ThreatModeler

PURDUE UNIVERSITY STUDY

A novel approach to teaching secure-by-design PRINCIPLES

Building cybersecurity skills through hands-on learning.

ThreatModeler × Purdue University

INTRODUCTION

Teaching threat modeling, rethought

Amid rising cyberattack risk and growing demand for skilled cybersecurity engineers, universities are central to teaching foundational skills. Threat modeling is one of them, and it is due for a rethink in how it is taught.

In threat modeling, an organization's attack surface is mapped to identify risks and put mitigations and controls in place. The practice enables a secure-by-design approach: proactive mitigation early in development, rather than reacting to attacks after deployment.

Teaching threat modeling in a software engineering course builds both secure software and a security mindset. Yet many academic approaches rely on component-level analysis rather than a systems-level view, and on tools that are outdated or unsuitable. Traditional methods have been labor-intensive manual processes that demand substantial resources, and their outputs often lose value soon after delivery.

ThreatModeler® accelerates threat modeling with an automated, enterprise-grade platform. Its intuitive interface, expansive threat library, and guided model creation have been shown to increase security-architect output tenfold and let non-security professionals build models and address threats. Recently, a semester-long software engineering project at Purdue University featured the ThreatModeler platform, helping students better identify and mitigate security threats in their applications.

> **"The project showed how industry-grade tools can prepare future engineers for real-world security demands, bridging educational gaps and fostering a new generation of security-conscious architects and engineers."**
>
> **Purdue University study**

THE STATUS QUO

Challenges with traditional methods

At Purdue's Elmore Family School of Electrical and Computer Engineering, Dr. James Davis teaches an undergraduate software engineering course of 75 to 150 third- and fourth-year students each semester. His method is problem- and project-based, learn by doing, anchored by a semester-long project in which students build tools that help engineers reuse software.

In earlier semesters, threat modeling was taught with the conventional STRIDE framework, where students built models by hand in general-purpose diagramming tools. STRIDE helped, but the manual work was labor-intensive and lacked the interoperability modern architectures need, which slowed students and made accurate models at scale hard to reach. STRIDE also focused on component-level threats, with little attention to the relationships between components, creating blind spots where emergent, system-level threats went unaddressed. Other challenges included:

Incomplete diagrams

Data flow diagrams that did not detail trust boundaries, components, problem regions, or risk scenarios.

System-level blind spots

Minimal exploration of system-level threats or how threats and components interact.

A single perspective

Inconsistent identification of threats from multiple angles, such as malicious insiders versus external attackers.

Searching for a better way, Davis discovered ThreatModeler after a student requested educational access, and adopted it for the course.

> **Once I learned about ThreatModeler, I realized it was perfect for our purposes.**
>
> **Dr. James Davis**
> Assistant Professor, Purdue University

IN THE CLASSROOM

The course project

Davis took three steps to deepen students' threat modeling skills:

1. **Training.** Systems-thinking content was added to the threat modeling unit, encouraging broader security analysis.
2. **Assessment.** The assignment was revised to ask for more detail on system models, identified threats, and mitigations.
3. **Tooling.** Most significantly, Davis obtained educational access to the ThreatModeler platform.

Equipped with better information and tools, students tackled a semester-long project with expansive functional and non-functional specs.

PHASE 1

Students built a command-line tool to weigh social and technical factors when comparing software components for reuse.

PHASE 2

Students extended another team's codebase to deploy an imitation package registry for JavaScript, with package recommendations from Phase 1, data storage and retrieval, and an ADA-compliant web interface.

ASSESSED ON

What students had to demonstrate

- Distinguish components from systems in threat scenarios
- Recognize systems and components in moderately complex systems
- Identify nuanced relationships between components and how they form systems
- Understand and integrate multiple perspectives on a threat situation

RESULTS IN PRACTICE

Expanding perspectives and skills

By prompting students to consider the relationships between components, the course produced more effective threat models. With ThreatModeler, students built comprehensive models faster, identifying weaknesses and the controls needed to mitigate them with greater speed, ease, and accuracy. Built-in support for common cloud components further simplified their workflows, as did automated guidance on several classes of threats.

ThreatModeler also exposed students to a more comprehensive, systems-level format than STRIDE: one that models the relationships between components across the entire software development lifecycle, not just individual components in isolation. Because the platform is built for users of all proficiencies, students could build complete threat models regardless of their prior security expertise.

> **ThreatModeler itself is easy to use, with a natural experience for system modeling and defining threat boundaries. Because it provides automated guidance on some classes of threats, students give much more detail on the threats they identify and the mitigations they develop.**
>
> **Dr. James Davis**
> Assistant Professor, Purdue University

THE PAYOFF

Outcomes and benefits

The introduction of ThreatModeler, paired with a system-level perspective, produced clear benefits for students and instructor alike.

Higher-quality outcomes	Students showed a deeper understanding of access-related threats and how placing trust boundaries at weak points counters attacks from multiple angles.
Heightened interest	Working with industry-grade tools further stimulated students' interest in cybersecurity as a specialization or career path.
Efficient use of instructor time	Self-serve learning let Davis focus on teaching over training, fitting his learn-by-doing philosophy.

> **"ThreatModeler is a tool that has helped my students learn much better how to build secure web systems, while helping me spend my teaching time more effectively."**

Dr. James Davis, Purdue University

REFERENCES

Agarwal, Archie (Chief Technical Architect, ThreatModeler), "7 Easy Steps for Building a Scalable Threat Modeling Process," 2024

James C. Davis, et al., "A First Offering of Software Engineering," 2022

James C. Davis, et al., "Introducing Systems Thinking as a Framework for Teaching and Assessing Threat Modeling Competency," 2024

ThreatModeler, Amazon Web Services, Microsoft Azure, "DevSecOps Blueprint for Cybersecurity," 2024

ThreatModeler, Amazon Web Services, "5 Steps to Building a Scalable Threat Modeling Program for AWS," 2021