#28 TECHARTIKEL

Threat Modeling

08.05.2026

Christian Gießler

Inhalt

Threat Modeling

Moderne IT-Systeme sind heute komplexe, stark vernetzte Landschaften und damit ein ideales Ziel für Angreifer. Jede neue Schnittstelle, jede Integrationskomponente und jede Cloud-Verbindung erweitert die potenzielle Angriffsfläche. Die klassische Vorgehensweise, am Ende des Projekts Penetrationstests durchzuführen, reicht längst nicht mehr aus, um diese wachsende Komplexität beherrschbar zu machen.

Genau hier setzt Threat Modeling an. Dabei handelt es sich um einen strukturierten Ansatz, mit dem Teams bereits in frühen Phasen der Entwicklung systematisch Bedrohungen identifizieren, Risiken bewerten und passende Gegenmaßnahmen ableiten können. Anstatt Sicherheitslücken nur zufällig oder spät zu entdecken, wird Security zu einem planbaren und wiederholbaren Arbeitsschritt im gesamten Software-Lebenszyklus.

Dieser Ansatz wird für Unternehmen nicht nur aus technischer Sicht immer wichtiger, sondern zunehmend auch aus regulatorischen Gründen. Durch Vorgaben wie NIS-2 und den Cyber Resilience Act steigt der Druck, Sicherheitsrisiken methodisch zu analysieren und nachweisbar zu adressieren – und Threat Modeling bietet hierfür einen praxistauglichen Rahmen.

1. Ziele und Nutzen von Threat Modeling

Das Ziel von Threat Modeling besteht darin, Sicherheitsrisiken nicht zufällig, sondern systematisch und frühzeitig sichtbar zu machen. Anstatt nur auf akute Vorfälle zu reagieren, verschafft es einen klaren Überblick darüber, welche kritischen Assets geschützt werden müssen, welche Angriffswege realistisch sind und wo die größten Schwachstellen im Design liegen. Dadurch wird die Sicherheit von einem Bauchgefühl zu einer nachvollziehbaren und begründbaren Entscheidung.

Ein zentraler Nutzen liegt in der Reduzierung von Kosten und Aufwand: Sicherheitslücken, die bereits in der Architektur- oder Designphase erkannt werden, lassen sich deutlich günstiger beheben als später im Betrieb oder in der Wartung. Gleichzeitig verbessert sich die Kommunikation im Team, da Fachbereich, Entwicklung, Operations und Security gemeinsam auf Basis des Threat Models diskutieren können, anstatt nur über einzelne technische Maßnahmen zu streiten.

Darüber hinaus unterstützt Threat Modeling dabei, Security-Anforderungen mit der Business-Perspektive zu verbinden. Wer Angriffe und potenzielle Schäden klar benennen kann, kann auch besser priorisieren, welche Maßnahmen wirklich geschäftskritisch sind und welche nur „nice to have". Dies erleichtert Budgetentscheidungen, hilft beim Nachweis gegenüber dem Management, den Kunden und den Auditoren und schafft eine belastbare Grundlage, um regulatorische Vorgaben wie NIS-2 oder den Cyber Resilience Act mit vertretbarem Aufwand zu erfüllen.

2. Regulatorische Anforderungen (NIS-2, CRA)

Mit NIS-2 steigen die Anforderungen an das Risikomanagement für Betreiber wesentlicher und wichtiger Dienste erheblich: Sie müssen Bedrohungen für ihre Netz- und Informationssysteme systematisch identifizieren, bewerten und geeignete technische sowie organisatorische Maßnahmen nachweisbar umsetzen. Threat Modeling bietet hierfür einen strukturierten Ansatz, um eine solche Risikoanalyse durchzuführen und gegenüber Aufsichtsbehörden sowie internen Auditoren nachzuweisen, dass Sicherheitsrisiken methodisch adressiert wurden.

Auch der Cyber Resilience Act (CRA) verlangt von Herstellern, die Cyberrisiken ihrer Produkte über den gesamten Lebenszyklus zu analysieren und „Security by Design“ sowie „Security by Default“ umzusetzen. Ein belastbares Threat Model kann als zentrales Artefakt dienen, um die identifizierten Bedrohungen, die getroffenen Designentscheidungen und die implementierten Schutzmaßnahmen zu dokumentieren. Threat Modeling wird somit nicht nur zu einer bewährten Praxis der sicheren Softwareentwicklung, sondern auch zu einem wichtigen Baustein, um die Konformität mit NIS-2 und CRA effizient nachzuweisen.

3. Der Threat-Modeling-Prozess

Der Threat-Modeling-Prozess folgt typischerweise einer klaren Abfolge von Schritten von der Zieldefinition bis zur Umsetzung konkreter Maßnahmen. Im Zentrum steht dabei stets die Frage: „Was bauen wir, wovor haben wir Angst – und was tun wir dagegen?" Die folgenden Schritte lassen sich pragmatisch in jeden Entwicklungsprozess integrieren.

3.1. Scope und Ziele festlegen

Zunächst wird definiert, was genau betrachtet werden soll: eine einzelne Anwendung, ein Microservice, eine Schnittstelle oder ein gesamter Geschäftsprozess. Außerdem wird festgehalten, welche Sicherheitsziele im Fokus stehen (z. B. Vertraulichkeit, Integrität, Verfügbarkeit, Datenschutz).

3.2. System und Datenflüsse modellieren

Im nächsten Schritt entsteht ein technisches Modell des Systems, oft in Form von Data-Flow-Diagrammen oder Architekturübersichten. Darin werden Komponenten, Schnittstellen, Datenspeicher, externe Akteure und die dazwischen verlaufenden Datenflüsse sichtbar gemacht. So entsteht die Grundlage, um spätere Bedrohungen gezielt zu verorten.

3.3. Bedrohungen identifizieren

Auf Basis dieses Modells werden mögliche Angriffe systematisch gesammelt, beispielsweise mithilfe von STRIDE oder Evil User Stories. Das Ziel besteht darin, für

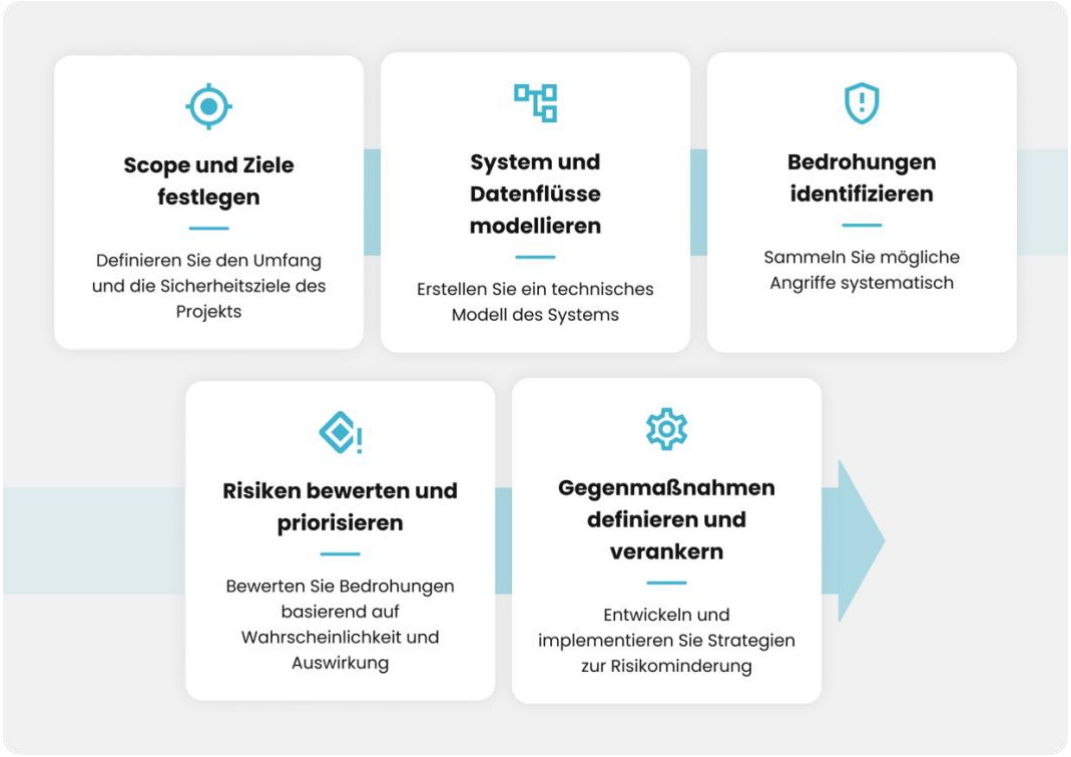
jede Komponente und jeden Datenfluss zu verstehen, wie ein Angreifer vorgehen könnte und welche Sicherheitsziele er dabei verletzen würde.

3.4. Risiken bewerten und priorisieren

Die gefundenen Bedrohungen werden hinsichtlich Eintrittswahrscheinlichkeit und möglicher Auswirkung bewertet, häufig unterstützt durch einfache Risikomatrizen oder Modelle wie DREAD. So entsteht eine priorisierte Liste von Risiken, auf die sich das Team gezielt konzentrieren kann, statt alle Probleme gleich zu behandeln.

3.5. Gegenmaßnahmen definieren und verankern

Für die priorisierten Risiken werden konkrete technische und organisatorische Maßnahmen abgeleitet. Beispiele hierfür sind zusätzliche Authentifizierung, Verschlüsselung, Logging, Härtung von Schnittstellen oder Prozessanpassungen. Diese Maßnahmen werden in Architektur-Entscheidungen, Tickets, Sicherheitsanforderungen und Tests überführt, sodass das Threat Modeling unmittelbar in die Umsetzung und den Betrieb hineinwirkt.



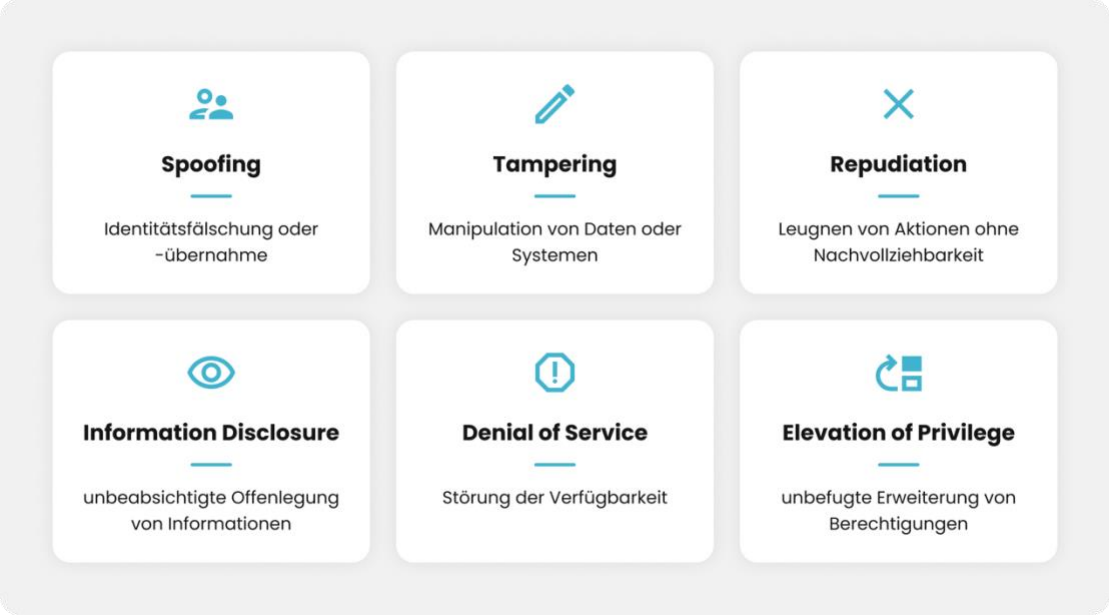
4. Methoden und Frameworks

Für das strukturierte Threat Modeling stehen verschiedene Methoden und Frameworks zur Verfügung, die je nach Kontext, Reifegrad und Zielsetzung ausgewählt werden können. Die gängigsten Ansätze lassen sich grob in kategorienbasierte, risikofokussierte und angreiferzentrierte Methoden unterteilen.

4.1. STRIDE

STRIDE ist das wohl bekannteste Framework und wurde ursprünglich von Microsoft entwickelt. Es kategorisiert Bedrohungen in sechs Typen:

- **Spoofing** – Identitätsfälschung oder -übernahme
- **Tampering** – Manipulation von Daten oder Systemen
- **Repudiation** – Leugnen von Aktionen ohne Nachvollziehbarkeit
- **Information Disclosure** – unbeabsichtigte Offenlegung von Informationen
- **Denial of Service** – Störung der Verfügbarkeit
- **Elevation of Privilege** – unbefugte Erweiterung von Berechtigungen



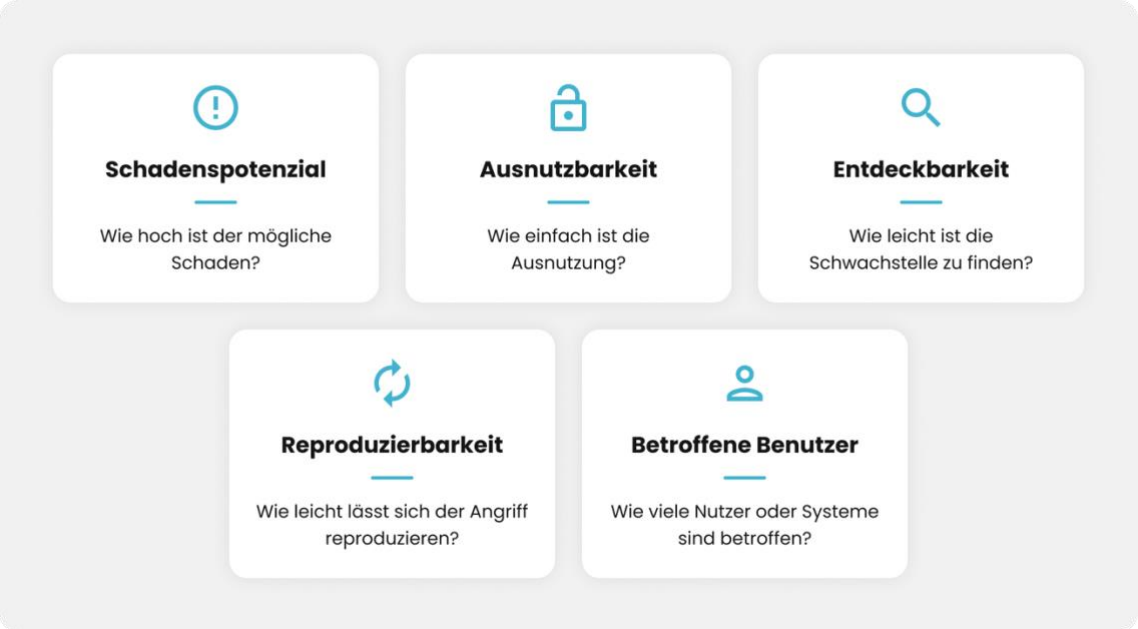
STRIDE eignet sich besonders gut für die systematische Analyse von Datenflussdiagrammen. Denn damit können jeder Datenfluss, jede Komponente und jeder Datenspeicher gezielt auf diese sechs Bedrohungskategorien hin überprüft werden.

4.2. DREAD

STRIDE hilft dabei, Bedrohungen zu identifizieren, während DREAD der Bewertung und Priorisierung gefundener Risiken dient. Das Akronym steht für:

- **Damage Potential:** Wie hoch ist der mögliche Schaden?
- **Reproducibility:** Wie leicht lässt sich der Angriff reproduzieren?
- **Exploitability:** Wie einfach ist die Ausnutzung?
- **Affected Users:** Wie viele Nutzer oder Systeme sind betroffen?
- **Discoverability:** Wie leicht ist die Schwachstelle zu finden?

Typischerweise wird jede Kategorie auf einer Skala von 1 bis 10 bewertet. Der daraus resultierende Durchschnittswert ergibt einen Risiko-Score, der bei der Priorisierung von Maßnahmen hilft. Oft wird DREAD gemeinsam mit STRIDE verwendet: STRIDE identifiziert die Bedrohungen und DREAD bewertet sie.



4.3. Evil User Stories

Evil User Stories sind ein agiler, angreiferzentrierter Ansatz, der sich nahtlos in Scrum- oder Kanban-Workflows integrieren lässt. Anstelle der Perspektive eines legitimen Nutzers werden User Stories aus der Sicht eines Angreifers formuliert, beispielsweise:

„Als Angreifer möchte ich Session-Tokens stehlen, um mich als legitimer Benutzer auszugeben."

Diese Methode macht Bedrohungen für das gesamte Team greifbar und fördert das Sicherheitsbewusstsein, auch bei Entwicklern ohne tiefgreifende Security-Expertise. Evil User Stories können direkt in Backlog-Items, Akzeptanzkriterien oder Testfälle überführt werden.

4.4. PASTA

Der Process for Attack Simulation and Threat Analysis (PASTA) ist ein siebenstufiges, risikobasiertes Framework, das die Bedrohungsanalyse stärker mit den Geschäftszielen eines Unternehmens verknüpft. Die einzelnen Schritte reichen von der Definition geschäftlicher Ziele über die Systemmodellierung bis hin zur Angriffssimulation und Risikobewertung. PASTA eignet sich besonders für Unternehmen mit ausgereiften Security-Programmen, die Threat Modeling in ihre Compliance- und Enterprise-Risk-Management-Prozesse integrieren möchten.

4.5. LINDDUN

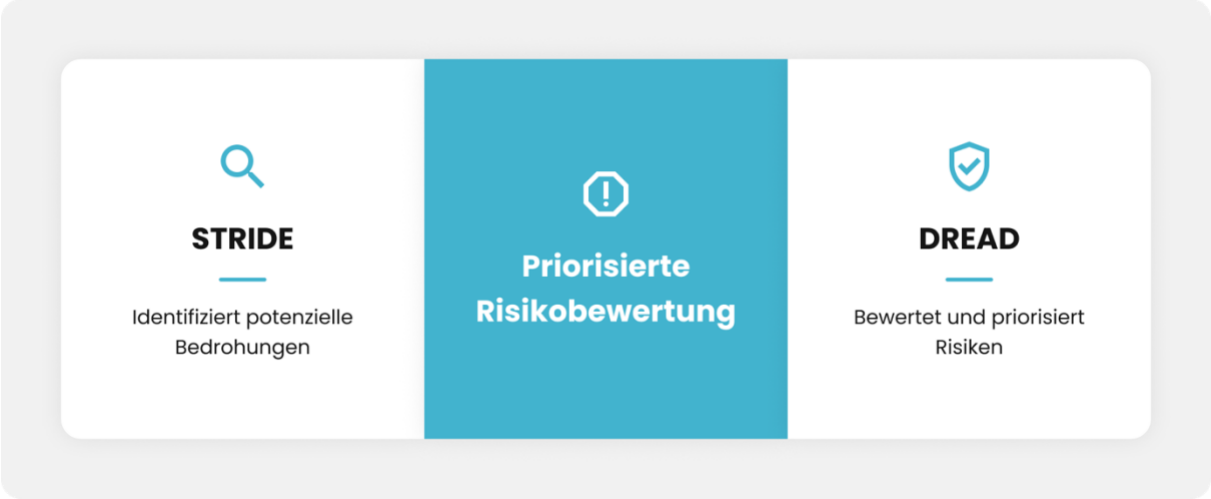
Während sich STRIDE primär auf Security fokussiert, adressiert LINDDUN gezielt Bedrohungen der Privatsphäre. Die Kategorien umfassen Linking, Identifying, Nonrepudiation, Detecting, Data Disclosure, Unawareness und Noncompliance. Dieses Framework ist besonders relevant, wenn Datenschutz und DSGVO-Konformität im Fokus stehen.

4.6. Attack Trees

Bei Attack Trees handelt es sich um einen angreiferzentrierten Ansatz, bei dem mögliche Angriffspfade in Form einer Baumstruktur modelliert werden. An der Wurzel steht das Angriffsziel und die Verzweigungen zeigen alternative Wege, um dieses Ziel zu erreichen. Mit dieser Methode lassen sich komplexe Angriffsszenarien gut visualisieren und Gegenmaßnahmen können gezielt an kritischen Knotenpunkten platziert werden.

4.7. Kombination und Ergänzung

In der Praxis werden diese Frameworks oft kombiniert oder durch weitere Ressourcen ergänzt. Beispiele hierfür sind das MITRE ATT&CK-Framework für reale Angriffstechniken oder CVSS zur quantitativen Risikobewertung. Welche Lösung die richtige ist, hängt letztlich von der Größe und dem Reifegrad des Teams sowie den spezifischen Anforderungen des Projekts ab.



5. Tools für Threat Modeling

Neben Methoden und Frameworks gibt es eine Vielzahl von Tools, die den Threat-Modeling-Prozess unterstützen. Diese reichen von kostenlosen Open-Source-Lösungen bis hin zu umfassenden Enterprise-Plattformen. Welche Lösung die passende ist, hängt von verschiedenen Faktoren ab, darunter Teamgröße, Budget, gewünschte Automatisierung und Integration in bestehende Entwicklungs-Workflows.

5.1. Microsoft Threat Modeling Tool

Das Microsoft Threat Modeling Tool ist ein kostenloses, Windows-basiertes Werkzeug, das besonders gut mit dem Microsoft-Ökosystem (Visual Studio, Azure DevOps) zusammenarbeitet. Über eine visuelle Oberfläche ermöglicht es die Erstellung von Datenflussdiagrammen und wendet automatisch das STRIDE-Framework an, um Bedrohungen zu identifizieren. Für Teams, die bereits im

Microsoft-Umfeld arbeiten, stellt es einen niederschwelligen Einstieg dar. Allerdings wirkt die Oberfläche etwas veraltet und der Funktionsumfang ist im Vergleich zu kommerziellen Lösungen begrenzt.

5.2. OWASP Threat Dragon

Threat Dragon ist ein Open-Source-Tool der OWASP Foundation. Es bietet eine einfache, browserbasierte Oberfläche zur Erstellung von Bedrohungsmodellen. Es unterstützt STRIDE, integriert sich in GitHub und ist gut für kleinere Teams oder den Einstieg ins Threat Modeling geeignet. Die Benutzerführung ist intuitiv. Allerdings fehlen erweiterte Automatisierungsfunktionen und der Community-Support ist begrenzt.

5.3. Threagile

Threagile ist ein Open-Source-Toolkit mit DevSecOps-Fokus. Damit können Architekturen und Komponenten direkt in der IDE als YAML-Dateien modelliert werden. Beim Ausführen werden Risikoregeln automatisch geprüft und Reports generiert. Threagile bietet Teams, die Infrastructure-as-Code und agile Methoden nutzen, einen modernen, code-nahen Ansatz.

5.4. ThreatSea

ThreatSea ist ein modernes Open-Source-Tool für benutzerfreundliche, visualisierte Bedrohungsmodellierung. Es bietet eine intuitive Oberfläche zur Erstellung von Datenflussdiagrammen und unterstützt die gängigen Frameworks STRIDE und DREAD. Dank seiner Webplattform ermöglicht ThreatSea kollaboratives Arbeiten im Team und eine einfache Integration in bestehende DevSecOps-Prozesse. Insbesondere kleinere bis mittelgroße Teams schätzen die Kombination aus einfacher Bedienung und funktionsreichem Threat Modeling, das ohne hohe Lizenzkosten genutzt werden kann.

5.5. IriusRisk

Bei IriusRisk handelt es sich um eine kommerzielle Cloud-Plattform, die automatisierte Bedrohungserkennung mit vordefinierten Komponenten- und

Bedrohungsbibliotheken kombiniert. Das Tool unterstützt Standards wie STRIDE, OWASP, PCI DSS und DSGVO und lässt sich nahtlos in ALM-Systeme wie Jira, TFS und Rally integrieren. Für Enterprise-Umgebungen mit hohen Compliance-Anforderungen ist IriusRisk eine der umfassendsten Lösungen – allerdings sind auch die Lizenzkosten entsprechend hoch.

5.6. ThreatModeler

ThreatModeler ist eine weitere unternehmensweite Lösung, die automatisierte Threat-Modeling-Prozesse mit Integrationen in JIRA, Jenkins und Azure Pipelines bietet. Damit können auch Anwender ohne tiefgreifende Security-Expertise Bedrohungsmodelle erstellen. ThreatModeler ist jedoch ebenfalls im höheren Preissegment angesiedelt.

6. Best Practices und Tipps

- **Früh starten und iterativ vorgehen:** Threat Modeling sollte bereits in der Konzeptphase beginnen und über den gesamten Entwicklungszyklus hinweg regelmäßig aktualisiert werden. So bleiben Bedrohungen auch bei Änderungen und Erweiterungen des Systems stets im Blick.
- **Interdisziplinäre Teams einbinden:** Entwickler, Sicherheitsverantwortliche, Architekten und Fachbereiche sollten gemeinsam am Threat Model arbeiten. Unterschiedliche Perspektiven fördern eine ganzheitliche Bedrohungsanalyse und stärken die Akzeptanz der Sicherheitsmaßnahmen.
- **Methoden anpassen, nicht blind übernehmen:** Je nach Projektanforderungen bieten sich STRIDE, PASTA oder Evil User Stories besser an. Wichtiger als das perfekte Framework ist die konsequente Anwendung und Dokumentation des Prozesses.
- **Automatisierung und Tools nutzen:** Tools wie das „Microsoft Threat Modeling Tool“, „OWASP Threat Dragon“, „Threagile“ oder „ThreatSea“ erleichtern die Modellierung und das Reporting, besonders bei größeren oder agilen Projekten.
- **Regulatorische Anforderungen beachten:** Insbesondere bei NIS-2 und CRA sollten Threat Models auch als Nachweisdokumentation genutzt werden, um Compliance-Anforderungen zielgerichtet zu erfüllen.

- **Security Awareness fördern:** Durch regelmäßige Workshops und die Integration von Threat Modeling in den Entwicklungsalltag wächst das Sicherheitsbewusstsein im Team nachhaltig, was die Security Culture verbessert.

7. Fazit

Threat Modeling ist ein unverzichtbarer Bestandteil moderner IT-Sicherheit. Es verwandelt Sicherheit von einer reaktiven Notmaßnahme in einen proaktiven, planbaren Prozess, der Risiken systematisch sichtbar macht und so dabei hilft, wirtschaftliche Schäden zu vermeiden.

Mit klar definierten Methoden und passenden Werkzeugen lässt sich Threat Modeling effizient in bestehende Entwicklungs- und Betriebsprozesse integrieren und bildet so die Grundlage für robuste, zukunftsfähige IT-Systeme. Gleichzeitig erfüllen Organisationen damit wichtige regulatorische Vorgaben wie NIS-2 und den Cyber Resilience Act.

Wer Threat Modeling frühzeitig und konsequent betreibt, schafft nicht nur technische Sicherheit, sondern stärkt auch das Vertrauen aller Stakeholder – vom Management über Kunden bis zu den eigenen Entwicklerteams.

8. Quellen und weitere Ressourcen

OWASP Threat Modeling Cheat Sheet:
https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html
Umfassender Leitfaden von OWASP

Microsoft Threat Modeling Tool Dokumentation:
https://learn.microsoft.com/de-de/azure/security/develop/threat-modeling-tool-threats
Offizielle Azure-Dokumentation zum Microsoft-Tool

MITRE ATT&CK Framework:
https://attack.mitre.org/
Datenbank realer Angriffstechniken zur Ergänzung von Threat Models

NIST SP 800-154: Guide to Data-Centric System Threat Modeling:
https://csrc.nist.gov/publications/detail/sp/800-154/draft
Offizieller NIST-Leitfaden für datenzentriertes Threat Modeling

Threat Modeling Manifesto:
https://www.threatmodelingmanifesto.org/
Community-getriebene Prinzipien und Best Practices

Weitere Ressourcen

Practical DevSecOps – Types of Threat Modeling Methodology:
https://www.practical-devsecops.com/types-of-threat-modeling-methodology/
10 verschiedene Threat-Modeling-Methoden im Detail

Codecentric Blog – Threat Modeling 101: https://www.codecentric.de/wissens-hub/blog/threat-modeling-101-wie-fange-ich-eigentlich-an
Praktischer Einstieg in Threat Modeling mit Fokus auf erste Schritte

Rapid7 – What is Threat Modeling: https://www.rapid7.com/fundamentals/what-is-threat-modeling/
Grundlagen und Prozess-Erklärung

iteratec Blog – Wie ist dein Threat Model?:
https://explore.iteratec.com/blog/threat-modeling
Praktische Anwendung von Evil User Stories und DREAD

entwickler.de – Threat Modeling: Keine Angst vorm Angreifer:
https://entwickler.de/security/threat-modeling-keine-angst-vorm-angreifer
Aktuelle Perspektiven auf Threat Modeling

Vodafone Business Blog – Bedrohungsanalyse:
https://www.vodafone.de/business/blog/bedrohungsanalyse-beispiel-17542/
NIS-2 und regulatorische Anforderungen

inovex Blog – Threat Modeling with LLM Support:
https://www.inovex.de/de/blog/threat-modeling-with-llm-support/
Moderne Ansätze mit KI-Unterstützung

Legit Security – Threat Modeling Frameworks:
https://www.legitsecurity.com/aspm-knowledge-base/threat-modeling-

frameworks
Umfassender Überblick über Frameworks wie STRIDE, PASTA

Cybersecurity Dive – The Art of Threat Modeling: https://www.cybersecuritydive.com/news/cyber-threat-modeling-framworks-STRIDE-LINDDUN-decisiontrees/713587/
Vergleich von STRIDE, LINDDUN und Attack Trees

Daily.dev – Top 10 Threat Modeling Tools: https://daily.dev/blog/top-10-threat-modeling-tools-compared-2024
Detaillierter Tool-Vergleich 2024

Practical DevSecOps – Best Threat Modeling Tools: https://www.practical-devsecops.com/threat-modeling-tools/
Aktuelle Tool-Liste 2025

Bücher

„Threat Modeling: Designing for Security“ von Adam Shostack – Das Standardwerk zum Thema

„Threat Modeling: A Practical Guide for Development Teams“ von Izar Tarandach und Matthew J. Coles – Praxisnaher Ansatz für agile Teams