Editor: **Robert Blumen**
SalesForce
robert@robertblumen.com

Adam Shostack on Threat Modeling

Justin Beyer

From the Editor

Adam Shostack, of Shostack & Associates and author of *Threat Modeling: Designing for Security*, discusses threat modeling, its benefits, and how to add it to an organization's existing software process. Host Justin Beyer spoke with Shostack about asset-, threat-, and software-centric approaches; diagramming applications and introducing trust boundaries; methods such as spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege as well as the kill chain and the Elevation of Privilege card game; tooling; bug bars; privacy threats; linkability, identifiability, nonrepudiation, detectability, disclosure of information, unawareness, and noncompliance; selling threat modeling to an organization; and threat modeling for the Internet of Things. We provide summary excerpts below; to hear the full interview, visit http://www.se-radio.net or access our archives via RSS at http://feeds.feedburner.com/se-radio.—*Robert Blumen*

Justin Beyer: What is a threat?

Adam Shostack: A promise of future harm; something that might happen, often accompanied by a condition for avoiding the threat. Threat modeling implements mitigations.

What's the goal of threat modeling, and how is the goal achieved?

It enables security and engineering to collaborate and consider threats that apply to the system, before writing code. It aims to develop a plan to solve threats and avoid them when you ship.

What are its benefits?

It gives a structured, systematic, comprehensive approach to security. Structured threat-modeling techniques identify what can go wrong and provide assurance that you're being comprehensive. Organizations get collaboration, rather than conflict, between teams. If you're not threat modeling, and engineering comes to security for advice on securing the product, security might recommend late-stage techniques, such as software-composition analysis, static analysis, penetration testing, and fuzzing. But all of these happen later in the design and development of the product and the process. When design choices get encoded in the application programming interfaces (APIs) and in the distribution of components, they become hard to change. Security needs a seat at the table from the beginning. Threat modeling diffuses potential conflict.

What kinds of projects benefit from threat modeling?

Any that include technology. Blockchain projects tend to have a threat model either explicitly or implicitly

Digital Object Identifier 10.1109/MS.2020.3017406
Date of current version: 22 October 2020

with them. Similarly, Microsoft has published a guide to threat modeling with machine learning, and the Berryville Institute of Machine Learning also published a guide. Threat modeling involves answering four key questions: what are we working on, what can go wrong, what are we going to do about it, and did we do a good job. An explicit threat model documents the answers to those questions. An implicit model is when someone was told to go think about it, they did, and they said, "Yeah, we're good," or, "No, I think we need to address these particular threats."

Please explain attacker-centric, asset-centric, and software-centric approaches and their advantages and disadvantages.

In attacker-centric modeling, we build personas for the people who could attack us and use them to understand methods and tactics. This approach often leads us astray. Usability experts recommend engaging with personas in their natural environments to understand what they do and why, but you can't talk to ransomware authors or spies about their work. Asset-centric threat modeling identifies what might be valuable to attackers, but it is easy to miss things and waste time on assets that are out of scope. What we think is most valuable may not be what is valuable to attackers. It is best to focus on what we're working on: the software, the technology. Software-centric threat modeling starts from what people know, what they're comfortable talking about, what we can scope and understand, what is within our control.

What are some of the newer methods of threat modeling?

SOFTWARE ENGINEERING RADIO

Visit www.se-radio.net to listen to these and other insightful hour-long podcasts.

RECENT EPISODES

- 423—Host Akshay Manchale speaks with Ryan Singer, head of strategy at Basecamp, about the mindset and culture behind successful remote work for engineers.
- 422—Michael Geers, front-end developer with over a decade experience in building user interfaces, discusses Micro Frontends with host Kanchan Shringi.
- 421—Doug Fawley, tech lead of the golang native implementation of gRPC at Google, discusses gRPC with host Robert Blumen.

UPCOMING EPISODES

- Jeroen Willemsen and Sven Schleier talk about mobile application security with host Justin Beyer.
- Matt Lacey discusses Mobile App usability with host Gavin Henry.
- Host Jeff Doolittle talks with Philip Kiely on writing for software engineers.

The newest is the use of a kill chain to ask what could go wrong. Kill chains come from the idea that an attacker must follow a set of steps to break in successfully. By going through and thinking about each of those steps relative to what are we working on, we can say, "What are the best delivery points here?" For example, if an application accepts file uploads, an attacker could weaponize that capability to execute files on the system.

What does STRIDE stand for, and how do you use it?

STRIDE was the first structured approach to threat modeling, a model of what can go wrong. It stands for spoofing, tampering, repudiation (denying that something happened or that you're responsible), information disclosure, denial of service, and elevation of privilege. STRIDE is a checklist to think through each piece of software you're working on: How could someone spoof themselves to this, pretend to be someone else? How could someone tamper with it or modify it without my authorization? And so on.

After applying threat modeling, how do you identify threats that you need to care about?

First, apply the easy fixes. Treat everything else as part of a backlog. Threat modeling identifies things on which we can do risk analysis later. Sometimes we defer to risk analysis, and sometimes it's easier just to act. With risk, there are four steps you can take: accept risk, transfer it, mitigate it, and eliminate it. Ignore is not an option, but accept might be—accept it, track it, quantify it in some way, and ask if the risk is appropriate for your

ABOUT THE AUTHOR

JUSTIN BEYER is an information security analyst in higher education. He holds various industry certifications for both defensive and offensive cybersecurity. Currently, he has interests in a wide variety of topics, such as software security, software architecture, security operations, identity/access management, and cryptography. He can be reached at justin@justinb.dev or on Twitter @jusbeyer.

seniority within the organization and for the organization as a whole. It goes into the backlog so it can be tracked along with standard bugs. Thinking about threat modeling as an engineering discipline that's designed to produce problems to be addressed leads to the most success. Bug bars encapsulate what the severity is and who's allowed to make decisions about it. Bug bars are well aligned with engineering at most organizations. With bug bars, we say that this severity of impact leads to this level of person needing to decide about the risk.

Does the threat modeling of new software change for API-driven software, the Internet of Things, containerized services running Docker, or some other Kubernetes cluster executing Docker?

It changes the answers to, What are we working on, what can go wrong, and what are we going to do about it? The thing that remains the same across all of these cases is the four-question framework. Just get started. Dive in and threat model; do an analysis. As you develop skill, you will realize that the threats to the web API tend to be like this, the threats to the IoT tend to be like this, and so on. But if I were to try to give advice for each of these specific things, we'd end up with an explosion of variance.

How do I sell the importance of threat modeling to my organization?

Take a structured approach to security and increase collaboration between security and development. Getting everyone at a whiteboard discussing architecture reduces rework. Even if you get no security benefit whatsoever, it gets everyone into the threat-modeling process. Elevation of Privilege is a card game that helps people threat model at a whiteboard. The game is built around STRIDE as a backbone. You don't need to know much about STRIDE to use it. It's a way of answering, What are we working on, and what can go wrong? Its gamified nature encourages a free flow of ideas and perspectives.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscribe to *IEEE Software* by visiting www.computer.org/software.

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.