SAFECode

Threat Modeling at Scale

Authors: Ashwini Siddhi and Matthew Coles, Dell Technologies
Industry Contributor: Altaz Valani
Academic Contributor: Zhenpeng Shi, Boston University
Expert Reviewer: Steve Lipner, SAFECode

Table of Contents

Objective

The white paper is intended for developers, architects, security organizations and other personnel who have dealt with basic threat modeling and wish to upgrade existing threat modeling processes to address the "scale" factor.

This paper aims to provide the reader with:

- Requirements and challenges that arise due to the scale factor;
- A framework for evaluating different solutions (including a comparison of classes of tools) to meet these requirements;
- Processes for rolling out and adopting threat modeling at scale.

Important Definitions/Terms Used in This Paper

Term	Meaning
Model	An abstraction describing a system, system-of-systems, a component, a network, or a process
Threat	An action that could be taken against a system to cause a (usually) negative effect * NIST 800-160 uses this definition for threat: An event or condition that has the potential for causing asset loss and the undesirable consequences or impact from such loss.
Risk	A value expressing the probability of a threat being acted upon and the cost of impact from a successful threat action
Vulnerability	An exploitable weakness
Weakness	A fault, gap, exposure or defect
Exploit	A process, method, or function for using a vulnerability against a system; exploits may be manually performed or automated by tools
Threat Actor	An individual who would attempt to impact a system using a threat
Complexity	A measure of intricateness or lack of simplicity. Complexity is multi-dimensional, and can take many forms, such as: • Systems - number and structure of objects and their interactions

Term	Meaning
	• Organizational - number of people, organizational structure , and processes • Technological - disparate technological stacks. • Operational - value streams at an Org level
“At Scale”	Factors that increase complexity, vagueness, and turn-around times for threat modeling represent “At Scale” elements. Factors that contribute to “At Scale” include (but are not limited to): • A system with many moving parts, interfaces, functions. • A distributed group of stakeholders (e.g. different developers working on a system). • Frequent updates/releases for the system, requiring regular updates to the threat model. • An organization with many teams or individuals working on various system components or processes (like compliance, privacy, security etc.) • A rapidly evolving threat landscape. Based on these factors, “at scale” requires the ability to: • perform component level and as well as system level threat modeling; • update existing threat models faster, to keep pace with the evolution of the system; • achieve real-time collaboration with multiple stakeholders across geographical locations.
Threat Library	A guide used by practitioners during a threat modeling exercise to identify threats based on design patterns or other criteria. Threat libraries may be static or dynamic. Note: Not all threat modeling methodologies rely on threat libraries.
Severity & Priority	Severity is an objective parameter/metric that depicts the impact of a security bug on a given system. Priority is a subjective parameter/metric that indicates how soon or in what order a particular security defect should be fixed.

Threat Modeling Overview

According to the Threat Modeling Manifesto, Threat Modeling is an activity "for analyzing representations of a system to highlight concerns about security and privacy and if applicable, safety characteristics".

Threat modeling is a crucial activity of the secure development lifecycle (SDL) for identifying and mitigating weaknesses and potential security vulnerabilities. Threat modeling is most effective when performed as part of a Design Phase.

Threat Modeling in a nutshell is a means for stakeholders to collectively understand a system and its characteristics, including how those characteristics may contain faults or gaps in defenses which could be leveraged to negatively impact the system, and to shore up defenses in a way that improves the posture of the system.

Performing threat modeling can be an informal "thought exercise" or a more formal risk analysis and management effort; the approach, and the objectives, should be meaningful to the organization performing the activity.

Tools are often used to facilitate a threat modeling activity (and arguably are a requirement in order to achieve threat modeling "at scale"), but the use of tools should not detract from the conversation and collaboration which is the true value of threat modeling.

As systems evolve or become more complex, threat modeling becomes increasingly challenging using traditional approaches. In this whitepaper, we propose a holistic approach to threat modeling at scale, which combines traditional threat modeling with modern software engineering practices.

Traditionally, Threat modeling has had 4 core steps:

- Model a system;
- Analyze the model for threats;
- Identify mitigations;
- Validate mitigations are effective.

To address the criterion for Scale, this paper calls out steps to include pre- and post-activity functions as well.

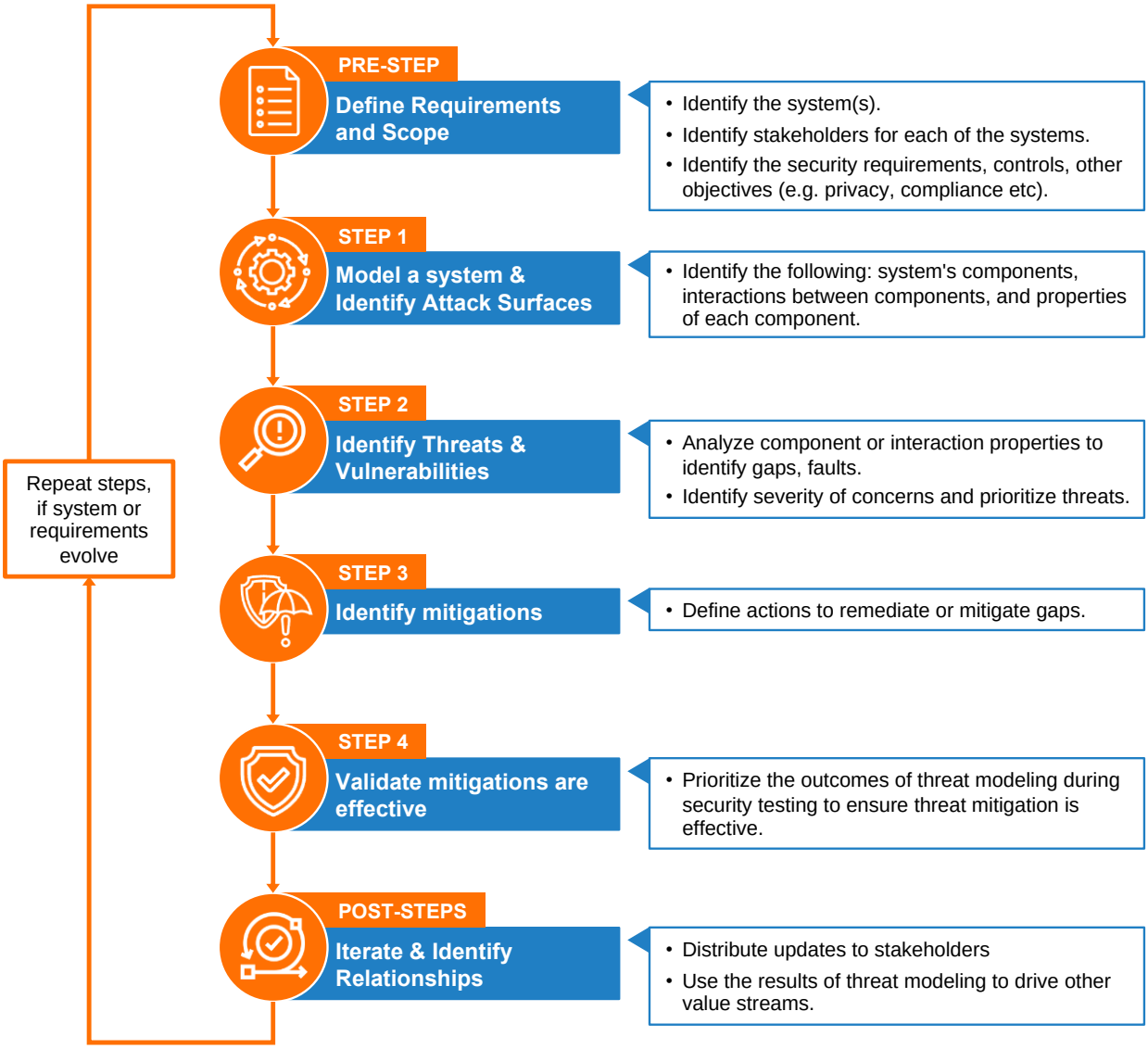


Figure 1: Steps for Threat Modeling
Refer to Appendix for more details

There are many recognized approaches to performing threat modeling; which approach is chosen for use by an organization is dependent on a number of factors, but the approach should support organization-specific business processes and needs, and result in outcomes that are meaningful for the organization and its stakeholders including customers.

Threat Modeling should eventually tie back to business objectives of the organization and the output of Threat Modeling should be used to drive other capabilities like secure coding, security testing, and security operations.

Defining Threat Modeling "At Scale"

Threat Modeling activity that encompasses multiple systems, across multiple releases and involves several stakeholders across an organization qualifies as *Threat Modeling at Scale*. This can also include other moving parts specific to an organization.

Typical example scenarios for requiring Threat Modeling at Scale include the following. Please note these are not mutually exclusive. For example, an enclave that addresses regulatory requirements can also impact architectural complexity.

- **Organizational Size** – the bigger the organization, the more stakeholders are to be involved in or kept informed on threat modeling.

  Example: Multiple divisions with varied processes, tools and deadlines giving rise to operational friction to threat modeling an integrated solution.

- **Architectural Complexity** – complex, hybrid architectures have a greater attack surface which takes longer to understand and to threat model.

  Example: An integrated application with components from multiple cloud providers, different technological stacks, and network segmentation spanning several geographical locations.

- **Regulatory Demand** – legal, security, privacy and other compliance requirements that are continually changing and may impact all or some parts of the system in scope.

  Example: Threat Modeling Cloud Technologies for compliance with Payment Card Industry (PCI), General Data Protection Regulation (GDPR), and International Organization for Standardization (ISO) for Application and Infrastructure Security standards.

- **Frequency of Release** – threat modeling has to keep up with the frequency of software development and configuration releases, avoid being outdated or delaying the time to release or time to market for a system that is being threat modeled.

  Example: High volume of design changes to a system in the Agile-DevOps environment with frequent, on-going releases (we recognize that some teams may release several times a day while other teams may have less frequent releases).

Requirements to Achieve Threat Modeling at Scale

Threat modeling impacts multiple roles at various levels of an organization. For the purposes of our discussion, we will limit it to project level activities (developers, threat modelers, product owners, etc) and stay away from executive stakeholders (CISOs, CTOs, CIOs, etc).

It is essential that all activities related to threat modeling are performed efficiently and the countermeasures recommended are translated into clear, atomic requirements that developers can understand and complete. For example, ambiguous requirements without clear exit criteria will slow down developers and lead to uncertainty about completion.

To be clear, we are not proposing a specific process methodology (waterfall, agile, etc). Rather, we are recommending that there be a mechanism through which threat modeling can aid in producing requirements and a feedback loop that allows the threat model to be updated. This keeps the threat model as a living artifact which can be queried and updated as new threats emerge and existing ones are mitigated.

Since threat modeling intersects with many other processes, it becomes necessary to consider how it should be done as efficiently as possible. That eventually touches on value streams and business value creation or benefits realization. However, our scope here is to limit the discussion to process efficiency at the project and program level (where many of the day to day development activities occur). For each step in the threat modeling process, teams need to consider whether there is waste while conducting the activity. Ultimately, we believe this involves business stakeholders who should participate in defining the scope of threat modeling.

Let us take a look at the different personas that deal with Threat Modeling on a day to day basis and understand their requirements or needs to address the "At-Scale" elements for Threat Modeling.

These requirements are prioritized as per the classification:

- **P0** - A core requirement to enable "at-scale"
- **P1** - A strong recommendation to improve "at-scale"
- **P2** - Good to have specialized capability for targeted users and generally useful in Threat Modeling.

Note: Please see Appendix B for a simpler mapping of requirements to priorities and personas.

Req 1	**Effective Threat Modeling at Speed**	**Priority: P0**
Requirement	Include Threat Modeling activity in the backlog/plan with appropriate story points (translate according to the Software Development Lifecycle (SDLC) followed). Security Acceptance Criteria set for every user story/feature going into the system for that release.	
Reasoning	Threat Modeling activity must avoid being time-intensive every release. The solution or methodology to threat model has to account for **agile methodologies** and other rapid development processes. One should be able to **replicate** existing Threat Models, apply **frequently changing requirements** and make necessary changes every release without compromising on the time-to-release or time-to-market of the products and applications. To account for frequent releases, one should not compromise on the effectiveness of the threat modeling activity either. The solution or methodology used to Threat Model should account for the security acceptance criteria for every user story going into the system for that release. Ideally, each security requirement should be captured as a user story.	
Applicable Personas	**Development and Engineering Teams**	

Req 2	Automated Diagramming	Priority: P2
Requirement	Generate Threat Models from Code or Workloads	
Reasoning	Most inexperienced threat modelers spend a considerable amount of time and effort in "beautifying" the threat model. This increases the overall time to complete a Threat Modeling activity. Solutions that eliminate this effort, like generating Threat Models from Code, JSON objects, or by connecting to running workloads should be considered. The principles of Infrastructure as Code can be used to solve this specific problem.	
Applicable Personas	**Development and Engineering Teams**	

Req 3	Application Lifecycle Management (ALM) Integration	Priority: P1
Requirement	Automated creation of backlog items for Dev teams	
Reasoning	The engineering team should be able to track the results of a Threat Modeling activity as a backlog item in an ALM system through *end to end automation*. For example, each threat and mitigation or requirement that has been generated during Threat Modeling should create a defect ticket or user story in the engineering backlog.	
Applicable Personas	**Development and Engineering Teams**	

Req 4	Refining the scope of SDL Activities	Priority: P1
Requirement	The results of Threat Modeling should drive static code analysis and/or other verification activities.	
Reasoning	To realize the full benefit of a Threat Modeling exercise and drive other SDL activities efficiently, the results of Threat Modeling should drive static code analysis and/or other verification activities through one of the following: • DevOps integration • Platform Engineering • Manual inputs. For example, if during Threat Modeling one realizes that a process is using an unmanaged programming language and could potentially be vulnerable to buffer	

Req 4	Refining the scope of SDL Activities	Priority: P1
	overflow attacks, then the automation scripts for testing should be able to specifically look for these issues in the system with specific secure development activities like fuzzing. (see SAFECode material in the footnotes on Secure Software Development practices).	
Applicable Personas	**Development and Engineering Teams**	

Req 5	DevSecOps Integration	Priority: P1
Requirement	Threat Modeling should be included in the CI/CD pipeline	
Reasoning	It is well known that CI/CD pipeline begins from the code and traditionally threat modeling has been out of scope for this activity. However, efforts should be made to integrate Threat Modeling into the DevOps pipeline for it to be a true DevSecOps operation. All of the above mentioned requirements work well with DevSecOps integration. Note that most diagramming tools require human involvement to generate the diagrams but the analysis, reporting and additional actions, if any, can be integrated with the CI/CD pipeline. There are also tools that are able to generate diagrams based on code and they can be used to completely integrate Threat Modeling into the CI/CD pipeline.	
Applicable Personas	**Development and Engineering Teams** **Security Management**	

Req 6	Standards for Security Architecture	Priority: P2
Requirement	Provide templates for classes of design.	
Reasoning	It is desirable for the solution or methodology to go a step further and support templates that provide a standardized architecture for each class of design objectives. Example: Payment gateways, E-commerce platforms, Single Sign-On (SSO) etc. With this add-on, Threat Modeling activity will be driving consistent security architecture as a standard across all products and applications in the organization/Business Unit and reduce the time and effort that would be required to create a new design for every solution. This will be especially useful for Cloud	

Req 6	Standards for Security Architecture	Priority: P2
	Technologies, Critical Infrastructure, IoT Systems, Embedded Systems or systems-of-systems.	
Applicable Personas	**Architects Modeling Complex Solutions**	

Req 7	Integration of Multiple Models	Priority: P0
Requirement	Capability to integrate several models and form a new model.	
Reasoning	Individual threat models of a complex solution may perform well at the system/component level but when they are integrated, they may introduce potentially new threats. Identification of these threats is at most times missed due to absence of DFDs of the bigger picture and solution objective. To counter this challenge in complex solutions, the threat modeling solution should provide capability to integrate several models and form a new model, highlighting the threats formed during the interconnection of these individual components. This can be applicable to IoT Systems, Embedded Systems or systems-of-systems. Note: Integration of models can mean systems to other systems or sub-components of a system. Microsoft Threat Modeling tool has the capability to have multiple tabs of diagrams within a given threat model but does not allow cross-correlation of diagrams and threats. As a work-around, the first tab can be used to represent the integration and the subsequent tabs can be used for individual systems/components. Commercial and other open source tools have the capability to integrate models of systems or subsystems within a single tab of a given threat model.	
Applicable Personas	**Architects Modeling Complex Solutions**	

Req 8	Chaining of Threats	Priority: P2
Requirement	Provide attack chain with associated severity	
Reasoning	Following the adoption of vulnerability chaining patterns by vulnerability researchers, the ability to link one threat with another weakness/threat in the same system or with another system in an integrated solution, provides chaining of threats. This attack chain/attack tree/kill chain along with its associated severity is a valuable input to developing mitigations to address the complete chain instead of building individual fixes with potential blind spots.	

Req 8	Chaining of Threats	Priority: P2
	For example, consider a Server Side Request Forgery (SSRF) in a web front end server, and a database access control issue in the DB layer. These threats might together form a chain that is valuable to be aware of in addition to the individual threats.	
Applicable Personas	**Architects Modeling Complex Solutions**	

Req 9	Threat Modeling Beyond Security	Priority: P1
Requirement	A holistic approach to threat modeling encompassing security, privacy and regulatory requirements.	
Reasoning	Currently, most organizations perform security, compliance and privacy activities separately and often they have overlapping requirements or controls. There is an opportunity to combine these activities together and identify and address issues at design time to enable ‘’start left’ instead of just ‘shift-security-left’. This will help ensure that architects and developers are not duplicating efforts and increase the value of the output of threat modeling.	
Applicable Personas	**Architects Modeling Complex Solutions** **Development and Engineering Teams**	

Req 10	Change Management/Change Records	Priority: P1
Requirement	An effective way to track the evolution of a model over multiple releases	
Reasoning	Having the ability to review the evolution of the model to see how it changes over time, and the threats that are identified and mitigated through this evolution, can help the security engineer (or other team members) understand capability and maturity of engineering teams and the threat modeling program, predict future concerns, and help manage risk and security posture effectively.	
Applicable Personas	**Security Engineers** **Development and Engineering Teams**	

Req 11	Real Time Collaboration	Priority: P0
Requirement	An effective way to manage the workflow for Threat Model Exercises	
Reasoning	Ability to create models and request reviews through automation, instead of attaching individual documents to emails/tickets etc is essential. Without this capability, things can quickly go out of control and it can become exceedingly cumbersome to track changes to the model and also to track who made what change.	
Applicable Personas	**Security Engineers** **Development and Engineering Teams** **Architects Modeling Complex Solutions**	

Req 12	Consistent or Complementary approach to Threat Modeling	Priority: P0
Requirement	Develop a common schema for Threat Modeling to achieve consistency	
Reasoning	Large organizations with engineering teams working on varied technology stacks use varied tools, processes and methodologies to build their threat models. This ad-hoc approach presents a challenge to the organization's security program in terms of defining what constitutes a good Threat Model and in ensuring the quality of Threat Modeling activities. To arrest this challenge, a consistent approach to Threat Modeling activity must be defined. However, a consistent approach cannot always be practical across a huge organization. Hence, complementary methodologies for Threat Modeling can be used, if other factors like common model definitions are supported. A common schema can help, even if standardization on a single solution is not available. Example: If a model is defined using an expressive language like Unified Modeling Language (UML) or the Open Threat Model schema, or another suitable system description language, then the source models can be consistently read and analyzed, potentially by different tools or methodologies. For more information, see references.	
Applicable Personas	**Security Management** **Security Engineers**	

Req 13	Other Valuable and Meaningful Outcomes	Priority: P2
Requirement	Drive towards policy goals that support business objectives	
Reasoning	Threat Modeling is a unique security verification activity to validate the resiliency of design and architecture of the system being modeled. The end objective of the activity is to identify security controls and build a security architecture to achieve resilience against attacks. It is important to keep this unique aspect in mind and not view threat modeling as a mere vulnerability assessment. For example: Instead of focussing on issues such as common web security concerns (e.g. HTTP Headers or CSP) when reviewing a model, a wide range of threats including threats to authentication/authorization, design flow and business objectives should be considered. The Threat Modeling activity may also be used to drive policy goals, or enforce/steer towards recognized or approved design patterns, or anything else that makes the activity "valuable" to an organization.	
Applicable Personas	**Security Management** **Development and Engineering Teams**	

Req 14	Cater to Varying Maturity or Understanding Levels	Priority: P0
Requirement	Enable Self Service when appropriate	
Reasoning	While it is important to define a consistent approach to Threat Modeling across an organization, one needs to be aware of realities. Not all engineering teams have the same skill-set and maturity levels. Threat Modeling tools and processes should cater to varying skill-sets and should also enable Self-Service when appropriate.	
Applicable Personas	**Security Management**	

Req 15	Centralized Solution	Priority: P0
Requirement	Provide a common platform for stakeholders to create and access latest artifacts (e.g. stencils, threat libraries etc.)	
Reasoning	In the era of microservices, a centralized solution might seem archaic but is an essential ingredient for the success of a Threat Modeling Service/Program across an organization.	

Req 15	Centralized Solution	Priority: P0
	It is desirable that the solution to Threat Modeling at Scale does not result in stand-alone artifacts which would require a medium like email or ticketing system to be shared across stakeholders for reviews or perusal. Instead, the solution should provide a platform for multiple stakeholders to access the latest artifacts for their perusal. This solution should automatically update threat libraries and stencils in real time and help in the adoption of the latest version without any waiting period. **Note**: Monolithic or centralized solutions are not always the answer to scale due to other factors like cost. There is also the potential for multiple solutions to co-exist, as long as the inputs and outputs can reasonably be interchanged or related. That way, perhaps, if one team is used to using an enterprise class tool for a large complex environment, but another sub-group is familiar with a basic CLI modeling tool, they could reasonably co-exist with a little overhead. Though not the ideal scenario, it can be a practical strategy during difficult economic situations.	
Applicable Personas	**Security Management**	

Req 16	Easy Adoption	Priority: P0
Requirement	An easy to use and understand solution that results in quick adoption	
Reasoning	A tool that is complex, and difficult to use and understand will result in delayed onboarding of Products and Applications and potential inconsistent or infrequent use. It is important that the Threat Modeling solution be **"sellable"** to an organization's internal customers. The features of the chosen Threat Modeling solution should be able to align and enable the business goals/strategy of the organization. For example: If business is migrating from on-prem to cloud, threat modeling should be able to adapt to these changes quickly and efficiently.	
Applicable Personas	**Security Management**	

Req 17	Actionable Results	Priority: P0
Requirement	Generate actionable list of results based on persona	
Reasoning	A Threat Modeling solution should generate an actionable list of results based on the persona viewing the results. What a developer wants to see is different from what a Security Program Owner would like to see in the results. The tool/solution should have the ability to integrate with reporting dashboards and defect tracking systems.	
Applicable Personas	**Security Management** **Dev and Engineering Teams** **Security Engineer**	

Req 18	Customizations	Priority: P0
Requirement	Encompass organization specific threats and their customization	
Reasoning	Each organization is different and may have specific threats to its domain that a generic tool would typically not encompass. A customizable solution to accommodate new changes in threat landscapes and account for new technologies specific to an organization is essential. This would also ensure that the solution is scalable and future-proof.	
Applicable Personas	**Security Management** **Dev and Engineering Teams**	

Req 19	Handle Conflicting Requirements	Priority: P2
Requirement	Encapsulate overlapping and confusing requirements from different pillars like Security, Privacy, and Compliance.	
Reasoning	Requirements when bridging different technologies and/or security/privacy/safety principles can be confusing as well as overwhelming. The chosen threat modeling solution should be able to internally encapsulate and objectively resolve this potential conflict. Typically, the end-users should not be left to resolve these challenges themselves as it could potentially decrease the quality of the threat modeling activity and increase turnaround times.	
Applicable Personas	**GRC and Organizational**	

Req 20	Cross Functional Threat Library	Priority: P0
Requirement	An extensive library of Threats with detailed mitigations and external industry references.	
Reasoning	A cross functional threat library that brings together data from different personas (security, legal, compliance, privacy, development, operations) is required. The data in this threat library is linked together to provide meaningful cross functional insights. The threat library is regularly updated and impact analysis can be performed when any change is made. This broadens threat modeling to include data flow analysis, regulatory changes, system changes, and business strategy changes.	
Applicable Personas	**GRC and Organizational**	

Req 21	Capability and Reasonable Costs	Priority: P0
Requirement	Capability to on-board multiple systems with varied technology stacks.	
Reasoning	The threat modeling solution should be capable of on-boarding new products and applications. If need be, it should also allow for existing threat models to be completely revamped to accommodate changes in the technology stack of the system being modeled and encompass the latest threats across these different technology stacks. Maintenance costs of the Threat Modeling solution itself must be reasonable including costs for 3rd party vendors (if any), resource costs, contractual costs, and infrastructure costs. Note that a “reasonable” cost is dependent on the organization implementing Threat Modeling	
Applicable Personas	**GRC and Organizational** **Security Management**	

Req 22	Value Stream Support	Priority: P2
Requirement	Drive aspirational goals of the organization	
Reasoning	An organization might like to drive other value streams like security testing, and hiring through the results of threat modeling. There can also be aspirational goals to translate specific findings from threat modeling to case studies to be published in security forums etc. Though not a priority item, it is certainly good to have the threat modeling solution enable driving of other value streams and achieving of aspirational goals. This could be an attribute of a cutting-edge, differentiator solution	
Applicable Personas	**GRC and Organizational** **Security Management**	

Req 23	Training	Priority: P0
Requirement	Solution has associated training materials or sufficient collateral to build training material.	
Reasoning	Learning how to Threat Model effectively takes time and requires practice. It is important to enable development teams with appropriate training and hands-on activities to learn the process. Development teams must enable developers and invest in sufficient cycles/sprints for training. The solution or methodology used for threat modeling must have appropriate training content that is accessible to all stakeholders. A solution should be easy to learn and not require specialized skills to use. A methodology should be accessible to most stakeholders, some of whom will not have a strong understanding of security or privacy. Training may increase the cost of the Threat Modeling Solution, especially in the case of a commercial product.	
Applicable Personas	**GRC and Organizational** **Security Management** **Security Engineer** **Architects Modeling Complex Systems** **Development and Engineering Teams**	

Framework for Evaluation of Solutions

Over the years, many solutions for threat modeling have been proposed and developed. The solutions vary in terms of capabilities and workflow, as they often focus on different aspects of threat modeling. As a result, it is often difficult to compare threat modeling solutions directly, which makes the selection of threat modeling solutions a non-trivial problem. A standardized framework for evaluating various threat modeling solutions would be highly valuable.

Threat modeling solutions include both methodologies and tools. Methodologies are typically abstractions, such as STRIDE [1], PASTA [2], and LINDDUN [3], of how threat modeling should and be performed. On the other hand, threat modeling tools provide assistance in the actual implementation of threat modeling in the SDL, often in the form of software, for example, the Microsoft Threat Modeling Tool [4] and OWASP pytm [5]. The design of such tools involves the selection of one or multiple methodologies for threat modeling, as well as other practical aspects, such as integration with other tools in the SDL, and collaboration between different teams in a large organization.

There have been a few attempts to build frameworks for evaluating threat modeling methodologies or tools, such as the work in [6] and [7].

Nonetheless, it is debatable whether the criteria proposed in previous work are the most appropriate ones, and there is still a lack of consensus on a standardized evaluation framework. General criteria for characterizing solutions provide valuable intuition, but sometimes can be subjective. For example, some users may consider a solution to be easy to learn and use and, as such, to be highly adoptable. However, other users may not feel the same way. One option is to use more concrete and objective criteria, such as the documentation of the solution, and let users decide for themselves based on these objective criteria.

Next, we propose our framework for evaluation of threat modeling solutions. Here, the solutions include both methodologies and tools. Moreover, we focus on solutions for threat modeling at scale, meaning that the solutions are suited for large-scale use in complex systems. As a result, the focus of our criteria leans towards suiting the needs of large organizations.

- **Scalability.** Products and Applications of large organizations run into more than thousands. The solution should have the capability to onboard these multiple complex systems and the systems' stakeholders as users as and when needed into the tool.

  Also, as the number of components in a system grows, the complexity of the threat modeling process increases exponentially, making it difficult to keep up with the pace of development. The tool should account for the need for multiple models for a single system based on different releases to address this.

- **Accessibility (for Collaboration).** Threat modeling at scale typically requires collaboration between multiple teams, and involves multiple stakeholders. The tool should support coordination between teams and help them reach consensus on threats without having to share multiple artifacts across multiple media/channels.

- **Adoptability.** The solution needs to be easy to learn and use in order to be adopted in large organizations.
  - Is the solution mature and ready to use in practice?
  - Has it been implemented in an organization successfully?
  - Does the solution provide adequate training and documentation?

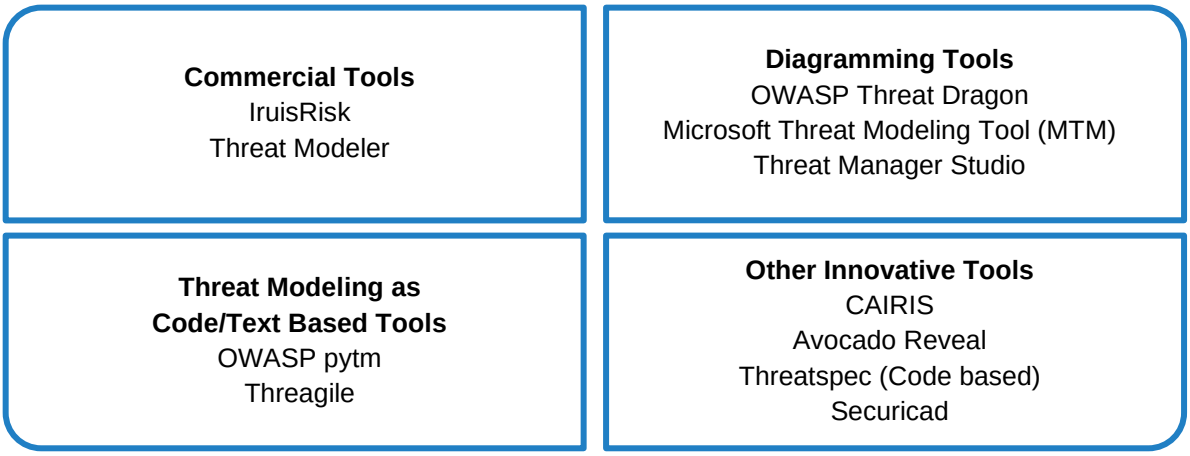
- **Viability (Cost).** The cost of a solution may increase significantly when used on a large scale.

  - Is the solution free to use (e.g., open source tools)?
  - If not, what is the pricing model of the solution?
  - How does the price change with the scale of the system?
  - What are the additional costs of the solution (e.g., costs of human resource and maintenance)?

- **Automation.** It is not practical to perform threat modeling at scale entirely manually. A viable solution needs to support at least some level of automation for efficient threat modeling. Having automated solutions is a key in the transition from DevOps to DevSecOps.
  - To what extent can threat modeling be automated - fully or partially?
  - How does the tool fit into the DevOps / DevSecOps toolchain?
  - Can the threat modeling solution/methodology be integrated with multiple tools across the software development lifecycle for agile methodologies?
- **Applicability.** Threat modeling includes multiple steps.
  - Different systems often have different perspectives of threats, which results in varying requirements. Does a solution work for a specific kind of systems (e.g., cyber-physical systems), or for general systems? Does the solution work for hardware, software and firmware systems?
  - Can the solution be customized to suit the needs of a specific system?
- **Maturity (of Threat Library).**
  - Which kind of threats (e.g., security, privacy, or safety) does the solution aim to identify and mitigate?
  - Can additional threats be added to the threat modeling solution?
  - Are detailed, low level threats reported?
- **Reusability.** Ability to create templates to
  - Drive architectural standards across the organization.
  - Create a baseline system design for conceptualization and use it across multiple releases as it evolves.
- **Extensibility.** The solution should be Adaptive and Current i.e. adapt to new technology stacks and the latest threats in an automated fashion, without requiring manual effort.
- **Methodology.** Though the internals of a threat modeling solution cannot be determined, it is good to understand what methodology the solution uses - is it attacker based, defender based or a combination of both? For holistic results it is always recommended to go with a model that combines both perspectives.
- **Actionability of Results.** The results of threat modeling should be consistent across multiple runs and provide appropriate mitigations for all of the issues reported. Rather than presenting lengthy reports, results should be concise and actionable for developers and architects to implement as part of their backlogs. The number of false positives reported should be negligible.

Evaluation of Different Solutions

Classes of Tools



For the purpose of this paper, we will only look at the first three classes. The other innovative tools have been called out for informational purposes only. Though under the same classification, each of these tools in this category can be different and they cannot be compared through the same lens.

Commercial Tools	Diagramming Tools	Text Based Tools
Maturity of Library		
Threat Libraries for Software and Cloud that are kept current, with a well defined update cadence. These tools tend to support certain technologies better than others (e.g. web or cloud systems over hardware and firmware). Customization of libraries is also an option.	Threat Libraries are usually generic but often extensible. The updates to the threat library have no regular cadence and can be ad-hoc. Most of these tools are open source; updates are irregular and based on the community for maintenance. As open source tools, users can make modifications to add their own threats as they require them.	Threat Libraries are usually generic in nature, but often extensible. The updates to the threat library have no regular cadence and can be ad-hoc. Most of these tools are open source; updates are irregular and based on the community for maintenance. As open source tools, users can make modifications to add their own threats as they require them.

Commercial Tools	Diagramming Tools	Text Based Tools
Scalability		
Supports multiple systems and users	These tools generally support the development of a single threat model (which may contain one or more diagrams) but do not provide the required aspects of scalability to manage models and diagrams across projects without extra effort.	Tools in this category are implemented through build processes (e.g. as scripts that are executed, or code annotations that are processed). Scalability of model changes or threat information updates comes from those processes e.g. ability to manage multiple programs, models, and/or user accesses, not the modeling tool explicitly.
Accessibility for Collaboration		
A centralized tool can be used for collaboration and there is no need to share artifacts over different mediums.	The Microsoft Threat Modeling tool creates stand-alone artifacts (individual files containing one model/one or more diagrams per file). Sharing of these files usually requires them being emailed to stakeholders or held in a file repository or share; modification of these files tend to be single-user edit. OWASP Threat Dragon provides the ability to tag Reviewers and Contributors, and supports a web-based version tied to github for sharing	Threat models are created in/as source code that can be managed with collaboration tools such as git, gitlab, etc.

Commercial Tools	Diagramming Tools	Text Based Tools
Adoptability		
These solutions come with in-built training and professional services at an additional cost.	Though there are no professional services, these tools are intuitive and easy to use. They also have brief documentation on their websites (MTM and Threat Dragon) that demonstrates how to get started with these tools.	Though there are no professional services, these tools are intuitive and easy to use. Some of these tools have decent documentation available alongside the tool or source code. However, the update cadence of these documents and support is not yet very well defined.
Viability (Cost)		
These tools can be cost-prohibitive in large organizations due to their licensing models/subscription fees. An unlimited licensing model/subscription with a price cap can be a good work around in such scenarios.	These are mostly open source and require no cost to purchase or maintain except perhaps time and effort to add threats and make the tool available to all stakeholders.	These are mostly open source and require no cost to purchase or maintain except perhaps time and effort to add threats and make the tool available to all stakeholders.
Automation		
These tools are built to support complete integration with the development lifecycle. Some of these tools also provide the option to Threat Model as Code.	The focus of these tools are purely threat modeling and they do not support integration into development pipelines, but automate the threat generation/identification process.	These tools are automated, and represent the activity as “Threat Modeling as Code”; they automate the process of threat identification and model rendering. Integration with the development lifecycle for reporting, analysis etc. is yet to be developed or in nascent stages.

Commercial Tools	Diagramming Tools	Text Based Tools
Applicability		
Tool/Solution can be customized for applicability. In some areas, they also provide clear segregation between categories of applicability.	Tools offer generic threats that can be used cross-domain. The tools can be customized, if required.	Tools offer generic threats that can be used cross-domain. The tools can be customized, if required.
Reusability		
Ability to create templates for further re-use is supported	These tools do not support creation of model templates, although they do contain templates (“stencils”) for objects used in the diagrams.	These tools are written as source code and can support creation of model templates.
Extensibility		
These tools are updated automatically by the vendor and the threats are in alignment with the industry best practices and understanding at that point in time.	Updates need to be manually installed, and will depend on deployment options (local installation or part of a shared environment).	Updates need to be manually installed, and will depend on deployment options (local installation or part of a build environment).
Methodology		
Not disclosed but based on demos the Threat Library is based on both attacker and defender mechanisms.	Microsoft Threat Modeling Tool implements STRIDE OWASP Threat Dragon implements STRIDE, LINDDUN, and CIA methodologies	pytm implements a Threat Library based on STRIDE categories and CAPEC/CWE, and which is extensible. Threagile implements a Threat Library which is defined by the developer and partly based on CWEs. Each threat has a corresponding CWE.

Commercial Tools	Diagramming Tools	Text Based Tools
Actionability of Results		
Results are provided with recommended mitigations in an approachable manner for developers and architects to take action. False positives are kept to a minimum.	Threat descriptions can be hard to understand for non-security users, and may only provide general mitigation information as these tools usually lack specific context for the system being modeled. These tools can generate higher false positive rates due to lack or incorrectness of provided information. Some tools, like the Microsoft Threat Modeling tool, offer the ability to customize the threat descriptions to provide better guidance and easier to understand information.	Threat descriptions can be hard to understand for non-security users, and may only provide general mitigation information as these tools usually lack specific context for the system being modeled. These tools can generate higher false positive rates due to lack or incorrectness of provided information. These tools offer the ability to customize the threat descriptions to provide better guidance and easier to understand information.

See Appendix C for more detailed tool-specific information.

Rolling Out Threat Modeling at Scale

Organizations should establish a standardized threat modeling process to ensure consistency and scalability. The process should align to the steps called out in Figure 1. Ensure that clear roles and responsibilities are assigned to individual team members to ensure that the process is effective, followed and tracked.

It is recommended to implement this process through a specific threat modeling program for the organization.

The Action Checklist for managing a roll out of a Threat Modeling program at Scale:

- Form a “Core Team” of stakeholders approved by the leadership
- Understand business stakeholder history with threat modeling, specific needs, and expectations
- Identify suitable methodologies for creating models and performing analysis
- Identify opportunities for automation
- Define a Reference Design for Threat Modeling, including automation capabilities
- Create and document threat modeling processes and supporting collateral.
- Integrate existing / deploy new capabilities (Day 1)
- Onboard initial stakeholder teams

- Collect telemetry, metrics, and feedback
- Identify and plan opportunities for improvement based on data (Day 2..n)
- Identify additional teams for onboarding
- Manage, Monitor, Report
- Continuous Training & Communications

Creating a threat modeling program and applying it “at scale” needs to be planned and carefully managed to support the needs of the organization and the systems it creates. While the actions and specific details necessary to create and roll out any program within an organization are going to be by necessity organization-specific, there are common themes [10] that can be leveraged. In addition to the common approaches, here are some considerations for delivering threat modeling to the business that support performing this activity “at scale”.

Form a Core Team of Stakeholders

Core team membership should include representation from security, privacy, and operations teams, engineering and development community members, and select members of business leadership. Good representation, with the visibility and sense of ownership and involvement that comes with representation, can make or break program success.

What is a good size for the Core Team?

Core teams can range in size from a handful of individuals to dozens of people; the exact size will depend on your organizational structure and needs. A core team larger than 20 people may be too cumbersome to manage and meet all the varied requirements and expectations that individual stakeholders may bring to the table. Consider limiting membership to key personnel who can act as delegates for a broader set of constituents. A general rule of thumb 7+/-2 is considered an “ideal” team size [11].

Is the Core Team a “democratic” body?

It need not be democratic (e.g. members with voting rights, majority votes to pass), but members should feel that they have a stake in the decision making process and an opportunity to elevate concerns so they can be addressed appropriately. Having a couple of members who lead the team and who moderate discussions, elicit feedback, draft proposals, and gain consensus (which may not be unanimous) among members, can be beneficial.

Understand Business Stakeholder Needs and Expectations and History With Threat Modeling

Knowing what the business users expect from threat modeling (implicitly or explicitly) will help create a program that can scale. If an existing non-scalable program or solution (or frequently very ad-hoc approaches) is already in place, understanding the delta between those solutions and what might be needed to support something “at scale” will help enable success.

See the Requirements for Threat Modeling at Scale section above for things to consider in this step.

Create a “Shortlist” of Methodologies, Refine the List to Suit Needs

Use the Framework for Evaluation of Solutions section above to identify the methodologies that might support the at-scale needs of the organization. Refine the list based on the selection factors and stakeholder needs and expectations to achieve a targeted list of one or two approaches to launch Program Day 1.

Identify a Path to Automation

Threat Modeling at Scale works when some automation exists to facilitate or perform “non-value-add” type of work, such as creating models from descriptions and generating documentation.

Create Supporting Collateral

Once a short list of methodologies has been identified, consider what it will take to implement and deploy a program based on it. Depending on the methodology(ies), and eventually any tooling that may support the approach, you will need to create guidance and supporting processes to facilitate adoption, integration, and use of the mechanism.

- Create a high level process document, highlight roles and responsibilities for individual team members. Define time cost in hours required for these roles and responsibilities to plan for capacity accordingly.
- Define a standard to ensure the threat modeling activity is effective. Call out what constitutes a “good threat model” and set the completion criteria for a threat modeling exercise.
- Define inputs, document guidance to prepare for and use, model types, and workflow/processes.
- Define threat sources and threat rules (if applicable).
- Define outputs/outcomes, document guides to facilitate triage, reporting, and mitigation/remediation.
- Define operationalization, how to integrate with DevSecOps (if applicable), and release code snippets and libraries/tools.

Deployment “Day 0” to “Day 1”

Release the solution in the BETA stage after fine-tuning the identified methodology/solution to meet organizational needs and gathering early stakeholder approval.

In this BETA stage, engage with diverse teams across the organization to gather feedback and opportunities for improvement.

Make sure to address the critical feedback comments first, then add the rest to your backlog or planning schedule.

After addressing critical feedback and other backlog items, the solution should be ready for 'Go Live' or to be 'Generally Available'. Communicate the Generally Available (GA) release well in advance using various platforms and channels to ensure adoption and engagement.

Deployment “Day 2”

Continue to roll out the solution to different engineering teams either in a phased manner or in a prioritized order.

After the solution is GA, there may be a lot of curiosity and interest by the targeted end-users. To address constant questions and queries by the end users and to keep the momentum going, keep the communication channel open with end users by doing the following:

- Provide email or other direct contact methods for users to reach to submit questions or concerns
- Define an automated channel for receiving feedback
- Engage with end-users in terms of refresher sessions, quizzes etc.
- Provide formal training, helpful guides, and FAQs.

Start working on Continuous Improvements for the solution in parallel during its adoption. Engage with core team members on a regular cadence (such as weekly) to review the open items on the feedback list, to plan how to address feedback items, and to take action. Apply and publish these feedback "fixes" in smaller increments as part of frequent releases.

If a centralized tool/platform is being used to Threat Model at Scale, ensure that there are different instances of it. The production instance is used for actual engagement and adoption. The staging/dev instance is used to try different configurations, apply new ideas, and as a “playground” to innovate.

Manage, Monitor, Report

Define appropriate KPIs and Metrics and measure to determine if success criteria have been met.

Put processes and tools in place to support these measurements and reporting outcomes.

Monitor progress of these metrics over time and adjust as needed.

These numbers can be reported to leadership to show-case the efficiency and the usefulness of the program.

Threats evolve, and the at-scale program needs to as well. Tweak rules, expand scope etc. to include evolving technologies etc. (like privacy) and suit organizational strategy.

Training & Communications

Threat modeling is a niche skill and requires specialized expertise. In large organizations, there may be a limited number of experts with the required skill set, making it difficult to scale up the threat modeling process.

Set up a training program and use different channels to develop a “threat modeler” community in the organization.

In large organizations, different teams work on different parts of the system. Coordinating between these teams can be challenging, and communication gaps can lead to incomplete threat modeling despite having a well-defined Threat Modeling process. Define and set up a team for communications and coordination across the organization.

Summary

Business leadership should be on the lookout for ways to improve productivity and effectiveness (and ROI) of security programs. Implementing a threat modeling program “at scale” can be an effective way to improve productivity and the security of systems being built or deployed by your organization. Creating such a program will support collaboration among engineering teams and improve the quality of findings.

References

[1] https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats

[2] https://versprite.com/blog/what-is-pasta-threat-modeling/

[3] https://www.linddun.org/

[4] https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[5] https://owasp.org/www-project-pytm/

[6] N. Shevchenko, B. R. Frye, and C. Woody, “Threat modeling for cyber-physical system-of-systems: Methods evaluation,” Carnegie Mellon University Software Engineering Institute Pittsburgh United States, Tech. Rep., 2018.

[7] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, “Threat modeling tools: A taxonomy,” IEEE Security & Privacy, vol. 20, no. 4, pp. 29-39, July-Aug. 2022.

[8] https://safecode.org/uncategorized/fundamental-practices-secure-software-development/

[9] https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf

[10] https://www.pmi.org/pmbok-guide-standards/foundational/pmbok

[11] https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two

In Closing

The main purpose of this paper is to demystify the practice of threat modeling at scale. The collective feeling among the authors is that the return on investment from threat modeling justifies the effort necessary to get it started; from there on, it is just about practice.

We encourage you to start now. We would like to receive your feedback on this paper. We are very interested in hearing about your experiences doing threat modeling in your industry and environment. Please send your comments to feedback@safecode.org.

About SAFECode

SAFECode is a global industry forum where business leaders and technical experts come together to exchange insights and ideas on creating, improving, and promoting scalable and effective software security programs. We believe that secure software development can only be achieved with an organizational commitment to the execution of a holistic assurance process, and that sharing information on that process and the practices it encompasses is the most effective way for software providers to help customers and other stakeholders manage software security risk.

Appendix A

Detailed Steps for Threat Modeling

Step	Actions
The Pre-Step should be done prior to beginning a threat modeling initiative, and for each system being evaluated as it enters the program. *Also, at a program level the below should have already been identified:* ● Identify a methodology to follow for the activity ● Identify metrics and/or KPIs to help to determine what success looks like	
Pre-Step. Define Requirements and Scope	● Identify the system(s) for evaluation ● Identify stakeholders for each system ● Identify the security requirements, controls, other objectives (e.g. privacy, compliance etc).
Regardless of the specific threat modeling methodology or approach chosen in the Pre-Step, the general flow follows in steps 1 through 4	
Step 1: Model a system & Identify Attack Surfaces	● Identify the system's components, interactions between components, trust boundaries, and properties of each component.
Step 2: Identify Threats & Vulnerabilities	● Analyze component or interaction properties to identify gaps, faults, or other exposures, using available sources of threat or vulnerability patterns. ● Identify severity of concerns and priority of threats
Step 3. Identify mitigations	● Optionally, consult with risk management to determine if effort to address should be taken (e.g. to fix, mitigate, or accept the risk) ● Define actions to remediate or mitigate exposures, based on system security, privacy, or safety engineering principles, and common architectural patterns.
Step 4: Validate mitigations are effective	● Prioritize the outcomes of threat modeling and inform lifecycle activities such as security requirements, functional and security testing, to ensure threat mitigation is effective
Following each threat modeling activity (new or an iteration), consider performing the Post-Steps below.	

Step	Actions
Post-Steps: Iterate & Identify Relationships	• Monitor for new threats and evaluate abuse cases for systems, accordingly. • Distribute updates to objectives and stakeholder needs (as needed) • Distribute updates to threats (if using a threat library) • Repeat steps 1 through 4 as and when the system (model) or requirements (threats) evolve • Based on organizational goals and aspirational values, use the results of threat modeling to drive other value streams.

Appendix B

Requirements to Priority Mapping

Requirements Number	Requirements	Priority
R1	Effective Threat Modeling at Speed	P0
R2	Automated Diagramming	P2
R3	ALM Integrations	P1
R4	Refining the Scope of SDL Activities	P1
R5	DevSecOps Integrations	P1
R6	Standards for Security Architecture	P2
R7	Integrations of Multiple Models	P0
R8	Chaining of Threats	P2
R9	Threat Modeling Beyond Security	P1
R10	Change Management/Change Records	P1
R11	Real Time Collaboration	P0
R12	Consistent or Complimentary Approach to Threat Modeling	P0
R13	Other Valuable and Meaningful Outcomes	P2
R14	Cater to Varying Maturity Levels	P0
R15	Centralized Solution	P0
R16	Easy Adoption	P0
R17	Actionable Results	P0
R18	Customizations	P0
R19	Handle Conflicting Requirements	P2
R20	Cross Functional Threat Library	P0
R21	Capability and Reasonable Costs	P0
R22	Value Stream Support	P2
R23	Training	P0

Requirements to Persona Mapping

Requirements - Priority	Dev & Engineering	Architects	Security Engineers	Security Management	GRC & Org
R1 – P0	Y				
R2 – P2	Y				
R3 – P1	Y				
R4 – P1	Y				
R5 – P1	Y			Y	
R6 – P2		Y			
R7 – P0		Y			
R8 – P2		Y			
R9 – P1	Y	Y			
R10 – P1	Y		Y		
R11 – P0	Y	Y	Y		
R12 – P0			Y	Y	
R13 – P2	Y			Y	
R14 – P0				Y	
R15 – P0				Y	
R16 – P0				Y	
R17 – P0	Y		Y	Y	
R18 – P0	Y			Y	
R19 – P2					Y
R20 – P0					Y
R21 – P0				Y	Y
R22 – P2				Y	Y
R23 – P0	Y	Y	Y	Y	Y

Appendix C

Threat Modeling Tool Assessments

This section includes information on a selection of tools that are frequently used for facilitating or performing threat modeling as an activity during a system's life cycle. Each tool is measured against the factors highlighted in the Framework for Evaluation of Solutions section and presented here as a reference guide for readers. The tools selected for this appendix are based on member usage experience and do not represent a complete list of possible tools or any specific endorsement of them. A thorough curated (though not complete) list of threat modeling resources can be found here.

1. Microsoft Threat Modeling Tool (free)

The Microsoft Threat Modeling Tool (MTM) has evolved over several years. For starters, it provides a default library (here called template). Once the model is built based on the template, the tool identifies threats by checking threat conditions in the template. Threats in the default template are categorized by STRIDE. The template can be customized to suit specific use cases. Elements in the diagram, threat types, and threat properties are all customizable. Threat properties include description, severity, and countermeasures. Users can also add community-contributed templates or build their own ones from scratch.

- **Scalability.** Templates can be created for modeling specific systems (e.g., template for development with Azure cloud service). However, system modules cannot be reused or imported, and new models need to be built from scratch. When the system scale is large, it can be difficult to build the entire system model. No integration with other tools in SDL.
- **Accessibility (for Collaboration).** No support for collaboration between multiple teams. All files are saved locally. Only works on Windows.
- **Adoptability.** The tool was first released in 2014 and regularly maintained since then. The latest version 7.3.21108.2 was released in November 2022. It is relatively mature and can be downloaded from the Microsoft website. Documentations on how to use the tool are also provided.
- **Viability (Cost).** Free to use. Maintained by Microsoft. However, it might result in additional costs of human resources to tailor the template for specific use cases.
- **Automation.** In the templates, conditions under which a threat might exist can be defined for each system component. The tool can then automatically identify potential threats based on the conditions. The basic severity (high/medium/low) and mitigations of each threat can also be pre-defined, regardless of the actual system model. A threat report of the system can be automatically generated. No other automation is available.
- **Applicability.** Focus mainly on modeling a system and identifying security threats. A default template is provided for the general purpose of threat modeling. Customizing the templates can make the tool work for specific kinds of systems.
- **Reusability.** Templates can be saved and reused. Models cannot be reused in new models.
- **Extensibility.** Templates can be customized to adapt to different maturities of the system model. A template can define general system components (e.g., Generic data store), or more specific ones (Azure SQL database).

2. OWASP Threat Dragon (open source)

OWASP Threat Dragon is designed as a relatively light-weight tool. Its threat library (here called threat rule engine) is not as customizable as the other tools in this section. Like the MTM tool, it provides support for users to draw DFDs and associate threats to the components in the diagram. Note that it suggests generic potential threats, while details of the threats are left for users to fill in. Its advantages include workflow integration with GitHub, and support for more threat categories (LINDDUN and CIA) besides STRIDE.

- **Scalability.** System modules cannot be reused or imported, and new models need to be built from scratch. When the system scale is large, it can be difficult to build the entire system model. Scripts are provided for integration with Jira.
- **Accessibility (for Collaboration).** Works as a desktop or web application. A docker image is also provided.
- **Adoptability.** Developed since 2015-2016. Version 2.0 was released in Feb 2023, and maintained since then. Installation and demonstration related artifacts are provided.
- **Viability (Cost).** Free to use. Maintained by OWASP team.
- **Automation.** A threat rule engine (similar to templates in Microsoft MTM) can automatically suggest possible threats, but only generic ones (e.g., generic tampering threat). More information of the threats needs to be filled in by users. A threat report of the system can be automatically generated. No other automation is provided.
- **Applicability.** Designed for general purpose threat modeling on a relatively abstract level. By default, it focuses on security threats. It can also be used for privacy threat modeling by applying the LINDDUN threat rule engine.
- **Reusability.** The threat rule engine provided by the tool can be reused. Models cannot be reused in new models.
- **Extensibility.** No customizable part. Nonetheless, the tool can be extended as it is open-sourced.

3. OWASP pytm (open source)

OWASP pytm describes the system model in Python language. It requires users to create a Python object for each of the system components based on its pre-defined Python classes. The tool then generates a system diagram and identifies threats from the text-based model. Its threat library contains comprehensive information about each potential threat. To mention a few, the “condition” attribute for threat identification, the “likelihood” and “severity” attributes for simple evaluation, and the “mitigation” attribute suggesting countermeasures.

- **Scalability.** System models are saved as Python files, as such, new models can be built on top of existing ones. Models of large-scale systems can be saved in multiple files for better modularity. No official integration with other tools in SDL, but could be easily used with other tools (e.g., Git) since all models are saved as code.
- **Accessibility (for Collaboration).** Works as a command-line interface (CLI) application. No official support for collaboration between multiple teams.

- **Adoptability.** Version 1.2.1 was released in March 2022. Actively maintained by a team of community members; support is provided via issues in GitHub or a Slack group monitored by developers. Documentation is available as part of the code base.
- **Viability (Cost).** Free to use. Maintained by OWASP team.
- **Automation.** Pre-defined threat rules can automatically suggest possible threats. System and sequence diagrams can be automatically generated from the code. A threat report of the system can be automatically generated from the same model (code).
- **Applicability.** Focus mainly on modeling a system and identifying security threats. Works for abstract and high-level system models in general. Nonetheless, customized threats can be added to the rule library such that they can be identified.
- **Reusability.** Pre-defined threat rules can be reused for different models. Code for system models can be reused and imported.
- **Extensibility.** Customized threats can be added to the library. Moreover, the tool can be extended at more levels as it is open-sourced.

4. ThreatModeler (commercial)

ThreatModeler's SaaS platform offers an API-first design, enabling automation and extensibility. Behind the interface which offers the ability to drag & drop numerous out-of-the-box components representing technical processes or architectural objects is a customizable framework (Threat Framework) that defines the behavior of threat modeling exercises. Although agnostic of any specific threat modeling process, users may highlight threats from any number of sources such as OWASP, MITRE ATT&CK, CAPEC, CIS, or STRIDE.

ThreatModeler can also be deployed as an On-Prem Solution.

- **Scalability.** Provides good support for scalability by defining baseline system models. Integrations include 2-way ticketing (eg Jira, ServiceNow), DevOps enforcement solutions, and the ability to dynamically create models from numerous sources (AWS, Azure, CloudFormation Templates, Terraform, or even arbitrary sources) via advanced product capabilities or APIs.
- **Accessibility (for Collaboration).** Works as a web application for users to collaborate within or take advantage of ticketing integration support to help with tasks identified during exercises without having to log into ThreatModeler. Role-based access control model controls visibility and permissions across various users and lines of business.
- **Adoptability.** Version 6.0 was released in September, 2022 and is actively maintained. Documentation inside of the product covers subject matters from beginner to advanced. Additionally, ThreatModeler offers a Community whereby users of other customers may be engaged for greater awareness of usage and processes. Hands-on, assisted onboarding is provided with the enterprise version
- **Viability (Cost).** Licensing structure is based on the number of threat models an organization is looking to build on a yearly basis. Annual subscriptions include unlimited user access, virtual hands-on training, support, and access to customer success teams.
- **Automation.** Creation of diagrams can be automated , either via CloudModeler (AWS, Azure), IAC Assist (CloudFormation), soon Terraform and ARM or even arbitrary sources using an open API format with a simplified JSON structure. End-users may also utilize APIs for tasks such as user management, reporting, or integration to/from other sources. DevOps plug-ins may be used

to enforce threat modeling via pass/fail automation tasks and 2-way ticketing is available for Jira, ServiceNow, or Azure Boards.

- **Applicability.** ThreatModeler offers dozens of templates out-of-the-box along with a Model Marketplace featuring several prepared threat model diagrams. The Threat Framework contains thousands of objects including architectural objects, components related to process flow, threats, requirements, and other such intelligence aggregated from numerous sources or produced by ThreatModelers research team.

  ThreatModeler can be used for generic threat modeling and as well as in depth threat modeling for various technology stacks like Cloud, Web Apps etc.

- **Reusability.** Templates, diagram objects, and threat model data may all be readily stored, configured, and designated for reuse. These templates support any number of repeatable patterns to minimize manual diagram efforts.

  Additionally, importing capabilities may be extended to diagrams created with other tools (Visio, MTM, Draw.IO).

- **Extensibility.** ThreatModeler's API-first design enables organizations to be creative with threat modeling processes and even beyond the numerous native integrations. Templates can be created to drive architectural patterns and baseline secure designs across multiple systems.

  The Threat Framework may be customized to support in-house controls/requirements, and compliance standards as well as contextualized prompting to understand the usage or implementation of various architectural components or processes.

5. IriusRisk (commercial)

IriusRisk embeds the Draw.io diagram editor in its web application to enhance user experience when drawing the DFDs of a system. When initializing the diagram and adding new elements, questionnaires are provided to help users with configuration. Its threat library covers not only common knowledge bases like CVE and CWE, but also self-defined threats from typical enterprise use cases, such as AWS deployments. IriusRisk provides rather comprehensive threat evaluation, and offers priority and cost estimation of the suggested countermeasures.

- **Scalability.** Provides good support for scalability, such as defining nested system components for reuse, and importing existing system modules. Also provides integration with other tools in SDL, including issue trackers (e.g., Jira) and testing frameworks (e.g, JUnit, Cucumber).
- **Accessibility (for Collaboration).** Works as a web application. Provides support for team collaboration and resolving conflicts.
- **Adoptability.** Version 1.0 was released in 2015. Regularly maintained and updated since then. The latest version 4.14.0 was released on April 5, 2023. Detailed documentation is provided. Hands-on, assisted onboarding is provided with the enterprise version.
- **Viability (Cost).** Pricing varies with the use cases, available as SaaS or On-Premise. A community edition is provided with very limited functionalities. Pricing is based on the number of threat models created and maintained.
- **Automation.** The threat library can automatically suggest possible threats. Estimated costs of mitigations are provided, and threats are prioritized based on the model. A threat report of the system can be automatically generated.

- **Applicability.** Focus on modeling a system, identifying security threats, and identifying mitigations. Works mainly for general development, cloud-based, and enterprise development use cases.
- **Reusability.** Existing system models can be reused and imported as modules when building new models.
- **Extensibility.** The combination of support for the Open Threat Model specification and customizable control and threat information provides good extensibility to organizations.

6. Threagile (open source)

Threagile is an open-source “threat modeling with code” solution written in Go. Threagile takes YAML model definitions as input and outputs threat analysis results and data flows diagrams automagically using pre-defined rules. The YAML-based system makes it easy for developers to maintain their threat models using developer-oriented tools (i.e. their development IDE tools, or any YAML-aware editor). The built-in analysis rules cover 40+ common security concerns.

- **Scalability.** System models are saved as YAML files, as such, new models can be built on top of existing ones. Models of large-scale systems can be saved in multiple files for better modularity. No official integration with other tools in SDL, but could be easily used with other tools (e.g., Git) since all models are saved as code.
- **Accessibility (for Collaboration).** It can be run as CLI or a server with REST API. Docker images are also available. No official support for collaboration between multiple teams, but common code management and collaboration tools (e.g. GitHub, GitLab, IDEs and related tool chains) can be used. Results in JSON format enable organizations to be DevOps ready in terms of Threat Modeling.
- **Adoptability.** Last stable release Nov 2021. No update on its GitHub repo since then. Brief documentation/video made available.
- **Viability (Cost).** Free to use. Community-supported.
- **Automation.** Pre-defined threat rules can automatically suggest possible threats. System diagrams can be automatically generated from the code. A threat report of the system can be automatically generated.
- **Applicability.** Focus mainly on modeling a system and identifying security threats. Works for abstract and high-level system models in general. Nonetheless, customized threats can be added to the rule library such that they can be identified. Works for general threat modeling on a relatively abstract level.
- **Reusability.** Pre-defined threat rules can be reused for different models. Model definitions can be reused.
- **Extensibility.** Customized threats can be added to the library. Moreover, the tool can be extended by developers as it is open-sourced under a permissive license.


01010
0100000
010001
01000
0100001
SAFECode