use STRIDE on a small diagram. If you are protecting a payment flow from fraud, add attack trees. If you need to justify security investment to leadership, PASTA's business framing helps. If personal data is central, add LINDDUN.Your team's maturity. Beginners benefit from structure and a fixed checklist. Experienced teams can use lighter, conversation driven sessions and rely on their own judgment, using a technique only as a prompt.

Time and cadence. A thirty minute review on every significant design change beats a two week model done once a year. Pick a method that can be repeated at the pace your delivery team works.Tooling and integration. Consider where findings will live. If the answer is your issue tracker, choose a method and a tool that can export into it.Regulatory context. Some sectors require documented risk assessments with traceability. In those cases a more formal methodology pays for itself.

A practical default for most application teams is this combination data flow diagrams for representation, STRIDE for discovery, a simple likelihood and impact grid for ranking, attack trees for the few flows involving money or sensitive actions and a short written record of decisions.Consider prompt injection, data leakage through context, excessive tool permissions and unsafe output handling, which can reopen classic flaws such as crosssite scripting (see our guide on how to prevent XSS attacks).

Embedding Threat Modeling in the Secure Software Development Lifecycle

A methodology that lives outside the delivery process is ignored. The goal of a secure software development lifecycle is to place the right security activity at the stage where it is cheapest and most effective. Threat modeling belongs at design time, but it also touches other phases.

SDLC phase	Threat modeling role	Typical output
Requirements	Identify assets, actors and abuse cases; derive security requirements	Security requirements list
Design	Create and review the model; perform the security design review	Threat register, design changes
Implementation	Developers use the model to guide secure coding and code review	Checklists, secure coding tasks

Testing	Turn threats into test cases and penetration test scope	Abuse case tests, test plans
Release	Confirm that required mitigations are in place	Release gate evidence
Operations	Update the model after incidents and architecture changes	Revised model, detection ideas

Security requirements analysis

The bridge between threats and code is the security requirement. For each high priority threat, write a requirement that is specific, testable and traceable. For example "The system shall reject any request in which the object identifier does not belong to the authenticated user's organization and shall log the rejection." That statement is something a developer can build and a tester can verify.

Security design review and architecture review



A security design review looks at proposed designs before they are built, while an architecture review looks at the structure of a system, often as it exists. Threat modeling is the engine of both. In a design review, the model is built or updated for the change under discussion. In an architecture review of an existing system, the model reveals longstanding assumptions, such as flat internal networks or shared credentials, that individual code reviews never surface.

Triggers for modeling

Do not model everything. Define triggers that start a session, such as a new externally facing feature, a change to authentication or authorization, handling of a new class of sensitive data, a new third party integration, a significant architecture change, or an incident with designlevel root causes. Small changes can be handled with a short checklist. Our overview of web security best practices provides the baseline controls to compare against.

Risk Based Prioritization

Threat modeling produces more findings than any team can fix. A risk based security assessment ranks them so that effort goes where it matters. A workable approach has four steps.

1. Define impact in business terms. Loss of customer data, fraud loss, downtime, regulatory exposure and reputational harm should be mapped to simple categories, such as low, medium, high and critical.
2. Estimate likelihood using actors. Consider who could attack, how motivated they are and how much skill and access they need. Internetexposed flows with no authentication score higher than internal administrative paths.
3. Combine into a rating. A two dimensional grid is enough for most teams. Where you need finer ranking, use a recognized scoring approach.
4. Decide the response. Mitigate, eliminate, transfer or accept, with an owner and a date.

A short example shows the logic. A missing authorization check on an invoice endpoint has high impact (customer financial data) and high likelihood (any loggedin user can try it), so it is critical. A theoretical timing side channel on an internal batch job has low impact and low likelihood, so it is accepted and documented. Showing this reasoning in writing makes prioritization defensible.

Attack Surface Analysis as a Companion Activity

Attack surface analysis complements threat modeling. Where a model reasons about design, an attack surface review enumerates what is actually reachable public endpoints, open ports, exposed administrative consoles, third party scripts, file upload points, message queues and cloud storage permissions. You can build an initial inventory with scanning and discovery tools. For example, the commands in our Nmap cheat sheet help you identify exposed services and an intercepting proxy such as the one explained in our guide to what Burp Suite is helps map web endpoints and parameters.Feed the results back into the model. Surprises, such as an unlisted test endpoint or a forgotten legacy service, are exactly the kind of gap that makes the model more accurate.

Tools That Support the Methodology

Tools are helpers, not substitutes for thinking. Common options include

- Microsoft Threat Modeling Tool free, Windowsbased, built around DFDs and STRIDE with a rulebased threat generator and customizable templates
- OWASP Threat Dragon open source and crossplatform, with a web and desktop experience and issue tracker friendly threat tracking
- Threat modeling as code libraries approaches that describe the system in text or code and generate diagrams and threat lists, convenient for version control and automation
- Commercial platforms products that integrate with architecture repositories and ticketing systems and offer questionnaire driven models
- Plain diagramming tools and spreadsheets perfectly adequate for many teams

When evaluating a tool, ask whether it fits your workflow, whether the output can be reviewed in the same system developers already use and whether the threat library can be customized to your technology stack. Be wary of any tool that implies a model is complete because it generates many threats. Completeness comes from analysis.

Adapting the Methodology to Modern Systems

Methodologies must keep pace with how software is built.Cloud and microservices. Increase the emphasis on identity, service to trust, secrets handling and permissions. Trust boundaries now include accounts, networks, namespaces and pipelines.

APIs. Model each API as a set of operations and review authorization per operation and per object, because objectlevel and functionlevel flaws dominate realworld incidents.AIgenerated and AIassisted code. Generated code can introduce unnoticed assumptions and insecure defaults. Treat the generated components as a distinct source of risk in the model and align review practices with guidance on AI code security best practices.

Measuring Whether the Methodology Works

You cannot improve what you do not measure. Useful indicators include the percentage of qualifying changes that received a model, the time from model completion to the first mitigation being scheduled, the proportion of high risk threats with a verified fix, the number of designlevel issues found in later penetration tests (which should fall over time) and developer feedback on usefulness. Avoid counting the number of threats found, since it rewards volume rather than quality.

Conclusion

A threat modeling methodology is more than a mnemonic. It is a repeatable process that connects a clear picture of the system to a ranked list of risks and a set of verified decisions.

STRIDE gives most application teams a dependable way to discover threats, attack trees sharpen the analysis of fraud and chained attacks, PASTA links technical risk to business context and LINDDUN protects privacy. Whichever mix you choose, embed it where design decisions are made, write requirements that can be tested, rank by risk and measure outcomes. Start with one lightweight, repeatable process, improve it with each session and let evidence from real findings guide how it evolves.Developers who want to deepen their technical judgment can apply what they learn on the labs and challenges at AppSecMaster and pair threat model findings with disciplined secure code review to confirm that mitigations exist in the code.

Frequently Asked Questions (FAQs)

What is the most widely used threat modeling methodology?

STRIDE applied to data flow diagrams is the most widely taught and used technique for application teams. It is usually combined with a simple risk rating and a written record of mitigation decisions, forming a complete lightweight methodology.

What is the difference between a threat modeling framework and a methodology?

A framework or technique such as STRIDE helps discover threats. A methodology is the end to end process that includes scoping, modeling, discovering, ranking, deciding on responses and validating results. Most successful programs define the whole process, not just the discovery technique.

Should I use STRIDE or PASTA?

Use STRIDE for fast, designlevel reviews of features and services. Use PASTA when you need to tie technical risk to business impact for a regulated or highvalue system and can afford a longer analysis. They are not mutually exclusive; some teams use PASTA to frame the effort and STRIDE to discover threats.

When in the development lifecycle should threat modeling happen?

As early as there is a design to analyze, ideally during requirements and design and again when architecture or data handling changes significantly. Updating the model after incidents and major releases keeps it accurate.

Do I need a special tool to do threat modeling?

No. A whiteboard and a spreadsheet are enough to start. Tools such as the Microsoft Threat Modeling Tool and OWASP Threat Dragon add structure, templates and tracking, which become more valuable as the number of models grows.