

IriusRisk

Beginner's Guide to Threat Modeling



Imagine knowing how cybercriminals target your business before they even strike. You don't need a cybersecurity degree to safeguard your most valuable assets. This eBook demystifies the art of threat modeling, a once exclusive technique for security experts, and empowers you to think like a hacker.

With easy-to-follow guidance and practical examples, you'll uncover potential weaknesses in your systems, predict attacks before they happen, and develop robust defenses that protect your data, finances, and reputation.

It's time to stop being reactive and become proactive in the face of ever-evolving cyber threats.



Brandon Green

Solutions Architect @ IriusRisk

Brandon Green is a Jedi Knight and Sr. Solutions Architect at IriusRisk specializing in designing and implementing threat modeling solutions for enterprise organizations. He has 20 years of IT experience and served 10 years in the United States Air Force, where he worked as a Special Agent focusing on cybercrimes. His interests include writing fiction, sports betting, and video games.

Table Of Contents:

01 - What is Threat Modeling and Why Is It Important	05
1.1 Benefits of Threat Modeling	
1.2 Common Misconceptions About Threat Modeling	
02 - The Threat Modeling Process Step-by-Step	06
2.1 What are you building? Identifying Assets	
2.2 What can go wrong? Identifying Threats	
2.3 Analyzing Risks	
2.3 Prioritizing Risks	
2.4 Implementing Countermeasures	
03 - Creating a Data Flow Diagram for Threat Modeling	08
3.1 Elements of a Data Flow Diagram	
3.2 Best Practices for Creating Data Flow Diagrams	
04 - Identifying and Analyzing Potential Threats	10
4.1 STRIDE Threat Model	
4.2 Attack Trees	
4.3 Threat Intelligence	
05 - Prioritizing and Managing Identified Risks	11
5.1 Risk Rating Methodologies	
5.2 Determining Risk Appetite	
5.3 Implementing Security Controls	
06 - Integrating Threat Modeling into the Software Development Lifecycle	15
6.1 Threat Modeling in Agile Development	
6.2 Continuous Threat Modeling	
6.3 Threat Modeling Tools	
07 - Common Threat Modeling Techniques and Methodologies	19
7.1 PASTA	
7.2 DREAD	
7.3 Trike	

Table Of Contents:

08 - Real-World Threat Modeling Examples and Case Studies	20
8.1 E-commerce Application	
09 - Continuous Improvement and Updating Your Threat Models	21
9.1 Monitoring for New Threats	
9.2 Updating Threat Models	
9.3 Measuring Effectiveness of Threat Modeling	
10 - Conclusion	22

What Is Threat Modeling and Why Is It Important?

Threat modeling is a critical process in cybersecurity that helps organizations understand their security posture. It's a structured approach to identifying, quantifying, and addressing the security risks associated with an application or system.

Think of it like a proactive security strategy - by understanding potential threats early in the development process, you can build more secure systems from the ground up. That's a lot better than scrambling to patch vulnerabilities later on. And much less expensive!

Benefits of Threat Modeling

I've seen firsthand how threat modeling can benefit an organization:

- It helps identify security threats early, when they're easier and cheaper to fix.
- It provides a systematic way to evaluate risk and prioritize security efforts.
- It encourages collaboration between security and development teams.
- It helps ensure that security is an integral part of the design process, not an afterthought.

Common Misconceptions About Threat Modeling

Let's talk about the elephant in the room – those myths about threat modeling that just won't go away:

1. **"Threat modeling is too time-consuming and expensive."**
 - a. In reality, the cost of fixing security issues increases exponentially the later they're discovered. Threat modeling can actually save time and money in the long run.
2. **"Threat modeling is only for big companies with mature security programs."**
 - a. False. Organizations of all sizes can benefit from threat modeling. It's about making security a priority, not the size of your security budget.
3. **"Threat modeling is a one-time activity."**
 - a. Wrong. Threat modeling should be an ongoing process. As new threats emerge and systems evolve, threat models need to be regularly reviewed and updated.

The key is to start small, iterate often, and make threat modeling a regular part of your development lifecycle. Your future self will thank you.

The Threat Modeling Process

Step-by-Step

While there are different methodologies out there, most Threat Modeling workflows follow a similar high-level structure.

Here's a step-by-step breakdown based on my experience:

What are you building? Identifying Assets

Your first step is to identify the assets that need protection. This could be sensitive data, critical functionality, or key infrastructure components. You can't protect what you don't know about.

When tackling a new project, start by identifying what assets you're working with. It's like making sure all your doors and windows are locked at night. We can make sure everything gets the attention it deserves and nothing gets overlooked.

This is called your attack surface.*

* An attack surface is the external-facing area of a system that's susceptible to hacking. It includes all possible entry points and vulnerabilities that an attacker can use to access a system and extract data.



What can go wrong? Identifying Threats

Next, brainstorm all the sneaky ways a bad actor could potentially compromise the system. It's time to think like an attacker and identify any potential threats that could lead to a breach.

I find it helpful to use a framework like **STRIDE** (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) to guide the threat identification process.

Analyzing Risks

Once you've compiled a list of potential threats, dive into analyzing the risks that come with them. Consider the **probability of each threat happening** and the **magnitude of its consequences**.

There are various risk rating methodologies out there, but I prefer a simple High/Medium/Low scale. The key is to be consistent and use a methodology that work for your organization.

Prioritizing Risks

With the risk analysis complete, you can start prioritizing which threats to address first. Focus on the high-risk items that are most likely to be exploited and could cause the most damage.

Remember, you can't eliminate all risks. The goal is to reduce it to an acceptable level based on your organization's risk appetite.

What are you going to do about the threats you identified? Implementing Countermeasures

With the risks laid out on the table, the next move is to put security controls in place to address them. Think about strengthening security features, optimizing configurations, or revamping operational processes – whatever it takes to keep your system secure.

The key is to strike a balance between security and usability. Security controls that are too restrictive will just encourage users to find workarounds.

Threat modeling is an iterative process - as you implement countermeasures, new threats may arise.

That's why it's important to regularly review and update your threat models.

Creating a Data Flow Diagram for Threat Modeling

One of the most useful tools in the threat modeling practitioner's toolkit is the data flow diagram (DFD). A DFD is a graphical representation of how data moves through a system.

Drawing up a DFD is an awesome way to get a bird's-eye view of your system and pinpoint any potential entry points that hackers might exploit. Trust me, *when you can see the whole*

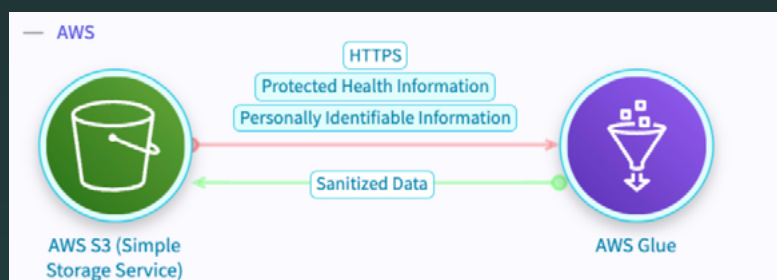
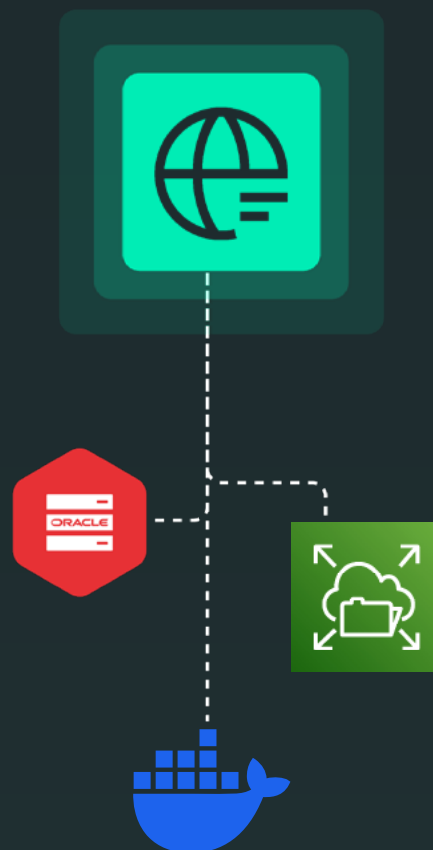
Elements of a Data Flow Diagram

A DFD consists of four main elements:

1. **External entities** - these are outside actors that interact with the system, like users or third-party services.
2. **Processes** - these are the activities that manipulate data within the system.
3. **Data stores** - these are the places where data is stored, like databases or files.
4. **Data flows** - these are the paths that data takes as it moves through the system.

Visualizing the system's components and how they interact reveals the path data flows take, exposing potential weak points along the way.

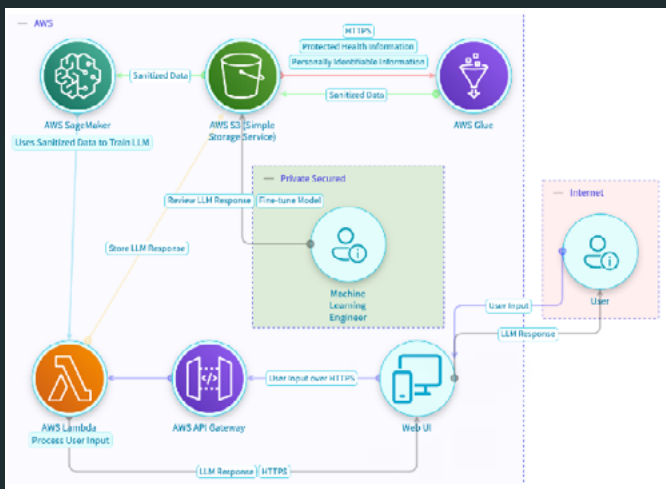
I'm using IriusRisk's built-in diagramming in this example.



Best Practices for Creating Data Flow Diagrams

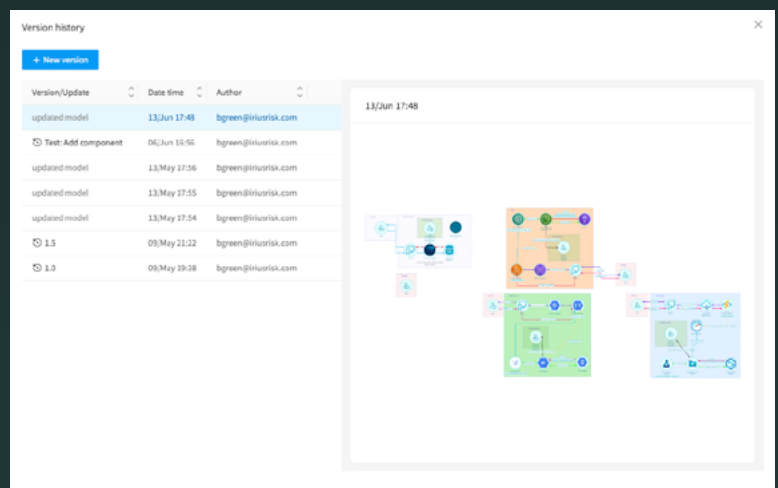
Creating effective DFDs is an art, and I've got some insider tips from my experience to help you master it.

- Keep it simple - start with a high-level diagram and add detail as needed
- Use consistent notation - this makes the diagram easier to understand and maintain
- Focus on the data - the goal is to understand how data moves through the system, so don't get bogged down in implementation details
- Collaborate with stakeholders - DFDs are a great communication tool, so involve the right people in their creation and review them at least annually.



Remember, a DFD is a living document. As the system changes, so should the diagram. Regular updates will help ensure that your threat model stays accurate and relevant.

Create different versions of the diagram as changes are made so you can see the changes over time.



By understanding the risks and implementing the right countermeasures, you'll create applications that are as resilient as they are functional.

Identifying and Analyzing Potential Threats

Once you've got your data flow diagram created, it's time to put on your hacker hat and start thinking like an attacker. This is where the real fun begins - identifying potential threats at each point in your diagram.

It's all about getting inside the mind of a malicious actor and considering all the creative ways they might try to exploit vulnerabilities and compromise your system.

The STRIDE threat modeling framework is great for this.

STRIDE helps you break down threats into six main categories:

STRIDE Threat Model

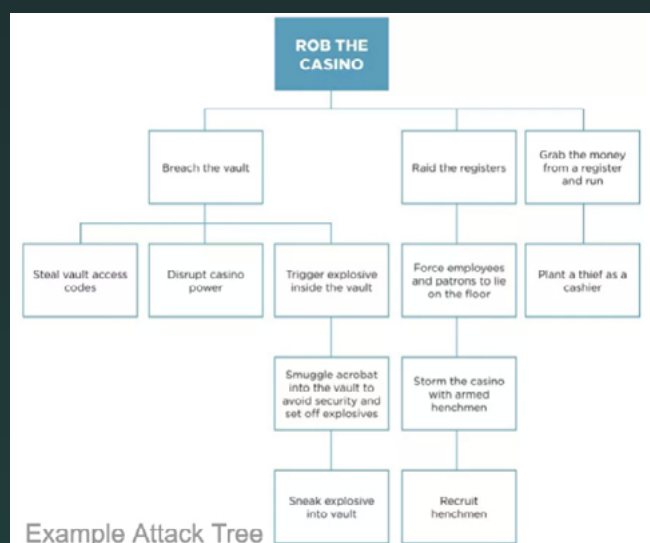
STRIDE stands for:

- Spoofing: Attackers pretending to be someone or something they're not to gain unauthorized access
- Tampering: Modifying data or code for malicious purposes
- Repudiation: Performing actions that can't be traced back, denying involvement
- Information Disclosure: Exposing sensitive info to unauthorized parties
- Denial of Service: Disrupting system availability and resources
- Elevation of Privilege: Gaining higher-level permissions than authorized

By systematically analyzing each element of your data flow diagram through the lens of STRIDE, you can uncover a wide range of potential threat scenarios to assess and prioritize.

Attack Trees

Imagine an attack tree as a roadmap used by cybercriminals to infiltrate your business. Like mind maps, these visual diagrams dissect every potential route an attacker might take to reach their goal, whether it's stealing sensitive data or disrupting your operations. Each branch of the tree represents a different attack scenario, detailing the specific steps and vulnerabilities an attacker could exploit to achieve their objective.



For instance, consider a simple attack tree targeting a company's financial data:

- **Goal:** Steal customer credit card information.
- **Branch 1:** Phishing attack
 - Step 1: Send a fraudulent email impersonating a trusted source.
 - Step 2: Trick employees into clicking a malicious link.
 - Step 3: Install malware on the employee's computer.
 - Step 4: Use malware to exfiltrate data from the company's network.
- **Branch 2:** SQL injection attack
 - Step 1: Identify a vulnerable web application.
 - Step 2: Craft a malicious SQL query.
 - Step 3: Inject the query into the web application's input field.
 - Step 4: Extract sensitive data from the database.

By mapping out these attack paths, you gain a comprehensive understanding of the various ways your system could be compromised. This insight allows you to prioritize your defenses, fortify vulnerabilities, and develop mitigation strategies for each scenario. Attack trees reveal the “attack surface” – the total area where your systems are exposed to potential threats – and empower you to proactively defend against them.

Threat Intelligence


Speaking of threat intelligence, it's an absolute must for a robust threat model. You've got to stay plugged into the latest threat data from reputable sources like US-CERT, your industry's ISAC, and commercial threat intel feeds. I've learned the hard way that you can't just rely on your own experiences and imagination when it comes to anticipating threats. Real-world threat data on active threats, vulnerabilities, and attack patterns needs to inform your threat modeling process.

Prioritizing and Managing Identified Risks

So you've uncovered a bunch of potential threats - now what? It's time to prioritize based on risk level and figure out your plan of attack (pun intended). Not all threats are created equal. You've got to assess the likelihood and potential impact of each one to determine which to tackle first.

This is where risk rating methodologies come into play.

Threat details


 AWS API Gateway / Networking


An attacker eavesdrops on the communication between the clien...

Medium Risk


Description :

Eavesdropping on communication is a network attack that captures network packets transmitted by other computers and reads the data content. This type of network attack is most effective when weak encryption services are used. An attacker could eavesdrop on the communication between the client and server and decrypt the encrypted data.


 8 Recommended

☐  AWS API Gateway


Enable request validation in API Gateway

☐  AWS API Gateway

Use client-side SSL certificates for HTTP backend authentication

☐  AWS API Gateway

Rotate Expiring SSL Client Certificates

☐  AWS API Gateway

Enable Content Encoding for API Responses

Risk Rating Methodologies

There are a bunch of different risk rating models out there, like CVSS or a custom scoring system. Whichever one you choose, the key is to consistently evaluate each threat based on factors like:

- How easy is it to exploit?
- How much damage could it cause?
- How many users would be impacted?
- What's the risk to sensitive data?
- Could it result in compliance violations?

The screenshot displays a 'Threat details' window for an AWS API Gateway threat. The title is 'An attacker injects new items into an existing command to execu...'. A red box highlights the 'High Risk' status. The description states: 'An adversary looking to execute a command of their choosing, injects new items into an existing command thus modifying interpretation away from what was intended. Commands in this context are often standalone strings that are interpreted by a downstream component and cause specific responses. This type of attack is possible when untrusted values are used to build these command strings. Weaknesses in input validation or command construction can enable the attack and lead to successful exploitation.' The 'Countermeasures' section shows '1 Improper Neutralization of Special Elements used in a Command ('Command I...'. The right sidebar shows a 'Risk summary' with 'Inherent risk: High', 'Current risk: High', and 'Projected risk: High'. At the bottom, there are buttons for 'Enable request' and 'Recommended'.

Running through these factors helps you put numbers to the risk and rack and stack your priorities. In my experience, a solid risk rating process is crucial for getting everyone on the same page and making smart tradeoffs.

Determining Risk Appetite

You've also got to have a good handle on your organization's risk appetite - how much risk are you willing to accept? Every company has a different threshold based on their industry, regulations, and culture.

Threats that fall below that line can be deprioritized or accepted, while those that exceed it need to be mitigated asap.

And it's not a one-and-done deal - **risk appetite can shift over time**, so it needs to be revisited regularly.

Implementing Security Controls

Once you've got your prioritized list of risks, it's time to implement security controls to mitigate them.

Security controls can be preventative (stopping threats before they happen), detective (identifying threats in progress), or corrective (minimizing impact after the fact)

The screenshot shows a web application security tool interface. At the top, there's a 'Countermeasure' header with a 'Recommended' badge, a 'Save' button, and a close button. The main title is 'Use client-side SSL certificates for HTTP backend authentication'. Below this, there's a 'Medium priority' dropdown and a 'Create issue...' button. A dropdown menu is open, showing priority levels: 'Calculated (Medium)', 'Very high', 'High', 'Medium', and 'Low'. The 'Medium' option is selected. The main content area contains a detailed description of the countermeasure, including a 'Note' and 'Impact' section. On the right, there's a 'Fields' section with a 'Test state' dropdown set to 'Partially tested', a 'Result source' dropdown set to 'Manual', and an 'Expiry date' field set to 'dd/mm/yyyy'.

Countermeasure

Recommended Save ... X

Use client-side SSL certificates for HTTP backend authentication

Medium priority Create issue...

Calculated (Medium)

Very high

High

Medium

Low

You can use API Gateway to generate an SSL certificate and then use its public key in the backend to verify that HTTP requests to your backend system are from API Gateway. This allows your HTTP backend to control and accept only requests that originate from Amazon API Gateway, even if the backend is publicly accessible.

Note

Some backend servers might not support SSL client authentication as API Gateway does and could return an SSL certificate error. For a list of incompatible backend servers, see Amazon API Gateway important notes.

The SSL certificates that are generated by API Gateway are self-signed, and only the public key of a certificate is visible in the API Gateway console or through the APIs.

Impact:

None

Fields

Test state

Test result: Partially tested

Result source: Manual

Expiry date: dd/mm/yyyy

The key is striking the right balance. Try to bake security in as early as possible with secure design principles, rather than bolting things on later. This is referred to as “Shifting Left”.

Addressing risks early in the development process with the right security requirements and architecture decisions is way more effective than scrambling to patch things up post-release.

This is the benefit of Threat Modeling!

Integrating Threat Modeling into the Software Development Lifecycle

Now, threat modeling isn't just a one-time exercise - it needs to be a continuous process integrated throughout the software development lifecycle (SDLC).

From design to deployment to maintenance, threat modeling should be a regular, iterative part of how you build and evolve software.

This is how you catch risks early and bake security in from the start.

Threat Modeling in Agile Development

Agile moves fast, but that doesn't mean threat modeling goes out the window. You've got to find ways to weave it into each sprint.

In my experience, it's all about breaking threat modeling down into bite-sized chunks that can fit into agile cadences. Rather than one big upfront model, we do lightweight threat modeling sessions during sprint planning, backlog refinement, and retrospectives.

Developers play mini threat modeling games, security champions do story-level threat analysis, and we always have a security-focused acceptance criteria for each user story.

It's not perfect, but it keeps security top of mind.

Continuous Threat Modeling

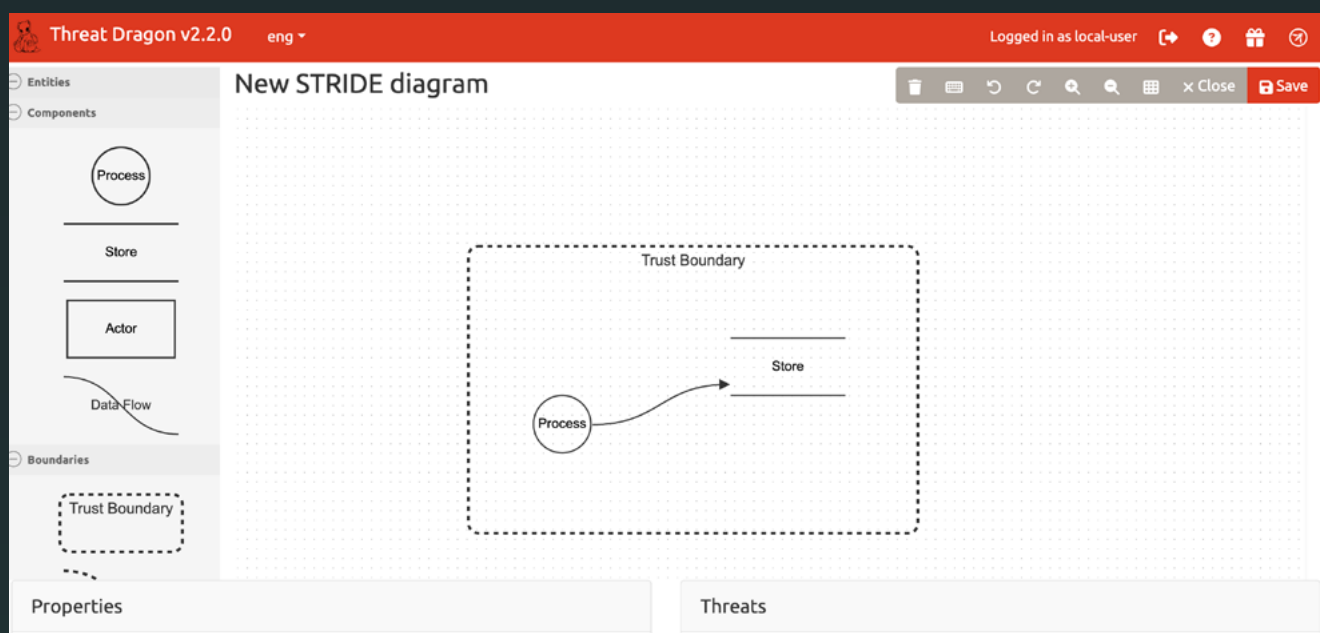
The main idea is to make threat modeling a continuous, living process. As the system evolves with new features and the threat landscape shifts, your threat models need to keep pace. You've got to instill a mindset of continuous improvement, where developers are constantly thinking about "what could go wrong" and "are we still doing the right things" in terms of security.

OWASP Threat Dragon

OWASP Threat Dragon is an open-source threat modeling tool that allows users to create data flow diagrams and identify threats using the STRIDE methodology. It is a web-based tool that can be used online or installed locally.

Threat Dragon provides a simple, intuitive interface for creating threat models and supports collaboration between team members. I appreciate Threat Dragon's open-source nature and its focus on simplicity and ease of use.

It's a great option for teams that want a lightweight, flexible threat modeling tool that can be easily customized and extended to meet their specific needs.

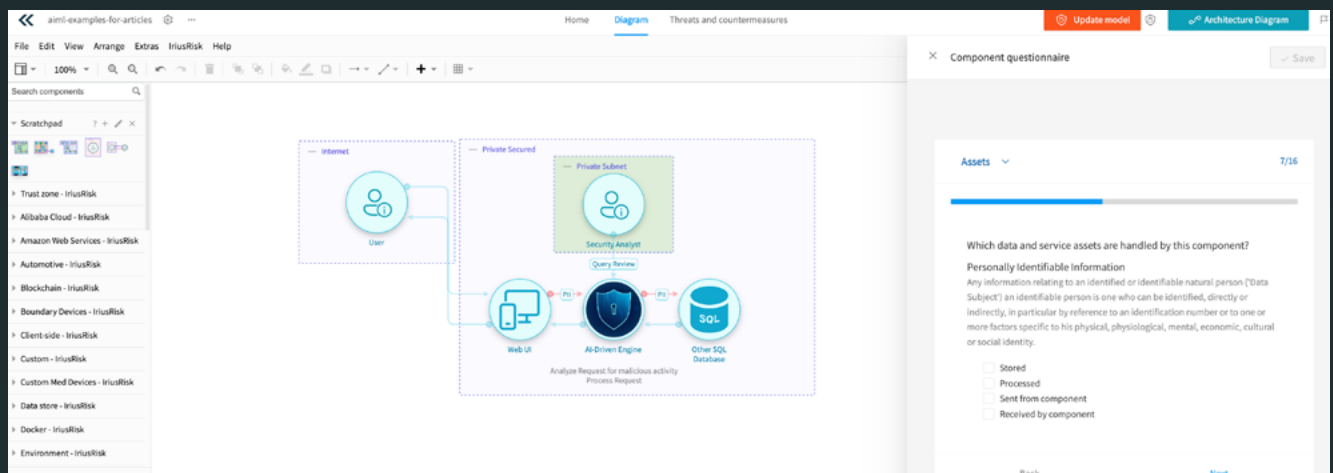


IriusRisk

IriusRisk is a powerful automated threat modeling tool designed to streamline and simplify the threat modeling process for teams of all sizes.

By leveraging a vast library of threat patterns and security standards, IriusRisk automates the identification and prioritization of threats based on your specific system architecture.

This significantly reduces the time and effort required for manual threat modeling, allowing you to focus on implementing effective countermeasures.



Common Threat Modeling Techniques and Methodologies

There are several common threat modeling techniques and methodologies that organizations can use to identify and prioritize potential threats. We talked about STRIDE, but some other examples are: PASTA, DREAD and TRIKE.

Each of these methodologies has its own strengths and weaknesses, and the choice of which to use depends on the specific needs and context of the organization.

PASTA

PASTA (Process for Attack Simulation and Threat Analysis) is a seven-step threat modeling methodology that focuses on aligning technical security requirements with business objectives. It involves defining business objectives, identifying security requirements, decomposing the system, analyzing threats, and specifying countermeasures.

PASTA emphasizes the importance of collaboration between security and business stakeholders. I've found PASTA to be particularly effective in complex environments where multiple stakeholders need to be involved in the threat modeling process.

By bringing together different perspectives and expertise, PASTA helps ensure that all relevant risks are identified and addressed.

DREAD

DREAD (Damage, Reproducibility, Exploitability, Affected Users, Discoverability) is a threat modeling framework that was once popularized by Microsoft but has since fallen out of favor there. It assesses security risks by assigning scores to five categories. Each category is rated on a scale of 1 (low) to 10 (high), and the average score determines the threat's overall severity. This provides a quick and easy way to prioritize vulnerabilities for remediation.

DREAD has its limitations such as subjective scoring, which can lead to inconsistencies and doesn't account for the likelihood of a threat being exploited. It also oversimplifies complex risks by reducing them to a single number.

While DREAD can be a helpful starting point in threat modeling, it's crucial to complement it with other methods for a more comprehensive assessment.

Trike

Trike is a threat modeling methodology that focuses on risk management and security auditing. It involves creating a requirements model, an implementation model, and a threat model, and then analyzing the interactions between them to identify risks.

Trike places a strong emphasis on documentation and provides a structured approach to risk assessment.

While Trike can be more time-consuming than some other threat modeling techniques, I've found that its thoroughness and attention to detail can be invaluable in high-risk environments where a deep understanding of potential threats is critical.

Real-World Threat Modeling Examples and Case Studies

Examining real-world examples and case studies of threat modeling can provide valuable insights into how the process is applied in practice. These examples demonstrate the benefits of threat modeling and highlight common challenges and best practices.

By learning from the experiences of others, organizations can improve their own threat modeling processes and avoid common pitfalls.

E-commerce Application

An e-commerce application is a prime candidate for threat modeling due to the sensitive nature of the data it handles, such as customer information and payment details.

Threat modeling for an e-commerce application would involve identifying assets, such as the customer database and payment processing system, and potential threats, such as SQL injection attacks or cross-site scripting (XSS). The team would then prioritize risks and implement appropriate security controls, such as input validation and encryption.

In one example I worked on, we used the STRIDE methodology to identify potential threats to an e-commerce web application. We discovered several high-risk vulnerabilities, including insecure direct object references and insufficient logging and monitoring.

By addressing these issues early in the development process, we were able to significantly reduce the risk of a successful attack and protect sensitive customer data.



Continuous Improvement and Updating Your Threat Models

Threat modeling isn't a "set it and forget it" type of thing. It's an ongoing process that requires continuous improvement and updating as the threat landscape evolves.

Monitoring for New Threats

The world of cybersecurity is constantly changing, with new threats emerging every day. To stay ahead of the game, you need to keep your finger on the pulse of the latest threat intelligence.

This means regularly monitoring for new vulnerabilities, attack vectors, and emerging trends in the threat landscape. One of the biggest challenges in today's digital world is the rapid pace of digital transformation.

As organizations adopt new technologies and migrate to the cloud, their attack surface expands, creating new opportunities for attackers to exploit.

It's crucial to stay on top of these changes and update your threat models accordingly.

Updating Threat Models

Once you've identified new threats, it's time to update your threat models. This involves revisiting your existing models and incorporating the latest information about potential vulnerabilities and attack scenarios.

It's important to do this regularly, as even small changes to your systems or applications can introduce new risks. In my experience, it's best to update your threat models at least once per quarter, or whenever there are significant changes to your environment.

This could include things like deploying new software, making changes to network configurations, or onboarding new third-party vendors.

Measuring Effectiveness of Threat Modeling

Of course, updating your threat models is only half the battle. You also need to measure the effectiveness of your threat modeling efforts to ensure that they're actually reducing risk.

One way to do this is by tracking security issues over time and analyzing trends. Are you seeing a reduction in the number of vulnerabilities or incidents related to specific threats? Are there areas where your threat modeling efforts seem to be falling short?

By regularly measuring and analyzing these metrics, you can continuously improve your threat modeling process and ensure that it's delivering real value to your organization.



Conclusion

We've covered a lot of ground in this beginner's guide to threat modeling! From understanding the fundamentals to creating data flow diagrams and identifying potential threats, you're now equipped with the knowledge to tackle threat modeling head-on.

Threat modeling is like a dance – it's all about staying in step with your system's rhythm. As the beat changes, so should your moves. Keep fine-tuning your technique, stay on top of the latest trends, and make sure your whole crew is in sync to keep your system grooving securely.

Congratulations on taking this important step in your cybersecurity journey.

Keep learning, keep growing, and most importantly, keep threat modeling!

Automate Threat Modeling to fit your existing SDLC

Secure design right from the start

Visit

www.iriusrisk.com

to book a demo

IriusRisk»