Brought to you by:

ThreatModeler

Threat Modeling

for dummies®

A Wiley Brand

Explore the anatomy of a threat model

Learn to think like a hacker

Automatically create threat models for cloud

Sara Perrott

ThreatModeler™
Special Edition

About ThreatModeler

ThreatModeler is an automated platform that provides a sustainable, self-service threat modeling practice that evolves as your infrastructure grows. ThreatModeler encourages collaboration through its simple process flow diagram-based approach, which aids in the creation of complete threat models that enable even non-security, non-architect users to identify, prioritize, and mitigate threats while communicating them broadly. ThreatModeler provides a holistic picture of your security posture and compliance. Armed with this full view, key stakeholders gain support in making rapid, security-driven business decisions, scaling the organization for growth. Teams are empowered to code fearlessly and build secure deployments from the beginning (design phase), removing security as a blocker in the long run. ThreatModeler integrates with AWS and Azure (with Google Cloud on the way) via its patented onboard Architecture assist and patent pending Accelerator features. With just one click, automatically build threat models for cloud architectures and create task lists to complete threat models.

Threat Modeling

Threat Modeling

ThreatModeler™ Special Edition

by Sara Perrott

for dummies®
A Wiley Brand

Threat Modeling For Dummies®, ThreatModeler™ Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-119-74324-8 (pbk); ISBN 978-1-119-74327-9 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

We're proud of this book and of the people who worked on it. Some of the people who helped bring this book to market include the following:

Project Manager: Martin V. Minner

Senior Managing Editor: Rev Mengle

Executive Editor: Steve Hayes

Business Development Representative: Molly Daugherty

Production Editor: Tamilmani Varadharaj

ThreatModeler Contributors: Archie Agarwal, Alana Ezderman, Dennis Sebayan, Stuart Winter-Tear

Table of Contents

Introduction

Most organizations today have at least a few systems running in the cloud or are planning to move systems to the cloud. The cost savings, optimizations, and other benefits made available by running infrastructure at a cloud service provider are too valuable a proposition to pass up. With new technology and services, however, come new risks that must be identified.

Security architects who are used to creating threat models for traditional on-premises datacenters must now evolve their skill sets to create threat models for the cloud. They must understand the service offerings to comprehend the added risks. They must be up to date on where responsibilities for cloud security lie (for example, the AWS Shared Responsibility Model), and they must understand what hosting information systems in a multi-tenant environment can entail. Though this task may seem daunting, services like ThreatModeler Cloud can simplify the effort with automated threat modeling capabilities and automatically generated task lists. These features help you secure your environment whether you are using AWS or Azure (as of this writing, Google Cloud is under development).

I'm assuming that you, my awesome reader, are a security architect, security engineer, developer, or systems administrator and that you want to learn about threat modeling in the cloud. Maybe your organization is joining others in moving systems to the cloud, or maybe you already have systems in the cloud, and you want to better identify the threats to your information systems and data. Your goal may be to incorporate security into your design processes rather than adding it afterward. Regardless of your motivations, you've come to the right place!

About This Book

Threat Modeling For Dummies, ThreatModeler Special Edition, introduces threat modeling and walks you through the threat modeling process. This book shows you the creation of an

architecture diagram and helps you understand how it fits into the threat modeling process.

With a focus on threat modeling in the cloud, this book prepares you for securing your assets as your organization moves to the cloud. If you're a security architect, you need to understand the challenges of moving your infrastructure to the cloud as well as how best to protect your information systems.

Icons Used in This Book

The icons in the margins of this book indicate information that may be of interest to you. Reading the information that accompanies the icons can expand your understanding of threat modeling. I highly recommend reading them!

Here's what the icons mean:

Tips provide information that may save you time and effort. These are typically based on real-world experience and are there to help you hit the ground running with your first threat model.

The Remember icon identifies material that you should remember above all else in the chapter. I use this icon when presenting information that you'll need to know to make amazing threat models.

When you see the Technical Stuff icon, you've found some great technical detail. This icon marks information that will broaden your technical understanding of threat modeling and the related processes used to create a threat model.

IN THIS CHAPTER

- Exploring the history of threat modeling
- Fitting threat modeling into your DevSecOps environment
- Exploring threat modeling methodologies
- Exploring ThreatModeler's use of process flow diagrams

Chapter 1
Introducing Threat Modeling

Put simply, *threat modeling* is the process of identifying potential threats and appropriately stopping those threats. We all engage in some form of threat modeling in our daily lives, whether it be crossing the road or considering a more robust cycle chain lock. In each case, we assess the risks of a threat (car speed and distance) and take action as needed to mitigate those threats (wait or cross the road). We assess the threats and act accordingly to prevent them. That's threat modeling.

The same applies in information security. However, because of outmoded approaches, when many information security practitioners think of threat modeling, the image conjured is that of a security architect spending many hours on a complicated diagram that will become outdated in a few months.

This perception comes in large part from the historical method of performing threat modeling. However, the discipline of threat modeling has had some important advances. New methodologies and new thought processes have helped to shape threat modeling into a dynamic, scalable process that is easy to keep up to date and gives actionable results.

As organizations move their information systems to the cloud and adopt Agile mindsets, it is even more important that threat models are kept up to date. Consequently, it is crucial that they can be updated quickly because the cloud landscape is constantly changing. New services are added, new systems are set up, and new applications are provisioned in the blink of an eye.

In this chapter, you learn about threat modeling, including its history, common approaches, and methodologies. You also learn about the differences between process flow diagrams (PFD) and data flow diagrams (DFD), as well as how to integrate your threat modeling processes into your DevOps environment.

Understanding Threat Modeling

In the technology sphere, threat modeling involves accurately mapping the component parts along with their roles and functions (an architectural diagram) and uncovering potential threats based on factors such as the protocols used, environment, value, sensitivity of data, and so on. At its simplest, a threat model should accurately diagram and model the application or system, from which a list of potential threats may be derived.


TIP

You may be wondering, "Why should I build a threat model? It sounds like a lot of work, and I already know how to protect my system or application." The short answer is that the security landscape is constantly changing. The protections that work today may be inadequate tomorrow. By adopting a threat modeling strategy, you can stay ahead of issues as they arise with existing systems. Even better, if you create a threat model early in the development process, fixing security issues is simpler and less expensive than it would be to fix them after the system or application has been deployed.

Fully automated threat modeling platforms ingest the architectural diagram and convert it to a threat model automatically. This means developers, as well as non-security technicians and architects (those with little or no security knowledge), can build a threat model without relying on security experts, thus speeding the process and avoiding bottlenecks. Developers who do not understand DFD language can speak the language PFDs provide.

If the threat modeling platform leverages the modern PFD approach, as ThreatModeler does, rather than the traditional DFD approach, the platform will be compatible with the developers' function-oriented view of applications and services perfectly, enabling them to be a part of this process. This synchronization is a much needed benefit of the PFD approach, given that DFDs are foreign and poorly understood by developers.

REMEMBER

For threat modeling to be successful, you must address impacts to the technology side as well as the business side. After all, not all threats are technical in nature.

Social engineering, for example, is not a technical threat but poses a great risk to organizations. Social engineering is often deployed to psychologically manipulate an insider to do a criminal's bidding, and facilitate cyberattacks for the miscreant such as bribery or blunt force; it may also come in a more damaging, subtle guise. A recent stark example is the infamous attack upon Twitter in July 2020 in which hackers used social engineering to hijack the Twitter accounts of prominent and famous users. In this case, social engineering was deployed against Twitter employees. Threat modeling can protect against these types of threats by identifying them and putting security controls in place such as multi-factor authentication, the principle of least privilege, identity and access management procedures, and so on.

TIP

Risk is also an important consideration in threat modeling. For example, if a social engineering attempt is successful, but publicly accessible data is the only thing stolen, the risk to the organization from this threat is low. However, if the same social engineering attempt is successful and confidential information is stolen, the risk is much greater. Threat models allow you to identify the context of a threat so that you can make better decisions on how to reduce or mitigate the risk.

Now that you understand what a threat model is and why you should perform threat modeling in your organization, the next section examines how threat modeling began and how it is evolving.

Exploring the History of Threat Modeling

Threat modeling got its start from system engineers who were attempting to map out how data moves within and interacts with a system. They did this with DFDs. This method became popular in the 1970s after the book *Structured Design* was published by authors Ed Yourdon and Larry Constantine. Only four objects appeared in the early DFDs. They were:

- Data stores
- Data flows
- Processes
- Interactors

TIP

A *data flow diagram* documents the flow of data in, out of, and around a system, application, or process. It looks similar to a flowchart.

Of course, for a threat model, an important consideration would be systems or parts of an application that are at different trust levels. For example, think about a traditional three-tiered application consisting of web servers, application servers, and database servers. The web servers are typically at a lower trust level than the application and database servers. In 2000, the concept of a trust boundary was added to DFDs and threat modeling matured. Now a threat model could not only depict the movement of data, it could also help security architects visualize areas of lesser or greater trust levels.

As DFDs evolved, other approaches to threat modeling evolved as well. The idea of the *threat tree,* for example, was introduced in the 1994 book *Fundamentals of Computer Security Technology* by Dr. Edward Amoroso. The idea of the threat tree was made practical by Bruce Schneier in his article "Attack Trees," published in the December 1999 issue of *Dr. Dobb's Journal.* Schneier proposed starting with your goal as the root of the attack tree and envisioning the attacks to help you achieve that goal as the nodes underneath the root.

Around the same time Schneier's article appeared, Microsoft formally introduced DFDs as its method for creating threat models, utilizing the STRIDE model (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege)

to help identify threats against systems, applications, or processes. In 2002, Bill Gates wrote the "Trustworthy Computing Memo," which described methods to create secure applications. In response to the memo, Microsoft began working on threat modeling tools and methodologies. The first generation of tools produced by Microsoft was referred to as Microsoft Threat Analysis and Monitoring (TAM), which was officially released in 2004. TAM has long since been deprecated in favor of newer tools, such as the Microsoft Software Development Lifecycle in 2011 and the current Microsoft Threat Modeling Tool (TMT), released in 2014.

Microsoft TMT, which is based on the DFD approach, has severe limitations such as needing the manual efforts of security experts to leverage it, lack of interoperability, and the inability to work collaboratively. These limitations mean TMT cannot realistically be scaled or performed at the speed that modern operational environments demand. Additionally, as organizations began to move to the cloud, mindsets regarding development began to change. Instead of the traditional Waterfall method of application development, which was rigid and expensive if changes had to made, Agile methodology began to thrive. Agile allowed smaller iterative changes made more frequently. This approach complemented the flexibility and scalability available with cloud environments.

Because of these developments, Microsoft TMT has sadly become a security bottleneck.

REMEMBER

In the modern operational environment, much emphasis was placed on communication between developers and operations personnel to allow for smaller, more frequent changes to reach production with testing and proper review in place. The process always flowed from left to right — from the developers to the operations personnel. The term coined to describe this process change was *DevOps.*

BLAST FROM THE PAST

You can read Bruce Schneier's work on attack trees in the Archives section of his website, *Schneier on Security:*

`https://www.schneier.com/academic/archives/1999/12/attack_trees.html`

If something had to travel to the left — for example, from Operations back to Development — this meant that an issue had been identified and needed remediating. As Agile processes allowed frequent changes to become more common, the traditional threat modeling approaches became more difficult to create and maintain. In addition, security often became an afterthought. Significant delays resulted if vulnerabilities were found or other security issues had to be addressed.

REMEMBER

The importance of building security into applications and processes earlier on in technology development became recognized as necessary and practical. Because security was included earlier in the process, deployment delays were minimized and more secure applications were released. This model that included security within the process came to be known as *DevSecOps.*

Fitting Threat Modeling into Your DevSecOps Environment

DevSecOps focuses on including security earlier in the development of applications, systems, and processes. This may be for the initial development or might include upgrades and/or updates to the applications, systems, or processes. The goal is to "shift security left" and involve developers in the security process coupled with automation, all with the aim of taking the entire burden of security from security teams. Development and operational teams have grown exponentially in size while at the same time utilizing automation to move at a pace never before imagined. However, security teams have remained static and small. They must embrace automation and include development and operation teams in the process to keep pace. With the addition and growth of Agile methodology, the goal is to continually make small incremental improvements quickly. This approach is so well suited to the cloud environment that cloud development has its own life cycle.

The Cloud Development Life Cycle (CDLC) puts an emphasis on constant, iterative improvements — this should sound familiar if you use the Agile methodology to develop software, systems, or processes. In short, organizations should be able to quickly release bug fixes, implement new features, or even take advantage of new

capabilities in the cloud as they are released. In Agile, these time frames are referred to as *sprints.* With the CDLC, you might, for example, commit to releasing two bug fixes, one feature enhancement, and three security updates in a two-week sprint. This may seem pertinent mainly to developers, but system administrators and operations teams can take advantage of it as well.

This shift to the cloud has introduced the concept of Infrastructure as Code (IaC) templates. Whole systems and networks can be defined in a template that can be deployed almost instantaneously as needed. How can you ensure that your IaC template is deploying a secure network or system? Integrate threat modeling into your DevOps (DevSecOps) processes! You can build the template and that template can be fed to a threat modeling platform where it will be validated against best practices and for security misconfigurations. Once the threat model is complete, the template can be released to operations and consumed by the cloud provider. If a change is made to the template, the process starts all over.


TIP

Using an automated solution like ThreatModeler makes it simple, fast, and painless to update existing cloud threat models so that changes are made visible and additional threats can be called out. As new releases of cloud applications and services are deployed, the threat model evolves, automatically tracking the changes between releases and the correlating updates to threats and mitigations. Threat model versioning is another powerful tool in ThreatModeler, which allows you to threat model the differences between releases and track changes rather than beginning the process again for each release.

Learning About the Various Threat Modeling Approaches

Three approaches to threat modeling are commonly in use. Each is best suited for certain business requirements. The approaches are:

- **Attack-centric approaches** focus on identifying attacks and creating attack trees. The attack trees examine various methods, utilities, and tools that attackers may use. This approach can be effective for penetration testers who are

trying to determine a plan of attack. Process for Attack Simulation and Threat Analysis (PASTA) is an example of an attack-centric threat modeling approach.

- **Asset-centric approaches** focus on identifying assets, what the asset does in your environment, and the asset's criticality within the environment. The reason an attacker might choose to target an asset and how they might attack an asset are parts of asset-centric approaches. Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) as well as Trike are asset-centric approaches to threat modeling.
- **Software-centric approaches** examine application design. One example is the DFD approach popularized by Microsoft in TMT. As problems with this approach have been identified — especially in today's modern, fast moving operational environments — a new approach has been developed: the Visual Agile Simple Threat Model (VAST) exemplified in the ThreatModeler platform.

Did you enjoy all that alphabet soup? I don't know about you, but PASTA makes me hungry. If you aren't familiar with these threat modeling methodologies, the next section is for you. I promise it has nothing to do with food.

Exploring Threat Modeling Methodologies

Five methodologies are commonly used in threat modeling. Each was designed with specific goals in mind, and each looks at risks and threats in a different way. This section examines each of the five in a little more detail and then discusses which methodology might be the best fit for your organization. The methodologies are:

- **STRIDE** is a method Microsoft Threat Modeling uses to categorize threats. STRIDE stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. This methodology requires expert security personnel to implement successfully and utilizes DFDs.

- **VAST** stands for Visual, Agile, and Simple Threat modeling. Unlike other approaches, this enables others to be part of the threat modeling process, even if they have little to no security expertise. VAST was designed for anyone to be able to threat model. This is the heart of the DevSecOps movement for speed and scalability, and the secret sauce is process flow diagramming. This meshes perfectly with organizations leveraging DevOps or Agile models and makes use of two separate threat models — one for application threats and another for operational threats.
- **Trike** is an open source threat model that focuses on cybersecurity risk management. It uses DFDs to show how data flows in an information system. Trike does not easily scale with larger information systems.
- **Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE)** was designed by the Software Engineering Institute (SEI) at Carnegie Mellon. This methodology focuses on non-technical risk (for example, business risk) and does not scale well for large organizations.
- **Process for Attack Simulation and Threat Analysis (PASTA)** utilizes the view of a would-be attacker on identified threats as well as an analysis of the risk and impact of the attack being successful. PASTA is complex and can require training for practitioners to use it successfully.

With all these options, you may be wondering which approach is better for your enterprise. There is no silver bullet, though if you want a threat model that scales well (a rarity in threat modeling methodologies) and does not require extensive training, VAST is the only realistic option.

Progressing from DFD to PFD — Past to Present

The section earlier in this chapter on DFDs showed how they have been leveraged for threat modeling. Although DFDs were great for system administrators because they allowed for the mapping of communications, protocols, and so on, they were inherently limited in their use in cybersecurity, even after trust boundaries were

added. Here's a look at the pros and cons of DFD-based threat modeling.

Pros of DFD-based threat modeling:

- DFD-based threat modeling has been around for a long time.
- It's good for identifying categories of attacks (for example, STRIDE).

Cons of DFD-based threat modeling:

- It requires large amounts of documentation.
- It does not scale.
- It's hard to integrate.
- It tends to produce lots of false positives and false negatives because it does not take into account the context of differing technologies and their impact on overall threats.
- It doesn't take into account the perspective of an attacker.
- A huge learning curve is required to adopt this approach.

A better way of threat modeling was clearly needed. A good threat model needed the ability to scale. It had to be easy to grow and maintain. It had to reflect an attacker's point of view. And it had to be easy for non-security experts. Those are the challenges that PFD-based threat models were created to overcome. Here are the pros and cons of PFD-based threat modeling.

Pros of PFD-based threat modeling:

- It considers the perspective of an attacker (think like a hacker).
- It's easy to understand.
- Onboarding non-security experts is easy.

PFD-based threat modeling has one con — it's a newer method so it isn't as widely known.


TIP

If your organization is currently using DFD-based threat models, you should consider making the change to PFD-based threat models. This approach will allow you to future-proof your threat modeling process, and do so with speed and scale in mind,

utilizing everyone in the security process irrespective of their security knowledge. Ultimately this method will give your organization the flexibility you need to be secure and successful.

Exploring ThreatModeler's Use of Process Flow Diagrams

Now that you know about PFDs and their advantages over DFDs, you may be curious about how to put that knowledge into practice. That's where ThreatModeler comes in.

Consider an online banking application. How would you map out the processes? Check out Figure 1-1. In a PFD, you map out processes from a use case perspective. For example, a user accesses your home page over HTTPS and then continues to the Login page, also over HTTPS, to access the online banking application. Once they are logged in, they have the option of checking statements, checking or updating their profile, or making an ACH transfer. Wire transfers happen in a trusted network — and the icon for wire transfer could potentially be an embedded threat model. That is a threat model inside a threat model, a concept known as *Threat Model Chaining,* patented by ThreatModeler, which reduces rework. Alternatively, if this is a new user and they click Registration, a credit verification is done before they are granted a new account.

In Figure 1-1, as the user moves through the application and interacts with the different functions and processes, each interaction with the application is tracked and modeled. This approach has two significant advantages:

- **Criminals look to subvert standard user behavior to achieve their goals.** For example, the DFD approach would highlight a database within which credit card data is stored and emphasize mitigation controls focused on the database, whereas, in reality, the miscreant may attack the database through the login form to leverage an injection attack. The PFD approach helps to better identify attack vectors.
- **Developers are function- and process-oriented.** When they view and build an application, they do so through the lens of functions and features. Therefore, the PFD view is

natural to them, meaning they can easily be employed in this security process. The DFD approach, however, tracks data as it flows through the system and risk-based trust zones, which is more alien to developers and demands extensive security understanding and training.

ThreatModeler makes it simple for you to create your threat model with an innovative toolbox on the left side of the screen. The toolbox contains common components used in a threat model like those in Figure 1-1, as well as icons that are proprietary to AWS and Azure, so that you can get as specific in your threat models as you'd like.

Once your threat model has been created, ThreatModeler generates a list of recommendations based on industry best practices. As you make adjustments to the threat model, these recommendations are also automatically updated.

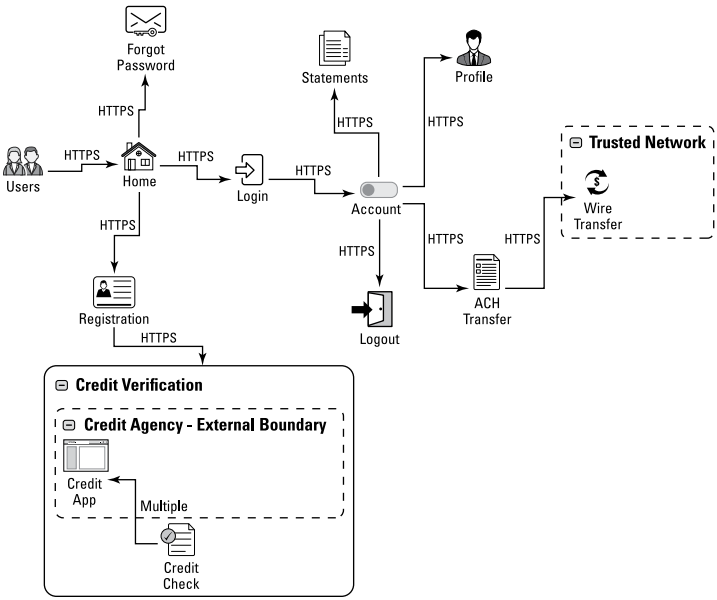

FIGURE 1-1: A process flow diagram (PFD) of an online banking application.

TIP

Alright, I can hear you groaning, "This sounds great, but it'll still take me forever to build out the PFD, and even longer to get the stakeholders to tell me how processes interact with each other so I can build an accurate threat model." I admit I saved the best for last. ThreatModeler has a feature called Accelerator (patent pending) that is available in AWS and Azure. Accelerator for Google Cloud is underway at the time of writing. You can use Accelerator to create PFDs automatically from either AWS or Azure, and it generates environment-specific recommendations for you.

When you've made the recommended changes, you can re-synchronize, and the threat model is updated for you. Has the environment changed? Re-synchronize and the threat model is updated for you. ThreatModeler integrates with your AWS or Azure environment, identifying your infrastructure and automatically creating a PFD diagram for you, meanwhile checking and validating your current security posture. If changes are made to the AWS or Azure environment without the knowledge of the threat modeler, this is taken care of with periodic updates advising you if there is any "drift" and what should be done about that.

Do you see where this is going? If you said, "Automation," you're correct! But what if you have a recommendation that must be addressed by a developer? ThreatModeler can integrate with systems like Jira, ServiceNow, and Azure DevOps to immediately create tickets so that the appropriate team handles the remediation. Once the ticket has been closed . . . what do you do? If you said, "Re-synchrnonize," you get a cookie! Seriously, go get a cookie. I'll wait.

I hope this chapter has you excited about threat modeling and about using PFDs to create dynamic, scalable threat models. Chapter 2 shows you what goes into a threat model — the inputs and outputs, and how they all fit together. Architecture diagrams are, after all, one input in the grand scheme of things.

IN THIS CHAPTER

- Interpreting threat model components
- Fitting the threat modeling components together

Chapter 2
Dissecting the Anatomy of a Threat Model

Many people mistakenly believe that an architecture diagram *is* a threat model. In truth, the architecture diagram is one input to a threat model, providing visibility into how an attacker might see an application or process. Other input ingredients are required to create the perfect threat model.

In this chapter, you expand your knowledge of threat modeling by examining the inputs and outputs of a threat model and seeing how they fit together for a secure Cloud Development Life Cycle (CDLC) process.

Interpreting Threat Model Components

Threat models depend on several types of inputs in order to give useful outputs. The care taken when creating these inputs affects the quality and effectiveness of the overall threat model and the resulting outputs. This section examines some of the inputs that you should account for in a threat model.

Architecture diagrams

When you hear the words *architecture diagrams,* you might think of the older data flow diagram way of thinking. With that method, security experts must analyze the flow of data throughout an application or service while taking into account the sensitivity of the data, plus the trust boundaries through which the data will travel. However, if you leverage process flow diagrams (which I highly recommend), you capture the actual processes, features, and functions involved in an application. For example, an online banking application may have processes defined for login, checking statements, transfers, and requesting loans. It is through these processes and by concentrating on subverting them that an attacker gains his or her prize.

REMEMBER

The focus is not so much on the systems where the application is running; it is on the processes that allow the application to function. This functionality is what attackers abuse when trying to hack into your systems. Thus, leveraging a PFD and adopting an attacker's viewpoint is a valuable part of threat modeling.

Decompose the application

When threat modeling an application, and in the development stage of the architecture diagram, you must do what is called *decomposing the application.* You must be able to understand the relationship of the processes, their flow, and how they interact.

This step in threat modeling requires the security architect (that's you!) to reach out to the subject matter experts (SMEs) for the application. They may be application analysts, developers, or operations, or you may need to call the vendor's engineers. With the help of these experts, you can ensure that you have decomposed the application properly and have captured its inner workings. When you've finished decomposing the application, your architecture diagram is complete. You now have an accurate representation of the application as the basis for building your threat model.

Data classification

When you create your threat model, one factor that can greatly influence the mitigating controls, thwarting potential threats, that you've put in place is how sensitive the data is that you are trying to protect. Protecting personally identifiable information

(PII), for example, requires a greater level of security controls than protecting publicly available data. To properly identify more sensitive data, you would use data classification.

Data classification works by adding a tag to data that represents the sensitivity of the data. The data classification scheme may vary by organization, though using at least three or four tags is typical. A common data classification scheme might look like this and use the following tags:

- **Restricted:** Data that would have significant or catastrophic impact to the business if lost, stolen, or corrupted. This category includes breached data that is considered to be restricted (for example, Social Security numbers, credit card numbers, and banking account numbers).
- **Confidential:** Data that might have a negative impact on the business if lost, stolen, or corrupted. This category may include names, addresses, phone numbers, or other PII that is considered restricted.
- **Internal:** Data that may be sensitive and should not be distributed publicly, such as emails, contacts, and other internal communications.
- **Public:** Data that is made available to the public such as press releases and blog posts.


TIP

The classification scheme and the meaning of each classification level varies by organization. To create your threat model, you should be familiar with your organization's data classification standards.

Use cases

The final inputs that you may want to consider are use cases. How a user interacts with an application can be just as important as how processes interact. For example, say you have an online banking application. One use case might be a customer logging in to check their balance. Another use case might be a customer making a wire transfer. By examining use cases, you can ensure that you have a complete picture, start to finish, of how an application is being used. This information is useful when trying to imagine how an attacker will want to subvert these standard use cases to gain access.

REMEMBER

Your threat models are only as good as your inputs. Spend the time needed to ensure that your inputs are accurate and complete.

The next few sections examine the outputs of a good, complete threat model. From the outputs you can plan remediations, ensure you are in compliance with regulatory requirements, and create reports for executives — such as chief information security officers (CISOs) — on your security posture.

Threats

Threats are among the more obvious outputs of a threat model. After all, you worked hard creating the architecture diagram and classifying your data so that you can identify the threats to that data.

TIP

Though generic threats can provide some value, such as "Weak Access Control for a Resource" (generated by Microsoft TMT), the real strength in identifying threats comes in identifying environment-specific threats. For example, Azure may present different security concerns than AWS. The remediations for those threats rely on different products. The output of your threat model should account for these differences and address the threats that are specific to Azure, AWS, or your on-premises environment.

Security requirements

Does your organization have specific security requirements like those required by the Health Insurance Portability and Accountability Act (HIPAA) or Payment Card Industry Data Security Standard (PCI-DSS)? A good threat model should be able to map threats to these security requirements so that you can see if regulatory compliance is being met.

DID YOU KNOW?

The Food and Drug Administration (FDA) has made the recommendation to medical device manufacturers to include threat modeling as part of their documentation to the FDA before it is released to market. You can download the draft guidance here:

`https://www.fda.gov/media/119933/download`

CIS benchmarks

If your organization uses Center for Internet Security (CIS) benchmarks, you know how valuable they can be. CIS benchmarks are secure baselines for operating systems. Two levels of benchmarks are available:

- **Level 1:** Base recommendations that should have minimal impact to the performance of the system
- **Level 2:** Robust recommendations that can have a negative impact on your systems if not implemented correctly or without proper testing

Level 1 benchmarks meet the requirements of most organizations. They can usually be implemented fairly quickly and the impact is minimal.


TIP

A good threat model should map to the CIS benchmark of your choice to show you if any gaps exist in your adherence to that benchmark.

Test cases

You've done your threat modeling and identified your security requirements. Next, you need to be able to test whether an identified threat is valid or not and test for successful remediation. That's where test cases come in. This raises the concept of threat modeling to drive testing.

Traditionally, services and applications have been "thrown over the wall" to pentesters and red teams in a black-box approach in which they have little or no prior knowledge of the application's internal workings. The hope is that in a given timeframe they will discover vulnerabilities to be remediated. This is, at best, a hit-and-miss approach. Rather, the output from threat modeling should be used to drive the direction of testing. The threat model identifies the critical functions, sensitive operations and data, as well as the potential threats and mitigations.

For example, say an identified threat is that the login page of your web application is vulnerable to SQL injection attack. Your security requirements state that web applications must not be vulnerable to this kind of attack. Thus, your first test case may be to validate that a SQL injection issue exists. The next step may be to

set up input sanitization on the field that allowed the injection to occur, and then test again to validate that the change effectively mitigated the issue. You can then mark this threat as remediated in your threat model.

Fitting the Threat Modeling Components Together

So now you've seen the inputs that go into a threat model, and you've seen the outputs that should be produced by a threat model. How do these components fit together? Take a moment to review the Cloud Development Life Cycle (CDLC).

Though most organizations are familiar with the Software Development Life Cycle (SDLC), a newer concept known as the CDLC has emerged. CDLC was created to address the issues that emerged when SDLC was used with a cloud environment. Organizations needed a life cycle that would better adapt to the state of constant change that exists with cloud applications.

CDLC meets this need and better allows for:

- **Deployment:** Using templates and API-driven automation, systems can be created, maintained, and decommissioned with less need for intervention from a system administrator.
- **Testing:** Because security testing can be performed on development systems rather than production systems, testing can be done more frequently and at far less cost.
- **Collaboration:** By testing cloud security tools in the development environment, you can assure developers that the security tool won't cause problems with their application. You can utilize their processes to push changes in security tools to production.

Figure 2-1 shows how threat modeling helps to secure your CDLC environment.

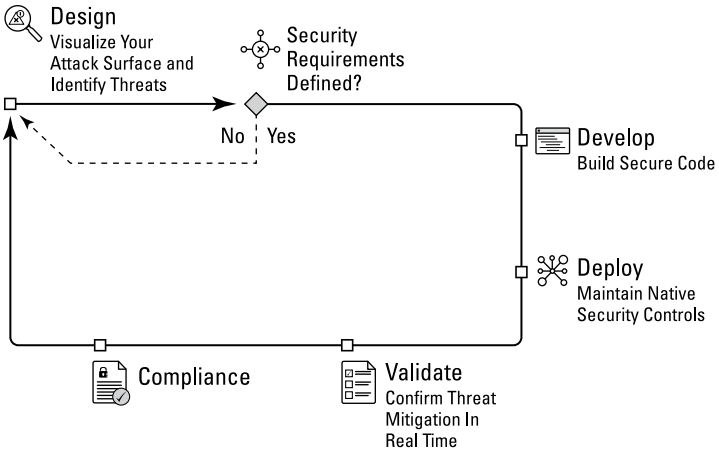

FIGURE 2-1: An illustrated view of a secure CDLC.

The process starts with a completed threat model. The outputs of the threat model — including the threats, security requirements, CIS benchmarks, and test cases — are analyzed. Remediations to the issues are identified, developed, deployed, tested, and validated. New issues found can be sent to a ticketing system like Jira, Azure DevOps, or ServiceNow for remediation. A new solution is designed, and the process repeats. Much like threat modeling, the CDLC is a continuous process — a cycle of design, development, deployment, validation, management, a return to design, and so on.

REMEMBER

By embracing threat modeling and making it part of your CDLC, you can better maintain a secure CDLC and a more secure cloud environment.

TIP

ThreatModeler helps you to create and maintain a secure CDLC by enabling you to create threat models that can be updated with a simple, automated analysis of the environment. New, identified security issues are environment-specific and can be sent to ticketing systems like Jira, Azure DevOps, and ServiceNow for remediation. ThreatModeler can even automatically sync with your environment so that as you remediate issues, they are marked as remediated within ThreatModeler.

IN THIS CHAPTER

- Considering use cases
- Thinking like an attacker
- Integrating threat modeling into a secure CDLC
- Looking toward the future

Chapter 3
Approaching Threat Modeling Today

It's easy to think of threat modeling as a do-it-once mentality, but that is exactly what you need to steer away from. A proper threat modeling practice lives and evolves in the same manner in which it models. As applications and services change, so should the threat model that monitors for and tracks those changes.

ThreatModeler exemplifies this concept with versioning functionality. As new releases are designed and deployed, a new version of the threat model is created. As new changes are modeled, new threats and mitigations are automatically generated. Consequently, the differences between application versions are threat modeled in context. This gives the ability to see what changes between versions have taken place and assess the attack surface anew as these changes take place.

Threat models are not simply a check mark on an auditor's checklist; they are a way to illustrate that you are aware of the threats in your environment and have considered how best to mitigate them. In short, threat models help your chief information security officer (CISO) and you sleep better at night knowing a baseline of security is in place before deployment!

In this chapter, you learn about the "whys" of threat modeling. You learn about how and why real people have used threat modeling and evolved their threat modeling processes to better keep up with the changing tide of technology.

Telling the Stories

I can think of no better way to introduce the "why" of threat modeling than to examine some case studies and the user stories behind them. Looking at the user's perspective and challenges brings a case study to life. To preserve the privacy of those involved, the names of the companies and employees were withheld.

A financial technology company

The first case study is that of a well-known financial technology company. When this organization first transitioned from on-premises datacenters to AWS, it was using manual methods of threat modeling. The company used tables to list what could go wrong and then associated a Jira ticket that contained the mitigation. Different teams used different approaches, however, and the organization wanted a single solution that would work across all teams.

The company began using the Microsoft Threat Modeling Tool (TMT), but found that it was meant to be a desktop tool used by a single person. For one person working alone, creating a threat model is a lengthy process because that person must coordinate and communicate with multiple parties in order to understand the architecture well enough to document it. Ideally, the threat modeling tool should be collaborative.

The organization began using ThreatModeler, which allowed it to develop threat models as a group and store them in a single place. With integrations into its continuous integration/continuous delivery (CI/CD) pipeline, the company went from producing 150 threat models a year to more than 1,000 per year with the same number of people. Those threat models were updated in real time so that as the architecture changed, new threats that emerged were identified and mitigated.

As a result, cloud security was built into the design phase of the technology development cycle, thus saving money over fixing security issues later in the cycle. This approach reduced the effort the security team expended to keep its threat models up to date, while ensuring the accuracy of the threat models generated. It also made security decisions simpler for security leadership because they were able to map decisions to policy compliance illustrated by the threat models that their teams had generated.

A financial institution

The second case study involves a Fortune 500 financial institution. Financial institutions are constantly under attack, and attackers are constantly thinking of new ways to exploit their victims. This financial institution wanted to stay ahead of the attackers and conduct monthly threat assessments that included threat modeling. The company's problem was that its processes couldn't keep up with emerging threats or even keep up with developers as new features were created and released.

To address the problem, the company moved from manual threat models to automated threat modeling. The newer threat modeling process integrates with the company's CI/CD pipeline so that security issues are mitigated long before they reach production.

This organization leveraged ThreatModeler's Accelerator feature (patent pending) as it migrated to the AWS cloud and closed its data centers, estimating an increase of 566 percent in the speed and efficiency of threat modeling. Couple this with a 99.4 percent reduction in the time needed to make go/no-go security and deployment decisions!

REMEMBER

I'm sure you noticed a similarity between the two case studies. Both organizations moved to a threat modeling process that could integrate with their CI/CD pipelines and seamlessly integrate with their cloud architecture, so that security issues were identified and mitigated earlier in the process. With security issues identified sooner, mitigating the issues was less costly, a more secure product was released, and the attackers were thwarted!

Understanding the Problems

When you think about why threat modeling is important, there are a couple of common problems or challenges that most organizations share to some extent. This section examines a few of these problems.

Managing Infrastructure as Code (IaC)

As workloads transition to the cloud, the old practice of purchasing and building individual servers from scratch is becoming history. Why build individual servers with such an error-prone method when you can simply create a template to ensure your systems are consistently created the same way — each time at speed? When you begin building systems from templates, however, the next issue is how to ensure your template is building a secure product. How do you prove the template is building a system that meets your organizational security baseline or maybe even the Center for Internet Security (CIS) Benchmarks?

The answer is to look at your templates as Infrastructure as Code (IaC). You may be thinking, "Well, of course! They're templates!" However, in my experience, these templates are often created by well-intentioned system administrators and are never validated by a security team, nor are they kept in some form of version control so that changes can be tracked over time.

With ThreatModeler's patented Architect feature, cloud templates can be created securely by anyone, irrespective of their security knowledge. Architect is like an onboard security architect guiding the user. Just because someone is tasked with creating a secure AWS environment for an application doesn't mean they know how. Architect analyzes cloud components as they're chosen for the Template and suggests security steps. These steps can be implemented manually or with one-click execution, such as the need for a security group when selecting an EC2. This functionality has the dual benefit of scaling the threat modeling process out to others who are not necessarily security trained!

With IaC, it is important to treat the development and maintenance of these templates as a software development function. A central source control should be used to track the initial creation and allow the security team to review it. Threat modeling

and additional scanning — for example, with a Static Application Security Testing (SAST) tool — may also be performed against the template to ensure that a secure system is being built. Additional vulnerability scanning may be done against a virtual machine built from the template to catch any flaws that were missed.

SAST examines code for vulnerabilities. Another type of testing, Dynamic Application Security Testing (DAST), identifies vulnerabilities by examining applications while they are running.

By managing templates as part of your software development process, you will release far more secure systems and have a higher degree of confidence that they meet all the security requirements laid out by your organization's policies and standards.

Isolating security

In many organizations, security teams sit in a silo kept away from the rest of the organization. Security functions are likewise kept separate, and very little communication takes place among security, development, and operations teams.

This old method of operation has an inherent flaw. Security should be treated as a more social construct. Security policies must be communicated to the organization, and security changes that might affect the uptime of systems must be communicated to the other IT teams.

Threat modeling, too, must be considered a social activity. A security architect working on a threat model in a silo will take much longer to come up with a good threat model, and it may still be incomplete. By bringing together the application stakeholders (system administrators, developers and so on), a security architect can get the full picture and make exceptional threat models in less time. Input should be received from multiple teams. Thus, a good threat modeling platform must facilitate collaborative working.

Decomposing complex applications can also be beneficial. For example, creating one threat model with the developers that illustrates the inner workings of the application, and another with the system administrators focusing on the interactions between systems, can provide a much more complete picture. With the ability to nest threat models, you can reference the developer's threat

model from inside the system administrator's threat model and get a full understanding from start to finish. The security architect could never do all that in a silo.

Another method may be to chain threat models together. ThreatModeler's patented threat model chaining feature enables one threat model to house another entire threat model within it, giving a holistic macro view of a large, connected system as its constituent parts interact with each other.

Thinking Like an Attacker

You can see the value in threat modeling, but what problems does it really help you solve? One of the biggest problems is trying to think like an attacker to determine how the threats identified could be used against you. The great thing is that when you use process flow diagrams (PFDs), you can look at your architecture diagram and your threat models from an attacker's viewpoint.

This capability provides invaluable data to the security architect, the entire security team, and the organization as a whole. As attackers tend to subvert processes with which standard users interact, it makes sense to build the model with PFDs, which are also intuitively familiar to developers who tend toward a process and function view — great for onboarding them into the threat modeling process.

The team at ThreatModeler refers to threat modeling as "evil brainstorming." I like that phrase. When you create a threat model, you look at your system, application, or process as if you were an attacker and try to identify the ways you might be able to compromise it.

Although threat modeling provides a way for you to think like an attacker, it does not require you to be an elite hacker to be successful. Because threat modeling is a social exercise, you can rely on the knowledge of your teammates, system administrators, and developers to help you build a full, complete threat model. Collaborative threat modeling with a threat modeling platform built for collaboration is key.

Integrating Threat Modeling into a Secure CDLC

Threat models are useful documentation, though they must also be actionable to provide their full value. Here's a sample workflow that illustrates how threat models can be made actionable:

1. An initial threat model is built.
2. The output of the threat model (mitigations to threats identified) is sent to software that can track issues, such as Jira or Azure DevOps boards, where items are added to a backlog.
3. Mitigations to the identified threats are put in place.
4. IaC templates or new features are deployed through the CI/CD pipeline with the mitigations in place.
5. Any existing gaps are identified, and validation is done to prove the identified threats were mitigated.
6. The status of the mitigated threats and feedback from the process are fed back to the threat modeling platform.

This cycle is not a one-time deal. It repeats continuously as new threats are identified and mitigated, and that feedback is integrated into the threat model. This ongoing process provides documentation for your security architects and CISO about the current security of your systems and applications, whenever they need to reference it.

Looking toward the Future

If I had to think of one word to sum up the future of threat modeling, it would be *automation.* I believe the future of threat modeling involves the extensive use of application programming interfaces (APIs) to automate security modeling and testing as part of the overall development pipeline. Many companies have already made the leap into this automation process, and they are seeing the rewards that come from performing security functions earlier in the development process.

Developers are happier as well, because with automation they can get their testing done and code reviewed in a fraction of the time it used to take with manual testing. If you've ever worked in an Agile environment that does one-to-two-week sprints, you know that developers are typically strapped for time.

In terms of the ThreatModeler platform, as of the time of writing, work is underway to automatically convert cloud threat models to IaC for secure deployment. Threat modeling will drive security in the cloud as well as continuously check and validate the security posture in case services, data stores, data assets, human users, or any other environment entities change.


TIP

Are you ready to see how easy it is to automate threat modeling? Check out AWS Accelerator in Chapter 4 or Azure Accelerator in Chapter 5 and begin your own threat modeling journey.

IN THIS CHAPTER

- Integrating with AWS
- Creating an AWS threat model

Chapter 4
Creating Threat Models for AWS

It isn't surprising that you're interested in threat modeling for AWS. Research by Gartner found AWS had 47.8 percent of the market share for public infrastructure-as-a-service (IaaS). Because AWS leads the pack in IaaS and other cloud services, knowledge of how to secure your cloud applications in AWS is crucial.

AWS forged a joint venture with ThreatModeler and has focused on a number of integrations within AWS to help you build a secure Cloud Development Life Cycle (CDLC). You can start a ThreatModeler subscription from within the AWS Marketplace or contact ThreatModeler directly.

In this chapter, you learn about ThreatModeler's AWS integrations and walk through two threat modeling exercises.

Integrating with AWS

AWS and ThreatModeler have a multitude of integrations available. Some utilize products within ThreatModeler, while others use products found within AWS. In particular, ThreatModeler's Joint

Offering With AWS enables CDLC to automate and accelerate the design of secure AWS cloud environments. This section examines some of these integrations.

ThreatModeler integrations with AWS

ThreatModeler has two offerings to help you with threat modeling in AWS: Architect (patented) and Accelerator (patent pending).

Architect

Architect enables you to automatically create tasks for items that may be missing as you create your architecture diagram. As you drop new items into your architecture diagram, Architect analyzes the components and their relationships and creates tasks for you to examine that are based on architectural rules defined in ThreatModeler's Threat Intelligence Framework. This acts as your onboard security architect, making suggestions as you go along and ensuring security from the outset.

For example, when you add an EC2 instance to your diagram, one of the tasks tells you that you need an AWS VPC subnet for an EC2 instance. As each task is remediated, a check mark appears next to it in the Tasks list so you have an easy-to-follow view of security best practices at your fingertips. As shown in Figure 4-1, the items in the Tasks list are specific to AWS. They aren't simply generic tasks that you have to interpret.

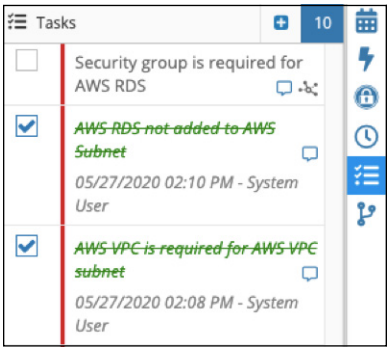

FIGURE 4-1: An example of the Tasks list generated by ThreatModeler Architect as you add items to the architecture diagram.

Accelerator

With Accelerator you can automatically generate a threat model of your current AWS environment. Threat models are kept synced to your AWS environments so you have real-time status feedback in your threat models when changes are made to your AWS infrastructure.

Once the integration to AWS has been set up for AWS Accelerator, AWS Config is utilized to provide data regarding the systems in your AWS cloud. In addition, AWS Security Hub gets updated on your overall compliance status. As shown in Figure 4-2, you have visibility into your entire AWS account. You can tell at a glance if your threat models are synced and up to date.

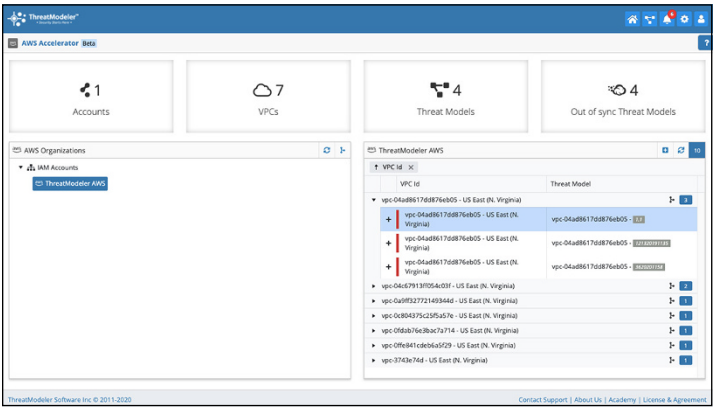

FIGURE 4-2: The Accelerator Dashboard shows, at a glance, the status of discovered assets and synchronization.

AWS products that integrate with ThreatModeler

ThreatModeler takes advantage of several products within AWS to produce threat models. This section walks you through a few of the core services.

AWS Config

Accelerator utilizes AWS Config to retrieve current configurations to build threat models. Additionally, rules in AWS Config are leveraged to keep the threat model up to date. You can use either pre-defined rules or custom rules to set your configuration.

AWS Config is a powerful configuration management tool. If you aren't familiar with AWS Config and its capabilities, you can learn more about it by visiting `https://aws.amazon.com/config/`.

AWS Organizations

AWS Organizations offers an area to centrally manage security, compliance, billing, and access across multiple AWS accounts. You can use organizational units to organize different AWS accounts. For example, you might have organizational units (OUs) for Prod and Test, or different applications.

The accounts and OUs you set up in AWS Organizations are visible in the Threat Modeler Dashboard. When you select an account, you see the related threat models that exist within that account.

AWS Security Hub

If you need to monitor compliance or how well you measure up to industry standards, AWS Security Hub is the place to do it. The 42 compliance rules in AWS Security Hub align to the CIS AWS Foundation benchmark. If you are using the CIS report, you will receive either a compliant or non-compliant status. These reported statuses can be made visible with ThreatModeler. If you are using Accelerator, ThreatModeler can also update the compliance status in AWS Security Hub. If you use Threat Modeler to update AWS Security Hub, these are the statuses you may see:

- Compliant
- Non-compliant
- Open
- Recommended
- Unable to test

Exercise: Creating an AWS Threat Model

Now that you know about the types of integrations available within AWS, this section walks you through building a threat model. You get to use Accelerator first so you can see how it simplifies the process. Then I walk you through creating a threat model manually using the ThreatModeler Toolbox and Architect.

Using ThreatModeler Accelerator

Follow the steps in this section to create an AWS Accelerator-enabled threat model.

TIP

The directions in this section assume that you have set up the integration within AWS.

First, go to the ThreatModeler Accelerator Dashboard. Then proceed as follows:

1. Log into ThreatModeler.
2. Click Settings and choose Accelerator.
3. Under the AWS Organizations pane, select the account in which you want to build the threat model.
4. Click the plus sign next to the virtual private cloud (VPC) in which you want to create a threat model. This is in the VPC pane in the bottom-right corner of the screen.
5. Set the priority of the threat model.
6. If the threat model is for an internal system, check the Internal check box. Otherwise, leave it unchecked.
7. Click Save.

That's all there is to it. Within minutes, you'll have an architecture diagram based on your current AWS infrastructure. You'll also have a list of tasks to examine based on what was found, including items that need to be accomplished in order to complete the architectural threat model. Pretty painless, isn't it?

Creating a threat model manually with Architect

When you manually create a threat model, the ThreatModeler Toolbox is your best friend. It provides AWS product and service components out-of-the-box that you can drag onto your architecture diagram. As soon as you drag an icon from the Toolbox to the architecture diagram, Architect creates a task list for you of the things that your diagram needs to complete. Some of them are foundational network concepts — for example, maybe the EC2 instance should be inside a subnet, inside a VPC. Other tasks are security best practices. These should be checked off as you implement them.

Figure 4-3 gives you your first glimpse at the Toolbox for AWS. One of my favorite things about the Toolbox is that most of the icons in the Toolbox are specific to AWS products and services. A few of the icons are generic, and these represent web interfaces, web applications, users, and so on.

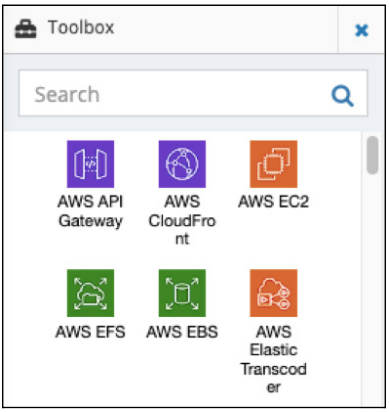

FIGURE 4-3: With the Toolbox in ThreatModeler, you can easily drag and drop AWS service components into an architecture diagram.

In this exercise you build a threat model and see how Architect generates a task list for you. Check out Figure 4-4 to get an idea of what you will be building.

Begin with a blank architecture diagram in ThreatModeler and then follow these steps:

1. From the Toolbox, search for Users and drag that icon to the architecture diagram.
2. Drag and drop the Web Interface icon onto the architecture diagram.
3. Search for AWS ALB and drop that icon onto the architecture diagram.
4. Click in the empty space on the diagram, then click Group in the toolbar.
5. Right-click on the text in the box that appears, then select Container.

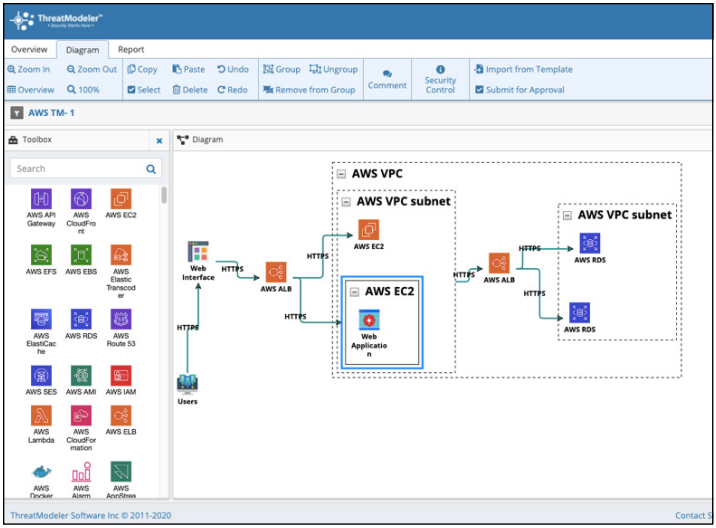

FIGURE 4-4: A sample architecture diagram using several components.

6. In the pop-up menu, select AWS VPC, then click Save.
7. Re-size the VPC component so that it is similar to the one shown in Figure 4-4.
8. Click inside the AWS VPC and select Group in the toolbar.
9. Right-click the text in the box that appears, then select Container.
10. In the pop-up menu, select AWS VPC Subnet, then click Save.
11. Repeat steps 8-10 to create a second subnet.
12. Position the subnets similarly to those in Figure 4-4.
13. From the Toolbox, drag an AWS EC2 icon into the left AWS VPC subnet toward the top.
14. Click below the icon, still within the same AWS VPC subnet.
15. Click Group and then Container.
16. Choose AWS EC2, then click Save.
17. From the Toolbox, search for Web Application. Drag the icon into the AWS EC2 container you just created.
18. From the Toolbox, drag an AWS ALB icon in between the two subnets.

19. For the last of the components, drag AWS RDS from the Toolbox into the second AWS VPC subnet. Repeat so that you have two of them. Arrange them similarly to Figure 4-4.

20. For each object in the architecture diagram, you can click a source object, then drag and drop an arrow line from the source object to the destination object. This documents communication between the objects with the default protocol that would typically be used. Do this for all the objects in the diagram.

To select a protocol other than the default, or to select multiple protocols, right-click the arrow line you created and select from the drop-down list.

That's all there is to it! You've built your first threat model in AWS, and now have a list of tasks that you need to complete in order to improve your threat model.

IN THIS CHAPTER

- Integrating with Azure
- Building an Azure threat model

Chapter 5
Building Threat Models in Azure

Owing to the large number of organizations moving their systems to Azure, the service has shown steady growth and is second only to AWS in market share. According to research by Gartner, Azure accounts for 15.5 percent of all public infrastructure-as-a-service (IaaS) systems. It's no surprise that you would want the ability to create threat models to ensure that you're protecting your Azure assets properly.

In this chapter, you learn about ThreatModeler's Azure integrations and walk through two threat modeling exercises.

Integrating with Azure

ThreatModeler has worked in conjunction with Azure to create multiple integrations that allow you to quickly build threat models and get task lists that define the needed next steps.

ThreatModeler integrations with Azure

ThreatModeler offers two integrations to help you with threat modeling for Azure: Architect (patented) and Accelerator (patent pending).

ThreatModeler Architect for Azure

When creating threat models manually for Azure, you need to identify outstanding items to be represented. These may be foundational elements like storage accounts or virtual networks, or they may be security needs like network security groups. As shown in Figure 5-1, Architect builds your task list as you create an architectural diagram. These tasks represent best practices and are specific to Azure products. There are no generic tasks! As you complete tasks, they are checked off in the Tasks list.

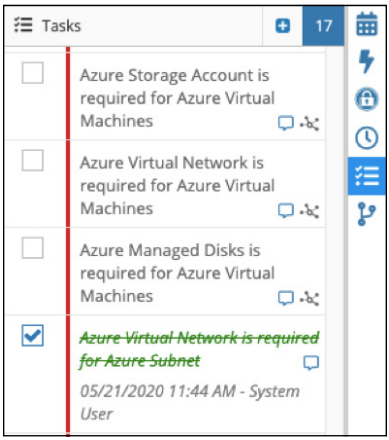

FIGURE 5-1: An example of the Tasks list generated by ThreatModeler Architect for Azure as you add items to the architecture diagram.

ThreatModeler Accelerator for Azure

If you're like most security architects and your time is at a premium, you'll love Azure Accelerator. With Azure Accelerator, you can automatically create complete threat models with task lists. You can also synchronize the data in the threat model so that as changes are made in Azure, the threat model is automatically updated. Figure 5-2 gives you a glimpse of the Azure Accelerator Dashboard. In a single "pane of glass," you'll see your Azure accounts and the synchronization status of your threat models.

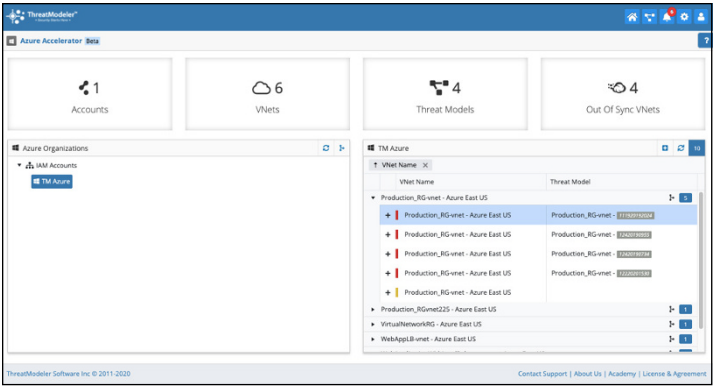

FIGURE 5-2: The Accelerator Dashboard shows you, at a glance, the status of discovered assets and synchronization.

Azure products that integrate with ThreatModeler

ThreatModeler utilizes some of Azure's services to help you with threat modeling. This section looks at one of the core services.

Azure Portal

ThreatModeler integrates with the Azure Portal for a variety of functions, including the access needed to generate threat models, as well as access to Azure Security Center and other resources.

Azure Pipelines

Azure Pipelines is the Azure offering that allows you to use continuous integration and continuous delivery (CI/CD) to test, build, and release code more often. ThreatModeler integrates with Azure Pipelines using a bi-directional API that allows threat modeling to become a part of your CI/CD process. Anytime a change is made, that change can be reflected in your threat model.

Azure Boards

Azure Boards is a component of Azure DevOps that allows you to track and manage all of the work for your software development projects. It supports Scrum and Kanban style boards and

offers customizable options. ThreatModeler integrates with Azure Boards to create work items based on threats identified during the threat modeling process so that you can track all your work in one place.

Building an Azure Threat Model Exercise

Now that you're more familiar with how Azure and ThreatModeler integrate, it's time to build your first threat model with ThreatModeler. In this exercise, you first use Azure Accelerator to sample just how easy creating a threat model is. Then you create a manual threat model using the Toolbox and Architect.

Using ThreatModeler Accelerator for Azure

Accelerator helps you quickly create threat models and hit the ground running. In this exercise, see for yourself by creating a threat model with ThreatModeler Accelerator for Azure.

TIP

The directions in this section assume that you have set up the integration within the Azure Portal. ThreatModeler requires read-only permissions for Accelerator to work its magic.

First, go to the Accelerator Dashboard in ThreatModeler. Then follow these steps:

1. Log into ThreatModeler.
2. Click Settings and choose Accelerator for Azure.
3. Under the Azure Organizations pane, select the account where you want to build the threat model.
4. Click the plus sign next to the Vnet Name where you want to create a threat model. This is in the pane in the bottom-right corner of the screen.
5. Set the priority of the threat model.
6. If the threat model is for an internal system, check the Internal check box. Otherwise, leave it unchecked.
7. Click Save.

That's all there is to it. Within minutes, you'll have an architecture diagram generated from what you currently have in Azure. You'll also have a list of tasks to examine based on what was found.

Creating a threat model manually with ThreatModeler Architect for Azure

Sometimes creating a threat model manually is more desirable than creating it automatically. In those cases, ThreatModeler still has your back. Azure Architect keeps threat modeling simple and quick by automatically generating task lists for you.

Figure 5-3 shows ThreatModeler's Toolbox for Azure. All the products and services are specific to Azure. You won't see other cloud service provider products in the list.

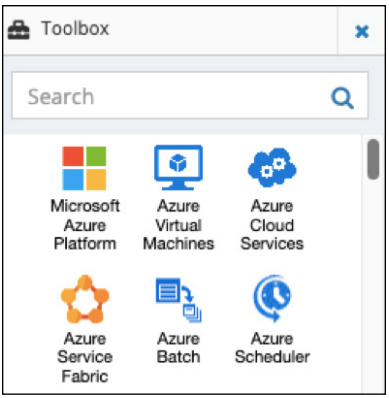

FIGURE 5-3: Use the Toolbox in ThreatModeler to easily drag and drop Azure services into an architecture diagram.

Have some fun and dig into the next Azure example! Check out Figure 5-4 to see what you will be building.

Follow these steps to manually create threat model for Azure:

1. From the Toolbox, drag and drop Azure Active Directory into the top left of the architecture diagram.
2. From the Toolbox, search for Internet, then drag and drop the icon under Azure Active Directory.

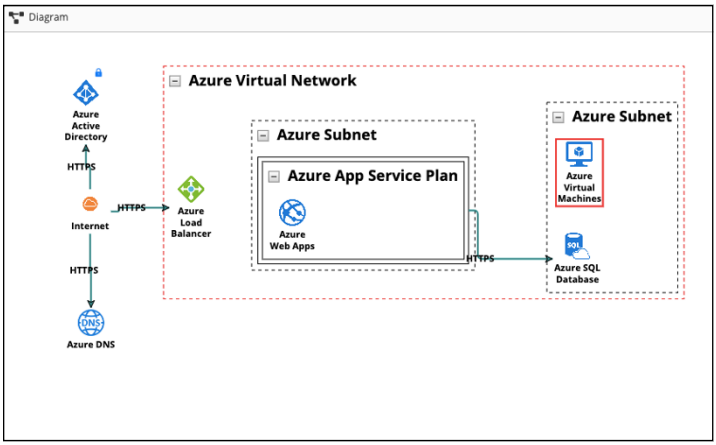

FIGURE 5-4: A sample architecture diagram using several components.

3. From the Toolbox, drag and drop Azure DNS under Internet in the architecture diagram.
4. Click in the empty space on the diagram, then click Group in the toolbar.
5. Right-click the text in the box that appears, then select Container.
6. In the pop-up menu, select Azure Virtual Network, then click Save.
7. Re-size the Azure Virtual Network component so that it is similar to the one in Figure 5-4.
8. From the Toolbox, drag and drop an Azure Load Balancer into the Azure Virtual Network. Place it similarly to the one in Figure 5-4.
9. Click the empty space next to the load balancer, then click Group in the toolbar.
10. Right-click the text in the box that appears and choose Container.
11. In the pop-up menu, select Azure Subnet, then click Save.
12. Click the empty space next to the Azure Subnet you just created, then click Group in the toolbar.
13. Right-click the text in the box that appears and choose Container.
14. In the pop-up menu, select Azure Subnet, then click Save.

15. Resize and arrange the subnets similarly to the way they're arranged in Figure 5-4.
16. Click inside the Azure Subnet in the middle of the architecture diagram, then click Group in the tool bar.
17. Right-click the text in the box that appears and choose Container.
18. In the pop-up menu, choose Azure App Service Plan, then click Save.
19. From the Toolbox, drag and drop the icon for Azure Web Apps into the container you just created for Azure App Service Plan.
20. From the Toolbox, drag and drop the icon for Azure Virtual Machines into the second subnet container toward the top.
21. From the Toolbox, drag and drop the icon for Azure SQL Database into the second subnet container toward the bottom.
22. For each object in the architecture diagram, you can click a source object and then drag and drop an arrow line from the source object to the destination object. This documents communication between the objects with the default protocol that would typically be used. Do this for all the objects in the diagram.

To select a protocol other than the default, or to select multiple protocols, right-click the arrow line you created and select from the drop-down list.

There you have it! A threat model for Azure built from scratch in minutes. Do you want to know the best part? A task list is waiting for your review so you can ensure your application is meeting Azure best practices for a complete architecture threat model diagram.

IN THIS CHAPTER

- Making better-educated security decisions
- Continuously validating the implementation of security controls
- Securing systems proactively with threat modeling

Chapter 6
Ten Ways Threat Modeling Reduces Time and Cost of Security

This book is all about threat modeling, how it can fit into your DevSecOps environment, and how ThreatModeler can help you achieve a secure Cloud Development Life Cycle (CDLC). I now have ten points to impart to you, my fabulous reader. These points explain how threat modeling, when done proactively, can reduce the time and cost associated with ensuring that your CDLC is secure.

Threat Modeling Is Core to the Concept of Security by Design.

By engineering security into the design phase of any system, application, or process, you can make better-educated security decisions and communicate them, in addition to helping your organization achieve compliance with policies or regulatory frameworks. ThreatModeler simplifies creating new threat

models with an easy-to use interface and automation in Azure, AWS, (and soon, Google Cloud). The best part is that by moving security earlier into the design process, you will end up with a more secure product and save money because you won't have to re-engineer for security later.

Threat Modeling Should Be Done Early in the Planning/Design Stage.

To fully understand the threats to a system, process, or application, you must do threat modeling early in the CDLC to ensure that threats are identified and mitigated appropriately. When you do threat modeling earlier, security issues are less expensive to resolve and less likely to delay a project.

Because ThreatModeler integrates with major cloud providers with features such as Accelerator (patent pending), threat modeling in the cloud is quick and efficient. ThreatModeler Accelerator interrogates your cloud architecture and automatically creates the architecture diagram while checking and validating the security posture for you. Accelerator also checks for "drift" in case any changes were made to the cloud architecture without the security team's knowledge, then takes steps to mitigate new threats.

Threat Modeling Helps You to Visualize and Understand Your Complete Attack Surface.

If you understand the assets in your environment, plus the entry points of your system and how an attacker might get in, you have a better idea of how to protect those assets. Process flow diagrams (PFDs) like those used by ThreatModeler enable visualization of attacks in an intuitive user interface. Therefore, threat modelers can think like a hacker to understand the tactics involved in compromising data and resources. Another benefit of PFDs is developers tend to think in terms of functions and features (processes), which means they can easily be involved in the threat modeling process, making security truly part of DevOps.

An Architecture Diagram Is an Input to a Threat Model.

The architecture diagram is not in itself the threat model; it is one very important input to the threat model. Outputs of a threat model include threats and the security mitigations that can be used to protect against those threats. ThreatModeler automatically converts your architecture diagram into a threat model and generates a list of threats and security mitigations, which, if properly addressed, will reduce the vulnerabilities later on.

Secure Design Patterns Should Be Reused to Reduce the Overall Attack Surface.

Reduce work by reusing design patterns. Many applications and services within an organization have "architectural patterns" — in other words, repeating components. A great example is an identity platform that gives users access to multiple applications. In ThreatModeler, threat model the identity platform once, create a template from it, and reuse it as needed. This practice allows you to use "known good" security controls to protect your assets and gain consistency.

Going further, ThreatModeler enables you to convert entire threat models into components for use in future threat models. This patented practice is known as *threat model chaining* and is ideal for reusing previous threat models that may be connected in some way. None of your previous hard work goes to waste with ThreatModeler, which is in stark contrast to the past.

A Good Threat Modeling Process Will Have Different Outputs for Different Stakeholders.

Threat models should provide meaningful output for different stakeholders. Engineers, for example, want technical outputs like security controls; developers want to know what security requirements they must meet; and board members want a higher level,

non-technical output that allows them to understand their security posture and make good business decisions. ThreatModeler has robust reporting mechanisms that provide relevant outputs to stakeholders, such as compliance teams, CISOs (viewable via dashboards), security architects, red teams and pentesters, developers, and risk management and compliance teams.

The Output of a Threat Model Should Be Pushed to CI/CD Tools.

By sending the output of the threat model to an application life cycle management (ALM) tool like Jira, Azure DevOps, or ServiceNow, you can track the mitigation of the identified threat during the development process. ThreatModeler natively integrates with Jira, Azure DevOps, and ServiceNow.

Going further, ThreatModeler comes out-of-the-box with an extensive API perfect for integrations with any — or all — upstream and downstream tooling in your CI/CD pipeline. One ethos of ThreatModeler is to set your data free so you can extract it for true pipeline integration, which is the heartbeat of the security automation drive.

Security Controls Reduce Duplication of Effort.

ThreatModeler is unique in including Security Control components within the architectural diagram. When you add the security control to the architecture diagram, ThreatModeler shows the remediated threats as crossed out. You can see at a glance which threats are outstanding and which are mitigated. A perfect example may be an API gateway, which itself remediates many potential threats. If you add this to your architectural diagram, you want threats thwarted by the API gateway to be automatically highlighted and closed as remediated. You don't want to spend time and effort mitigating a threat that has already been dealt with — that would be a duplication of effort. This is why ThreatModeler has the concept of Security Control components.

A Threat Model Is a Living, Breathing Document.

Threat models are not point-in-time exercises. They are living documents that must be routinely revisited. As your systems, processes, or applications evolve, so too must your threat model evolve. The life span of any threat model should be the same as the life span of the systems, processes, or applications modeled. With ThreatModeler's integrations with Azure and AWS, architecture diagrams can be updated quickly by re-synching with the cloud architecture.

ThreatModeler's versioning functionality is also a powerful mechanism for ensuring that threat models live and evolve with your applications and services. As changes are made to your applications and a new version deployed, a new version of the threat model is created, tracking those changes and ensuring that appropriate threat mitigations are in place to account for those changes.

A Scalable Threat Modeling Process Should Be Collaborative and Implemented as a Self-Service Model.

Keeping threat models up to date — or even working with threat models in general — shouldn't require someone with a huge amount of security experience. For threat modeling to be truly scalable, threat models must be available in a self-service capacity. For example, if a developer wants to create an application, who better than the developer to create a threat model of the application? When the developer collaborates with teammates or the security team, the threat model gets even better.

ThreatModeler facilitates cross-team collaboration as well as a self-service model that encourages teams to re-use secure designs as they build their own architecture diagrams. This way, whenever updates are made to an IT environment, threat models can be quickly and accurately updated. Automation is a great way to reduce human error and make certain that re-used design patterns consistently ensure security.

Glossary

architecture diagram: A graphical representation of data flow or process flow in an application.

asset-centric: An approach to threat modeling that focuses on identifying assets, what the asset does in your environment, and the asset's criticality within the environment.

attack-centric: An approach to threat modeling that relies on identifying attacks and creating attack trees.

AWS Accelerator: A ThreatModeler product (patent pending) that automatically generates threat models for AWS VPCs.

AWS Architect: A patented ThreatModeler product that automatically creates AWS-specific task lists from an architecture diagram.

Azure Accelerator: A ThreatModeler product (patent pending) that automatically generates threat models for Azure Virtual Networks.

Azure Architect: A patented ThreatModeler product that automatically creates Azure-specific task lists from an architecture diagram.

Cloud Development Life Cycle (CDLC): A life cycle model that adapts quickly to change and is optimized for scaling resources and applications in the cloud.

component: An object in an architecture diagram that can be a system, application, or process.

data classification: The process of organizing data by tagging it with its level of sensitive content.

data flow diagram (DFD): Used to document the flow of data in, out, and around a system, application, or process.

decomposing an application: To break an application into smaller components that can be examined individually rather than as a whole.

DevSecOps: Introduces security earlier in the development process (planning and design) to minimize vulnerabilities and cost.

Drift: When an operations person adds or removes something from the cloud environment without the security team being aware, ThreatModeler's Drift feature detects these changes and highlights them in the diagram for appropriate action to be taken if new threats are identified.

Infrastructure as Code (IaC): A method that allows the creation of systems from templates, typically written in JSON.

inputs: Components used to create a threat model, such as architecture diagrams, decomposing the application, classifying data, and identifying use cases.

Microsoft Threat Modeling: A software-centric approach to threat modeling that requires knowledgeable security personnel to implement.

OCTAVE: An asset-centric approach to threat modeling that focuses on business risk and does not scale well for large organizations.

outputs: The items you can expect to get out of a complete threat model, including identified threats, security requirements, and test cases.

Process for Attack Simulation & Threat Analysis (PASTA): An attack-centric approach to threat modeling that focuses on the view of a would-be attacker based on identified threats. This method is complex and requires training.

personally identifiable information (PII): Any data that might be used to identify a person.

process flow diagram (PFD): Used to visually demonstrate the relationships between processes. Allows a view similar to that of an attacker.

security blocker: A security issue severe enough that it must be resolved before a process or deployment can continue.

security debt: Historical security configurations or decisions that were meant to be short-term and still must be fixed. These are often put into a backlog.

security posture: An organization's overall security status, taking into account hardware, software, and data as well as the organization's ability to recover from an adverse event.

security requirement: A security feature or function that is required by the organization in order for an application or process to be considered secure.

software-centric: An approach to threat modeling that examines application design and the flow of data through an application.

STRIDE: Used in Microsoft Threat Modeling to categorize threats. Stands for spoofing, tampering, repudiation, information disclosure, denial of service, and escalation of privilege.

threat: An event or person that might negatively affect a system or application.

threat nesting: Also known as *threat chaining,* ThreatModeler's patented feature allows you to reference a threat model within another threat model at scale.

threat model: Gives you visibility into the threats in your organization, and allows you to assess impact and remediations for those threats.

ThreatModeler Cloud: An offering from ThreatModeler that enables you to automatically generate threat models for cloud workloads.

Trike: An asset-centric approach to threat modeling that places an emphasis on cybersecurity risk management.

trust boundary: A logical boundary that defines a change in trust, such as internal systems (private) and DMZ systems (public).

VAST: A software-centric approach to threat modeling that works well with organizations using Agile/DevOps models. Stands for Visual, Agile, Simple Threat modeling.

vulnerability: A weakness that allows a threat to successfully exploit an application or process.

Securely Design, Build, Deploy, and Manage Cloud Environments

Only ThreatModeler Automatically Identifies and Mitigates Cloud-Based Threats and Validates Secure Configuration

Over 12 years of innovation in partnership with 100+ Fortune 1000 companies spanning multiple verticals to secure highly sensitive, regulated, complex, specialized, and global environments.

Cloud- and AppSec-Ready Threat Modeling With Sophisticated Diagramming Capabilities

One-Click Execution, ThreatModeler Does It All For You

ARCHITECT (Patented) Take an AWS or Azure component (Google Cloud coming soon) and the onboard Architect guides you to complete an accurate threat model.

ACCELERATOR (Patent Pending) Automatically interrogate VPCs and build a threat model with threats and security requirements. Instantly rebuild the model based on architecture changes.

THREAT MODEL CHAINING (Patented) Nest an entire threat model within another threat model for a holistic macro view of the interconnected system as components interact with each other.

THIRD-PARTY FILE IMPORTS (Patented) Generate threat models that correlate with visual diagram components of third-party software applications, including VISIO and Microsoft TMT.

ENTERPRISE READY Role-based access control, built-in customizable reporting, export capabilities, identity provider integration, and audit trail at scale.

www.threatmodeler.com

Create threat models for AppSec and the cloud

As organizations increasingly move to cloud-based systems, security architects who are used to creating threat models for traditional on-premises data centers must evolve their skill sets to create threat models for the new cloud environment. They must understand the service offerings to comprehend the added risks. They must be up to date on where responsibilities for cloud security lie, and they must understand what hosting information systems in a multi-tenant environment entails. This book is your guide to threat modeling in the cloud and securing your assets in this challenging new world.

Inside...

- Understand threat modeling methods
- Automate, collaborate, and integrate at scale with process flow diagrams
- Threat model in a DevSecOps setting
- Secure your Cloud Development Life Cycle
- Integrate with AWS and Azure services
- Explore threat modeling use cases

Sara Perrott has an MS in cybersecurity and information assurance and holds several industry certifications such as CISSP and GCIH. She works full-time as a security engineer and teaches part-time.

ISBN: 978-1-119-74324-8
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.