Threat Modeling – The Cornerstone of SecDevOps

Threat Modeling is a fairly common term in Information Security and Application Security circles. However, widely as its known, little is it understood, a lesser still is it practiced with a high degree of efficiency and effectiveness.

To put it simply, Threat Modeling is the process of 'modeling' different threat scenarios to identify risks to the application and its environment. Once these threats are identified, one typically identifies mitigation actions and strategies for the application. The objective of threat modeling is:

- To identify and prioritize Security Threat Scenarios and Risks to the application environment.
- To identify and prioritize Security controls for the application environment.

While the above explanation for Threat Modeling sounds very useful and highly desirable, few organizations actually create and use Threat Modeling effectively and accurately. For most organizations, Threat Modeling has descended into another "Enterprise-driven" exercise which involves a great deal of documentation, massive ERDs, sequence diagrams and the like, burgeoned into a massive report filled with esoteric statements. For instance, one of our clients is a fast-moving, yet highly motivated Cloud Product company, that prides itself on security. Their latest threat model document is about 150 pages long and is from December 2013. This is neither current, nor effective, making Threat Modeling a token practice in most organization, hardly of any real value.

The obvious question here would be, "Why should we be threat modeling?" and "How do we make it effective?"

It is our firm assertion that Threat Modeling serves as an extremely beneficial practice on multiple fronts. Let's examine some of them:

1. Identifying Threat Scenarios and Risks to your Application gives you a *specific* and *relevant* understanding of the security requirements you need to have in your application. A lot of application security requirements are merely "cut and paste" operations of a standard application security playbook. However, with an effective threat model, you can *derive* security controls that are *relevant* to *your* application and its environment. This can be prioritized based on severity, etc that gives you a focused set of application security requirements that you can chase and achieve. This is invaluable in the long term.
2. An Effective Threat Modeling gives you Attack Modeling and SAST Inputs. When you create an effective Threat Model, your security testing team/penetration testers can identify the specific attacks and tests they need to be running to get to the heart of your application's security issues. A lot of times, penetration testers run into relevancy issues, where their results are either too trivial or have

no real business impact. Testers can focus on identifying the issues that are most relevant and specific to your application. The same holds good for SAST Inputs. For instance, if one of the key threats is that attackers may be able to steal your user passwords and decrypt them, you can add/amend your SAST tool to look for poor crypto implementations used to protect passwords and other sensitive information.

3. Threat Modeling also provides valuable inputs to Incident Response Training, Table-top exercises, understanding business impact of threat scenarios to management executives and so on.

However, there are right ways and wrong ways to do Threat Modeling. We use a combination of STRIDE, along with metrics like CVSSv3 to derive quantitative scores that can be used to prioritize. Our Methodology around Threat Modeling is predicated on the following practices:

Iterative Threat Models

The reason we use Agile and DevOps is due to Iterative Requirements. Applications undergo constant change and one of the principal complaints that Product teams have is that security is usually the laggard, which is unfortunately true. I believe that a lot of that security lag can be filled in with iterative threat modeling. For instance, Threat Modeling should begin at the Sprint Planning Meeting and diverge into security requirements, attack test cases that can be used across the sprint. Of course, to achieve this we need the threat modeling process to be simple, visual and easy-to-use. Lack of iterative threat models, results in lack of iterative security requirements, lack of updated attack test cases, which results in a huge amount of “catch-up” to be done by the security team, causing release delays, or worse, a vulnerable product being released to production.

Intuitive Threat Models vs Monolithic Documents

Many folks see Threat Modeling as an Enterprise Risk Assessment style document, akin to a procedure or a guideline. We don’t agree with this viewpoint. We feel that a Threat Model is a tactical “playbook” that needs to give actionable intelligence to the various groups consuming it. In my opinion, the ideal consumers of a Threat Model would be Devs, Architects, Ops, DevOps folks and in some cases, QA and Management folks. Large documents with complex diagrams is not the way to go, especially in a rapid, continuous delivery type of environment. We’d rather create a baseline Threat Modeling Document, that is simplified into a mindmap, which is customized for different groups of consumers. For instance, we would create an “Attack Model” for Security Testers, identifying the different kinds of threat scenarios drilling down into the types of attacks that could potentially lead to these risks being “brought to life”. For Devs, Architects and DevOps folks, we would look at “Mitigation Models”, mindmaps that would evolve from the “Attack Model” and provide specific security measures to resolve or mitigate these issues.

Application Process Flow vs Data Flow

While these terms are used interchangeably, there is a significant difference between Process Flow and Data Flow. In a Data Flow approach to Threat Modeling, one would look at Information Assets, Data Flow Paths and identify threats based on these inputs and parameters. However, this tends to not only be overwhelming, but also ineffective in drilling down to the critical threats and relevant scenarios. Approaching Threat Modeling from an Application Process perspective uses the Application's User process flow as a basis to perform the Threat Model. While the data points from both approaches may be relatively the same, the Application Process-driven Threat Models are far more effective and relevant than Data Flow Driven Threat Models. In addition to the effectiveness, Process Flow driven Threat Models are more efficient, (a key requirement for a fast-paced DevOps environment) making them more practical and attainable than Data Flow driven Threat Models.

The results of the Threat Modeling Exercise can serve as inputs to multiple activities, each of which is valuable in the Software Development Lifecycle:

1. Threat Models can be scored based on CVSSv3 or DREAD and security requirements and mitigation strategies can be defined specifically for these models, keeping in mind their severity and priority based on their scores. These serve as Security Stories or Security inputs for User Stories in the Product Backlog. This will ensure that security requirements are accounted for right in the beginning of the Lifecycle and the Sprint. We have seen organizations using apps like JIRA derive a great deal of benefit from this, because security implementation requirements are now visible to all stakeholders (previously relegated) and the fact that security is integrated to the story and the sprint in a very meaningful way, thereby ensuring that it can't be ignored.
2. Threat Models can be further chronicled as Attack Models and SAST Inputs. This ensures that security testers during and after the sprint know what to look for, target and attempt to compromise during a penetration test or vulnerability assessment. Again, this is very helpful in a SecDevOps environment, where there's limited time for security testing, especially when security testing needs to be executed iteratively.

Threat Modeling is a security playbook (rather than a static document) that needs to evolve with the application. It needs to be malleable and suitable to change to ensure that the organization can use it effectively throughout the lifecycle and beyond. Our methodology emphasizes the need for Threat Modeling as a cornerstone of a strong SecDevOps practice.