Wolcott Wheeler wolcottwheeler@gmail.com

1 Riverside Drive, #24 (914) 826-5188

Sleepy Hollow, NY 10591 July 22, 2016

**Engineering a Scalable Threat Model in Seven Easy Steps: A White Paper**

1. **INTRODUCTION: ONCE MORE INTO THE BREACH**

Have you been hacked yet? If not, consider yourself lucky. Widespread and thoroughgoing security breaches have become a major danger for organizations around the world—threatening financial stability, damaging customer confidence, and triggering massive regulatory fines.

Because of this growing danger, more and more organizations are turning to threat modeling—a process that assesses the probability of attacks and their potential harm and ranks them according to the potential for inflicting damage. This way, threats can be eradicated or minimized. Microsoft has made threat modeling a key part of its Security Development Lifecycle (SDL) process.

Previously, threat modeling was based on an *adversarial* perspective, using penetration testing and code review. The problem is, these techniques can only be employed after the software has been written. As a result, if vulnerabilities are detected, the software program has to be rewritten. According to the National Institute of Standards and Technology (NIST), if you fix code fixes after release, it can result in **30 times** the cost of fixes performed during the design phase.



In contrast, *defensive* threat modeling assumes the defender’s point of view. Before any code is written, threats are analyzed, and coders build appropriate protection into the software.

In this white paper, we’ll go into exactly how threat modeling works and what you can do to safeguard your software from attackers.

**The Role of Threat Modeling**

Threat modeling identifies threats prior to software launch and classifies those threats according to their potential for risk. As a result of this analysis, you can prioritize your mitigation efforts to reduce the risk to your web application.

Threat modeling identifies:

* potential threats to your system
* your data and asset exposure
* logical and architectural vulnerabilities
* relevant security controls.

For a strategic point of view, threat modeling helps you evaluate the security decisions you need to take to safeguard your program, and it acts as a guide for security testing. The end result? Threat modeling helps minimize risk exposure.

Threat modeling analyzes software from a granular level: you build applications from individual features. Since each feature can be attacked, it’s vital that you understand the components of the application and understand how each can be attacked. Below is an example of a basic login screen. See the all potential vulnerabilities that it offers?



**Why Employ Threat Modeling?**

There are several powerful reasons why your organization should engineer threat modeling into its software.

***New threats are always emerging.***

As a result, it just makes sense to constantly update your threat library as you identify new threats. With threat modeling, you can build an Attacker Profile that includes the attacker’s skillset and motivation. Threat modeling enables you to classify threats by risk to prioritize mitigation.

***There’s a tremendous cost benefit.***

Correcting a vulnerability defect could cost **30 times more** if that defect isn’t detected until the program reaches the production stage.

***Scanning tools are insufficient***.

Vulnerability scanners don’t identify vulnerabilities; they merely verify a given application’s current state of security. In contrast, threat modeling actually identifies vulnerabilities. Moreover, scanners produce:

* false positives
* false negatives; and
* a false sense of security.

By the time a scanner detects a defect, it’s too late in the game; the software now has to be corrected.

***Compliance Considerations***

By building security into your Software Development Life Cycle (SDLC), you’re protecting your company and its end-users from intrusion and data theft—and thereby avoiding reputational damage, major fines, and loss of public confidence.

**The Top Five Reasons to Leverage Threat Modeling to Mitigate Risk**

With threat modeling, you can:

1. Allow security and development teams to **pinpoint high-value targets (assets) and data exposure** early in the design phase, before applications are moved to production.
2. Promote **the use of secure code**, enforcing standards across your organization.
3. Enable pen testers to focus on **the most critical entry points** in applications.
4. Generate reports and checklists to **validate that proper security controls** are in place to meet compliance objectives.
5. Identify threats in applications, classify them by risk, and **predict the business and technical impact** of a successful attack on your organization.

**What Are the Components of Threat Modeling?**

Threat modeling consists of three key components: methodology, key stakeholders, and actionable output.

* Its **methodology** can draw from the best practices of the Microsoft, PASTA, and PTA threat modeling approaches.
* **Key stakeholders** in the process of threat modeling include security experts; developers; information architects; product managers; and executives.
* What is the **actionable output** that threat modeling yields? After compiling a list of threats, it takes into account each danger’s threat score and uses it to assess that threat’s level of risk. In other words: ***List of Threats + Threat Score = Risk***

The diagram below illustrates the progression of the threat modeling process:



**Measuring the Effectiveness of Threat Modeling**

A given threat model works if it is:

* practical
* understandable by non-security experts
* comprehensive in terms of its threat coverage
* reusable and repeatable
* consistent and yields actionable output
* automated
* scalable.
1. **THE SEVEN EASY STEPS TO BUILDING YOUR OWN SCALABLE THREAT MODEL**

As you’ll see below, threat modeling is vital for organizations today because it’s both cost-efficient and powerful; this is because it’s scalable, and through automation, your organization can ensure software security across an unlimited number of programs or IT operations, without needing to invest capital into each program’s individual security.

Now we get down to the nitty-gritty—what you’ve been waiting for. Here are the seven steps you have to undertake to establish your effective threat model.

***Step 1: Assemble a Threat Library***

What’s the first step in creating a threat model? Know your enemy. First you compile a comprehensive threat library by drawing on existing threat libraries—such as those available from CAPEC, WASC and OWASP—and identify custom threats and organization-specific threats that you’re particular enterprise is likely to be attacked by.

Our product ThreatModeler provides protection against 750 threats as of this writing, and we’re adding more all the time.

A threat library, based your repository of threat data, will promote consistent and thorough security standards throughout your organization. Use a template to define the properties of a threat, which may include:

* details on how the threat can be executed
* security controls that can effectively mitigate the threat
* profiles of attackers that could carry out this threat, including their motivation and skillset
* the impact of the threat on your organization, based on real-time breach data in terms of technical and business impact.

The screenshot from ThreatModeler below illustrates how you can build a threat library with various attributes defined for a threat.



To prioritize your mitigation efforts, use a risk rating structure to rank the importance of your threats by asking, “Which are the most risky threats in terms of the danger they pose to my organization?” Associate risk with threats. Based on your organization's risk rating policy, classify threats according to risk in terms of business and technical costs to the organization. By classifying threats according to risk, you can prioritize the mitigation efforts you can budget and focus on high-risk threats, vulnerabilities, and software components to ensure robust and secure code.

You can easily build a comprehensive threat library by using reputable and industry-adopted resources such as threat libraries from MITRE’s CAPEC, WASC-TC and OWASP, as well as internal research data on threats. You can also draw on commercial threat databases and the work of external research teams.

By using real-time breach data from sources such as the DataLossDb or the Web Hacking Incident Database (WHID), you can gauge a threat’s impact on your organization based on statistical evidence of breaches against other organizations in your same industry.

When assembling your threat library, remember: forewarned is forearmed. If you first consider what’s liable to attack you, you’ll be prepared for the sneak attack when it comes.

Our product ThreatModeler not only helps you categorize threats by risk intensity—it offers a feature known as Threat Intelligence that creates a list of threats applicable to a particular application.

***Step 2: Institute Security Controls***

Okay, now you’ve identified your threats via careful analysis. Now it’s time to mitigate them so you can nullify their danger.

After assembling your threat library, construct a library of security controls and associate them to the threats you’ve identified. A security control can be defined as one of the following:

• Secure development guidelines and code snippets

• Code review guidelines

• Test cases

• Secure configuration best practices

• Compliance activities

Below you can see how you can define a security control to promote secure coding standards, thereby effectively mitigating a threat.



You can identify mitigation steps by adopting the secure coding standards offered by the OWASP Secure Coding Quick Reference Guide and the OWASP Developers Guide.

Next, you adopt security frameworks that have been established by the OWASP ESAPI; the Microsoft Enterprise Library; and Microsoft AntiXSS Library. And then, since you’re going to be dealing with threats unique to your organization and situation, you can develop your own custom and homegrown security framework.

Here’s some detail on these guidelines:

**Secure Development Standards**

• OWASP Secure Coding Quick Reference Guide: A technology-agnostic set of general software security coding practices in a comprehensive checklist format that you can integrate into the development lifecycle.

• OWASP Developer Guide: Covers all forms of web application security issues.

**Security Enhanced API**

• OWASP ESAPI – ESAPI: The OWASP Enterprise Security API is a free open-source web application security control library that makes it easier for programmers to write lower-risk applications.

• Microsoft Enterprise Library: A collection of reusable software components (application blocks) designed to assist software developers with common enterprise development challenges.

• Microsoft Anti-XSS Library: Helps you to protect your current applications from cross-site scripting attacks, while also enabling you to protect your legacy application with its Security Runtime Engine.

• Suhosin: An advanced protection system for PHP installations designed to protect servers and users from known and unknown flaws in PHP applications and the PHP core.

• Your custom and homegrown library: Your unique requirements will dictate the contents.

Our product ThreatModeler comes bundled with secure coding standards.

Finally, link your mitigation steps to your threats. Customize the way you’re going to deal with threats according to their special characteristics.

***Step 3: Use the Information Intelligence You’ve Gathered***

Now that you’ve developed your threat library and established your security framework, you can make use of them to fight threats as they arise. By analyzing your data, you can build a library of reusable threat patterns and attack trees for each component of your software. Using a centralized library, you can impose the policy of “Write Once, Use Anywhere”—so that updates will ripple through each component automatically, and each component will be protected in kind from the updated threat countermeasure. This also acts as an abuse case for developers and Quality Assurance (QA) personnel so that they can measure potential harm to the system.

To improve reusability across the applications in your organization, you need to build two libraries:

• a reusable component library

• an attack tree library

Below is an illustration of a reusable component library entry:



Below is a sample attack tree that maps how a threat can ripple through a system:



Analyze your components—your software features or infrastructure components such as the web server, database server, and so forth. Associate possible threats to this component based on its features. For instance, if a component accepts user input, associate all injection-related threats to it, and associate rules for a component based on your organization’s security policy. As an example, your password policy for authentication might have a certain number of required characters and certain types of characters that can be used. Take these security requirements into consideration and map them to a component.

Let’s take a registration page as another example. You can identify the many vulnerabilities it offers via the many user-provided input fields by considering the interaction with the database in the backend. Sensitive data entered in this page can include:

* password details
* a secret answer to a secret question requirement; and
* possibly Social Security numbers, which U.S. government job portals require.

Here is where the beauty of “Write Once, Use Anywhere” comes in. For every application across your organization, you use a similar registration component. By including this component in your central repository of components with its corresponding threat pattern, you can easily change this component scale across all applications that use it in their design.

***Step 4: Ensure Actionable Output***

So that your new threat model actually does its job of battling attackers, you next have to engineer its actionable output—the fact that it will respond quickly and effectively to attackers.

To enable you to correlate the information gathered in your libraries into a useful, actionable output, you have to understand every detail of your system and applications to be built, then break them down into individual components while considering their technical and business aspects.

Below is a threat model generated by ThreatModeler for a simple e-commerce application:



First, build a comprehensive threat profile, or threat map, of the system you can use to understand the system and generate actionable output. Your threat map will enable you to identify:

* High-value targets: You can identify these by data type, such as sensitive information, or by services, such as the application’s usability.
* Threat scoring: Using well-known scoring metrics such as CWSS and CVSS, use threat scoring to gauge the gravity the threat presents to your organization, and plan mitigation accordingly.
* Data flow and exposure: Based on the type of data being protected, institute controls on the communication medium and impose validation checks for data integrity and safety.
* Negative impact to compliance: List the compliance objectives you need to achieve and the status of your activities under these objectives. Consider instances where breaches would seriously damage your organization’s compliance requirements.
* Security requirements: Assess measures you must have to safeguard your unique system from attackers.
* Abuse cases: Identify instances where a complete interaction between your system and a factor can damage your system.
* Threats to individual components: Detect which components might be in danger of a higher probability of an attack, such as a login component or a component dealing with credit card information. By analyzing these components, you’ll be able to associate all possible threats to them.
* Risk: Based on your organization’s risk management plan, deploy appropriate countermeasures against risk associated with the application as a whole based on a breach and against risk threats to individual elements, according to regulatory compliance requirements.
* Attack trees: Employing your threat pattern and component libraries, build attack trees for each component.
* Mitigation to threat mapping: Associate mitigation steps to threats.
* Security assessment checklists: Draw up a comprehensive checklist to review your application’s security and its adherence to company policy, compliance requirements, and industry standards.
* Vulnerability report: Based on your review, list the number of vulnerabilities left open.

***Step 5: Create a Dashboard and Ensure Reporting***

At this point, you want to start monitoring the effectiveness and efficiency of your threat model. You do this by fashioning an online dashboard that your stakeholders can access so they can view the report they need based on their role. With the dashboard, you can measure the state of your application security and observe trends in the application security profile.

Your Threat Management Console will generate your Threat Portfolio and a Threat and Vulnerability Management Console to report on the success of your threat management and vulnerability management. This way, you can prioritize your threat mitigation responses and defend yourself from threats effectively.

Next, create a Risk Dashboard that lists threats based on risk and provides an application risk profile to assist you in measuring your risk and plan remediation efforts. Your Risk Dashboard can categorize your:

* top 10 threats and vulnerabilities.
* risk profile; and
* compliance portfolio that details your adherence to compliance requirements.

Our product ThreatModeler comes with a ready-made dashboard that’s easy to use. You can do a comparison across multiple software releases and across multiple applications. Here’s a sample ThreatModeler dashboard:

****

Below is a dashboard available in ThreatModeler that highlights how you can display threat data in a dashboard to help your organization prioritize its mitigation efforts. It lists the total number of threats. The pie charts display the number of open threats based on their risk classification and the total threats, based on the status of security controls being effectively implemented.



The dashboard below lists the top 10 threats to a project, based on the number of times that the vulnerability leading to the possibility of a threat exists in the application’s different software components:



With ThreatModeler, you can categorize threat reports and view what your data exposure is. In addition, ThreatModeler also features an Enterprise Risk Dashboard that categorizes threats targeted to your particular organization.

***Step 6: Automate, Integrate, Collaborate***

Now it’s time to get your new threat model ready for enterprise-wide adoption. You can foster scalability by automating and integrating your threat model with your existing workflow. Since threat modeling concerns building security in the SDLC, promote collaboration among all your stakeholders.

1. Automate your threat modeling process. Through automation, you can correlate data in the different libraries and display this data in your dashboards and reports. By instituting the “Write Once, Use Anywhere” function, you guarantee that each component is updated automatically with threat protection.
2. Integrate your new threat model with your existing software development methodology and development workflow. Because the threat modeling development process uses many enterprise-level tools, you can feed the results from these tools into your threat library to further improve the accuracy of your threat prediction. Below is a model of how integration can work in your organization’s environment:



1. Get various stakeholders in your enterprise to collaborate. Impress on them the importance of adopting the new threat model.
* For architects, provide functional information about the application and architectural risk analysis.
* For developers: implement correct mitigation steps and security standards using abuse cases; institute peer code review.
* For your security team: identify threats and mitigation; inaugurate targeted threat testing; and enable vulnerability management.
* For senior executives: inform them of the threat profiles you’ve assessed and stress the importance of threat modeling in effectuating risk management and ensuring compliance.

Also, with an automated system, you can use your bug tracking system and focus your developer training on identifying the top 10 threats.

Whether you’re currently performing threat modeling manually or using automated tools, the threat modeling process described here allows you to build threat models in hours or days, versus weeks or months. This not only saves you time and resources, but it also **keeps threat models up-to-date automatically**, promotes **consistency**, and gives you crucial **scalability** across your organization.

***Step 7: Operationalize Your Threat Model***

After building a threat model in hours (or days, depending on the size of the application), update your threat models with every software release. Use your new threat model to impose effective risk management. Build reusable templates, and scale your threat model across thousands of applications and your development, QA and
security teams. Remember, you want to build nodes for hundreds or thousands of applications at the same time.

Previously, traditional threat modeling encountered a challenge because there was an issue with scalability, owing to the nature of software development workflows such as AGILE, where web application features are rolled out more often than traditional models, such as the waterfall model. No more. Using the methodology detailed in this white paper, your reusable component library will perfectly complement the AGILE methodology. You can further extend this to build reusable templates, based on the type of application and business functionality it’s linked to.

You will only need make minimal adjustments to these templates to use them across various software development projects, allowing scalability that’s both efficient and effective. This establishes a framework for building secure applications and reduces the high cost of fixing production vulnerabilities. By operationalizing this threat modeling process, you can create thoroughness and consistency, producing more effective risk mitigation.

**3: CONCLUSION**

If you adopt the seven steps listed above, it will enable your organization to implement security touchpoints in the various phases of the SDLC, while remaining methodology- and platform-agnostic. Incorporating security in the SDLC is a proven way to enhance software security. By carefully designing a threat model, your organization can guarantee secure coding best practices, without a great deal of additional training of architects and developers on the necessary security concepts.

There is really no other alternative to software security that is more scalable and cost-effective than threat modeling. Only threat modeling fortifies SDLCs against vulnerabilities and avoids costly post-production patches.

In an age of increasing multi-billion dollar data losses, there’s no question—by employing threat modeling to protect software security, it will give you a competitive edge, and your organization will profit immeasurably.

Please feel free to contact us to learn how ThreatModeler can strengthen your organization and protect it from data breaches.