Threat Modeling Tools: A Taxonomy

Zhenpeng Shi, Kalman Graffi, David Starobinski, Nikolay Matyunin

***Abstract*—**
Threat modeling tools allow to identify weaknesses in a system design. Yet, understanding conceptual differences between the tools is not trivial. We propose a taxonomy of threat modeling tools, and use it to compare several popular ones. Alongside, we conduct a case study to illustrate differences between the tools.

1. Introduction

Threat modeling is a structured process for identifying and understanding potential threats, as well as developing and prioritizing mitigations, such that valuable assets in the system can be protected. Performing threat modeling can not only help identify and mitigate security issues at an early stage, but also guide investment on system security. For instance, the work in [1] presents a case study of threat modeling conducted at New York City Cyber Command, a large-scale and high-risk enterprise environment. The results of the case study suggest that, when properly conducted, threat modeling is effective at the enterprise level and result in positive feedback from the involved participants.

Many threat modeling tools have been developed over the years to help with the threat modeling process. Although the tools share the same overarching goal, they vary significantly in terms of functionalities and workflow. Different tools might be effective for different threat modeling steps, such that the most suitable tool depends on the specific use case.

In this paper, we propose a set of criteria for evaluating threat modeling tools. We conceptually classify current threat modeling tools, and discuss their properties and design implications. We then apply the criteria to provide a taxonomy of popular threat modeling tools, and qualitatively evaluate the strengths and weaknesses of the tools. Note that quantitative comparison of the tools is difficult since most tools require manual configurations and design. Hence quantitative evaluation is left for future work.

Our proposed evaluation and comparison of threat modeling tools are based on the following criteria, which are explained in more detail in Section 5.

1) Model form. What form of system model is adopted by the tool (e.g., diagram vs. text-based)?
2) Scalability. How does the tool scale with the size of the system model?
3) Threat library. What libraries are used for threat identification? What are the sizes of those libraries?
4) Threat evaluation. How does the tool evaluate the identified threats?
5) Mitigation suggestion. Does the tool suggest effective countermeasures to the identified threats?
6) Availability. How does the tool run, as a desktop/web application or through command-line interface? Is the tool open-source or closed-source?
7) Development stage. Is the tool still in development or in a relatively mature state?
8) Software development life cycle (SDLC) integration. Does the tool offer integration

Published by the IEEE Computer Society
© IEEE

with other development tools regularly used in SDLC (e.g., issue tracking systems and testing frameworks)?

We envision that our proposed taxonomy of threat modeling tools will help the academic community and security practitioners to find the tools most suited for their needs. In summary, our work has three main contributions:

1) We propose a set of criteria for qualitative evaluation of threat modeling tools. The criteria cover different features and capabilities of the tools.
2) We use the criteria proposed in this paper to compare current threat modeling tools. The comparison results are summarized in a table to help understand the strengths and weaknesses of the different tools.
3) We perform a case study to investigate in more detail three popular threat modeling tools (i.e., Microsoft Threat Modeling Tool, OWASP pytm, and IriusRisk community edition). We illustrate the workflow of the threat modeling process when using each tool, and point out interesting findings.

2. Background

In this section, we give background on threat modeling, including major steps and popular methods or knowledge bases that help implement each step in practice. Threat modeling is a structured process for identifying potential threats and developing mitigations, with the purpose of protecting valuable assets in a system. We define the six steps of threat modeling process as follows:

(i) *Define* security requirements. Security requirements usually include a set of standards to meet and functionalities that are related to the system security. They help understand what data assets in the system are the most important, and to what extent those data assets need protection from threats. Some references exist to help with this step, such as the OWASP Top Ten[1], which represents the most critical 10 types of security risks to web applications. Such references provide developers with a quick and effective way to think about the fundamental security requirements that should be satisfied.

[1]https://owasp.org/www-project-top-ten/

(ii) *Model* the system. The system model is a graph of system artifacts as nodes, their grouping in trust zones and data flows between the nodes as edges. This model can be either written in a text-based model following a certain syntax to describe the model elements or visualized in a diagram form, for example, a data flow diagram (DFD). A data flow diagram is a common way to model system entities, events and boundaries of the system. It consists of four types of elements: processes, data flows, data stores, and external entities.

(iii) *Identify* potential threats based on the system model. Typical ways to implement this step include the STRIDE method and utilizing some knowledge base of threats. STRIDE is a widely adopted and mature method for threat identification. The word "STRIDE" represents six categories of threats by their names, which are: spoofing identity, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilege. STRIDE analysis is performed from the defender's perspective based on DFDs. Additionally, STRIDE provides checklists to help identify threats that fall in each category [2].

Common Vulnerabilities and Exposures (CVE)[2], Common Vulnerabilities Enumerations (CWE)[3], and Common Attack Pattern Enumeration and Classification (CAPEC)[4] are three knowledge bases that can further help identify threats. CVE contains publicly known cybersecurity vulnerabilities. CWE is based in part on CVE but less specific, and the weaknesses listed in CWE can be seen as groupings of vulnerabilities in CVE. CAPEC enumerates attack patterns from the attacker's perspective.

Threat modeling tools use the above libraries to map system artifacts (such as a database) to a set of weaknesses (such as potential for privilege escalation), which themselves are mapped to a set of mitigations. Thus, when the system developer sets a specific system artifact in the model, the corresponding weaknesses and mitigations are identified analogously.

[2]https://cve.mitre.org/
[3]https://cwe.mitre.org/
[4]https://capec.mitre.org/

(iv) *Evaluate* the threats that have been identified based on multiple aspects, such as accessibility to attackers, attack complexity, privileges required, etc. In order to quantify the impact of threats, scoring systems are employed. Moreover, since the threats are typically inter-connected, attack trees and attack graphs can be used to capture the relationships between the different threats.

To assist with evaluation, Common Vulnerability Scoring System (CVSS)[5] and Common Weakness Scoring System (CWSS)[6] are designed to provide numerical scores of known vulnerabilities or weaknesses. The scores generated by CVSS or CWSS reflect the severity, based on a few metrics. CVSS has been widely adopted with the strength of simplicity, but does not account for incomplete information, while CWSS has built-in support for incomplete information.

Attack trees are conceptual diagrams that represent attacks in a tree form, where the root node of is the attack goal, and the leaves are the ways to achieve the goal [3]. Attack graphs are directed graphs, where each vertex denotes a state (e.g., a certain vulnerability has been exploited) and each edge represents a state transition [4]. Attack trees and attack graphs are widely used for network security evaluation.

(v) *Mitigate* the potential threats according to the security requirements. The most suitable mitigations vary with the specific threat and the system where it resides. Nonetheless, some general suggestions for threat mitigation can be found in knowledge bases, for example, CVE, CWE, and CAPEC. These mitigations are often associated with the corresponding weaknesses (and system artifacts) in the library of the threat modeling tool.

(vi) *Validate* that the threats are mitigated after applying mitigation techniques. The OWASP Application Security Verification Standard (ASVS)[7] can help with this step by providing a basis for testing web application security controls.

Note that our definition of threat modeling steps is a variation of the five typical steps introduced by Microsoft on threat modeling [5]. We introduce here the additional "evaluate" step, as we believe that it is crucial to ensure that scarce security resources be allocated to the most important vulnerabilities in a timely manner.

The six steps defined in this section are to be iteratively applied during the SDLC. In some other definitions of threat modeling steps, the "define" step is omitted [6]. Since security requirements usually change over time, we believe that it is necessary to perform the "define" step each time, in order to appropriately prioritize threats.

3. Overview of Threat Modeling Tools

Many threat modeling tools have been developed over the years. In this section, we give a taxonomy of current threat modeling tools. More detailed evaluation and comparison are left for Section 5.

The tools can be categorized in several ways, for instance, non-commercial versus commercial, diagram-based versus text-based. Commercial tools are usually tailored for the needs of large companies (e.g., scalability and collaboration) but come at higher costs, while all the functionalities of non-commercial tools are available for free.

At the core of a threat modeling tool lies its library, which lists a set of system artifacts to select from, weaknesses associated to each artifact and mitigations associated to the weaknesses. A system developer can design a system using the available artifacts either through a visual diagram drawing plane or through text-based representation (e.g., in YAML format). The threat modeling tool then parses the system model and generates a report based on the artifacts used and weaknesses and mitigations associated. Some tools further provide an integration of the weaknesses and their mitigations in a ticketing system to allow the tracking of the mitigation in the software development.

In the following, we introduce and categorize threat modeling tools. We first divide the tools into two groups: non-commercial tools and commercial tools to address. While commercial tools are mostly diagram-based and share a lot of similarities, non-commercial tools can work very differently. We further categorize non-commercial tools by the form of system models that they use.

[5] https://www.first.org/cvss/

[6] https://cwe.mitre.org/cwss/cwss_v1.0.1.html

[7] https://owasp.org/www-project-application-security-verification-standard/

Non-commercial and commercial tools

The non-commercial tools introduced in this section are entirely free to use without limitations on functionalities. Moreover, all of them are open-source, except for the Microsoft Threat Modeling Tool.

Commercial tools emphasize usability, agile development, and collaboration. Almost all commercial tools are diagram-based. In the "model" step, commercial tools usually take measures to help users simplify the process of drawing the system diagram, for example, by providing questionnaires. Moreover, they typically maintain large libraries of threats and corresponding countermeasures. The tools also integrate well with issue tracking systems (e.g., Jira) and testing frameworks (e.g., JUnit). Overall, commercial tools offer a rather complete solution for threat modeling, though the accuracy of identified threats and the effectiveness of suggested countermeasures remain to be compared with other tools.

The pricing of commercial tools is usually not fixed and depends on the actual use cases. In order for users to get familiar with the tools, a community edition of the tool with limited features is often provided. Apart from the common properties mentioned above, each commercial tool has some unique features, which are described in the following subsections. Since IriusRisk, ThreatModeler, and SecuriCAD have community editions, we introduce them according to our experience with their community editions as well as the claims on their websites. For other commercial tools, our introduction is only based on their claims. We also find that the community edition of IriusRisk offers relatively more functionalities compared to that of other commercial tools, so we use it for a comparison to non-commercial tools in the following sections.

3.1. Diagram-based tools

With diagram-based threat modeling tools, threat modeling can be completed in a rather intuitive way: the tools first provide users with an easy-to-use interface to draw system diagrams (usually DFDs), then apply some rules from their library to detect potential threats. A report can be then generated as a summary of threat modeling results.

Microsoft Threat Modeling Tool. The Microsoft Threat Modeling Tool (MS TMT) is a mature tool that offers users with an intuitive way to perform threat modeling. 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 the threat conditions in the template. The threats in the default template are categorized by STRIDE. It also allows customizing the templates to suit specific use cases. The elements in the diagram, threat types, and threat properties are all customizable. Users can define multiple threat properties as needed, such as description, priority, and countermeasures.

Although a default template is offered, it is worth noting that, with its default template, MS TMT only works for the "model" and "identify" steps. It lists the threats that match the conditions in the template, and it is up to the users to decide which threats to accept and how to address them. In order to use the tool for threat evaluation and mitigation, users need to add the relevant information to the corresponding threats in the template.

OWASP Threat Dragon. OWASP Threat Dragon (OWASP TD) shares a very similar workflow to MS TMT. However, its current state is still relatively premature. For example, its library (here called threat rule engine) only suggests generic threats belonging to the six STRIDE categories, while details of the threats are left for users to fill in. It is also not as customizable as MS TMT. Its version 2.0 is planned to release in late 2021, which aims to provide an API to the CI/CD pipeline, and integrate the tool with some testing frameworks, which could be especially helpful to the "validate" step in threat modeling.

OVVL. The Open Weakness and Vulnerability Modeler (OVVL) works as a web application. According to its framework explained in [7], it has several unique features, one of which is that the CPE (Common Platform Enumeration) database is utilized for standardized naming scheme of elements in DFDs, which makes it easier to find related vulnerabilities. Moreover, its developers have trained a machine-learning model for predicting potential threats based on the attributes of a system artifact. However, the tool itself is still in a relatively early development stage. The most recent update of the tool was in April 2020, and it is unclear whether or how long the tool

will be maintained in the future.

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

ThreatModeler (commercial). The functionalities offered by ThreatModeler are similar to those of IriusRisk. However, in the "model" step, it applies process flow diagrams (PFDs). Compared to the commonly used DFDs which model the system from the defender's perspective, PFDs aim to provide a visual decomposition of the system from the attacker's perspective. It is claimed that PFDs make the system models easier to understand and more efficient to build.

SecuriCAD (commercial). SecuriCAD distinguish from other tools by featuring AI-based predictive attack simulations. It includes a potential attacker in the diagram-based system model, and uses it to simulate the possible ways that the attacker compromises the system.

3.2. Text-based tools

To build a model with text-based tools, users are required to express their model in a text-based description in a structured language, for the tools to analyze, instead of drawing the system diagram directly.

OWASP pytm. OWASP pytm is a threat modeling tool that stores the system model in Python. It requires users to create a Python object for each of the system components from its pre-defined Python classes. The tool can then generate the system diagram and identify threats from the text-based model. Its threat library contains comprehensive information about each potential threat, to mention a few, the "condition" attribute can be used for threat identification, the "likelihood of attack" and "severity" attributes can be used for simple evaluation, and the "mitigation" attribute suggests mitigation measures.

Threagile. Threagile is a text-based tool designed for agile threat modeling. The models in Threagile are represented in YAML format, which is more concise and abstract, compared to the Python representation used in OWASP pytm. Threagile also features evaluating the threats in a more rigorous way: instead of providing the possibility and severity of threats in the template beforehand, it calculates relative attacker attractiveness and data loss probability based on the model for more accurate results.

3.3. Other tools

There also exist a few tools designed with slightly different purposes or expected outcomes, such that they do not follow the six threat modeling steps defined in this paper.

TRADES Tool. The TRADES Tool is designed for supporting the TRADES methodology, which is a domain-specific modeling approach [8]. The tool adopts a new type of system diagram, which consists of general components, data, threats, and security control. Instead of identifying potential threats following some built-in rules, users need to allocate the threats to specific components in the diagram manually.

CAIRIS. CAIRIS is a platform for specifying and modeling secure and usable systems. In order to use the tool for threat modeling, users need to provide necessary information about different aspects of the system, including the assets, roles, tasks, and so on. With the system information, users can then use CAIRIS to manually model the system as DFDs, or a top-down representation similar to attack trees, and perform risk analysis and response.

Threatspec. Threatspec performs threat modeling directly on the code. In order to do that, it requires developers and security engineers write threat modeling annotations as comments inside source code, such that DFDs and reports can be generated from it.

SD Elements (commercial). Unlike other commercial tools, SD Elements does not require users to draw the system diagram, instead, it gathers information about the system by a survey. It does not have a community edition, which makes it difficult to evaluate.

Kenna (commercial). Kenna emphasizes data-driven threat modeling. It uses data science

techniques, such as natural language processing and predictive modeling, to assess, prioritize, and predict risks. It introduces its own risk score engine for quantitative threat evaluation, which is claimed to be more efficient and accurate than CVSS.

Tutamen Threat Model Automator (commercial). The Tutamen Threat Model Automater is able to leverage existing project design diagrams instead of letting users create new DFDs with the tool. Users can enter metadata as values into each of the diagram elements to describe the threat model. The tool is still in development and has not officially been released yet.

4. Case Study

In this section, we conduct a case study to investigate three representative threat modeling tools, namely a commercial and non-commercial diagram-based tool and a text-based tool. We apply the selected tools on the same sample model, compare qualitatively the results generated by them, and discuss our experience of using the tools.

For the sample application, we choose that provided by OWASP pytm. The DFD of the sample application is shown in Fig. 1. The application is a simple web application, where a user logs into the application and posts comments on the app. The app server uses the real identity database to verify the user's identity, stores the comments into the SQL database, and shows the comments back to the user when requested. In the meantime, a separate AWS Lambda service periodically cleans the database.

We also model the application above with MS TMT and the IriusRisk community edition, and compare the threat modeling process with OWASP pytm by observing the example Python file defining the model. When modeling the application with MS TMT, we use the default template.

Both MS TMT and IriusRisk use a diagram-based model. When drawing the diagram with the tools, users can select from pre-defined system artifacts or components. Fig. 2 and Fig. 3 show the interface of drawing system diagrams using MS TMT and IriusRisk, respectively. We find that IriusRisk has more pre-defined components for specific use cases, such as components in AWS and Google Cloud. Moreover, we find that the questionnaires used in IriusRisk to select the attributes of system components are more user-friendly than the manual selection of attributes required by MS TMT.

When building the text-based model in OWASP pytm, user needs to create an object for each system component, then set the attributes of the components (or leave them as default values). For example, the Python code used for defining the SQL database in Figure 1 is as follows:

```
db = Datastore("SQL Database")
db.OS = "CentOS"
db.isHardened = False
db.inBoundary = server_db
db.isSQL = True
db.inScope = True
db.maxClassification = Classification.
    RESTRICTED
```

Similarly, the code specifying the data flow from the SQL database to the real identity database is

```
db_to_secretDb = Dataflow(db, secretDb, "
    Database verify real user identity")
db_to_secretDb.protocol = "RDA-TCP"
db_to_secretDb.dstPort = 40234
db_to_secretDb.data = "Token to verify user
     identity"
db_to_secretDb.note = "Verifying that the
    user is who they say they are."
db_to_secretDb.maxClassification =
    Classification.SECRET
```

Compared to MS TMT, more configurable attributes of each system component are offered in OWASP pytm.

All the three selected tools support to generate reports of threat modeling results. For this purpose, they use the library to map the system components to weaknesses and then, if available, to mitigations. We compare the reports generated by the tools and arrive at the following conclusions:

1) In all three reports, duplicate threats are identified, though for different components. For example, OWASP pytm identifies the "privilege abuse" threat from both the web server and the real identity database. In MS TMT and OWASP pytm, this is because the tools iterate through each component to find a match of threat conditions in the library file. In other words, each component is examined independently when identifying threats.
2) Data flow diagrams include four types of components: processes, data stores, data

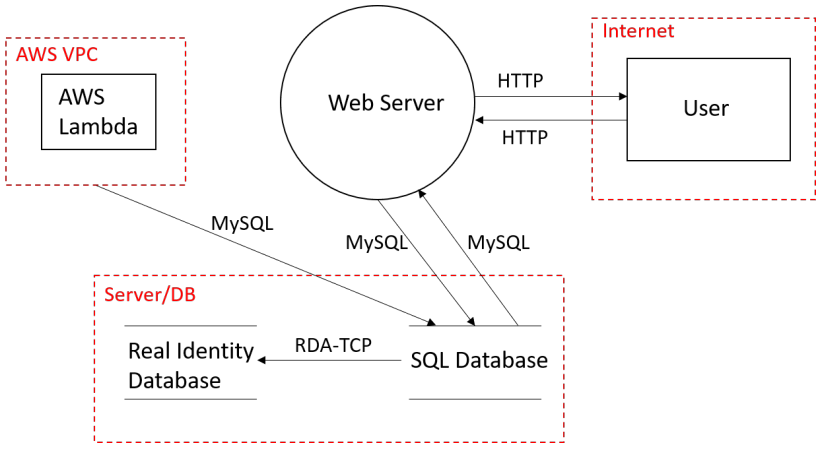


Figure 1. The sample application used for case study.

flow, and external entities. MS TMT finds threats by focusing on the data flows, while IriusRisk reasons about the other three types of components (processes, data stores, and external entities). In comparison, OWASP pytm examines all components in the diagram.

3) MS TMT provides the least information about the identified threats with its default template: no mitigations are suggested, and priority of threats are left for users to set. OWASP pytm offers some basic threat evaluation and mitigation from its threats library, and IriusRisk provides even more information (e.g., costs of countermeasures).
4) OWASP pytm lacks the functionality of tracking threats. MS TMT allows users to select the state of each threat from "not started", "needs investigation", "not applicable", and "mitigated". IriusRisk has a relatively more complete tracking system, and can also utilize some external issue trackers such as Jira.

5. Evaluation of Threat Modeling Tools

In this section, we compare five threat modeling tools by a set of criteria. The selected tools are: Microsoft Threat Modeling Tool (MS TMT), OWASP Threat Dragon (TD), OWASP pytm, Threagile, IriusRisk. Among these tools, MS TMT and OWASP TD are non-commercial diagram-based tools, OWASP pytm and Threagile are non-commercial text-based tools, and IriusRisk is a commercial diagram-based tool. Regarding IriusRisk, we choose the community edition as the basis for our comparison. This version can be used for free but has limited functionalities, for instance, no sync with issue trackers. We evaluate the tools based on the criteria introduced in Section 1, and summarize the results in a table.

Quantitative comparison between the threat modeling tools can be difficult, since the tools are mostly manual and support customization (e.g. customizing rules used for threat identification), which enables users to greatly influence the results provided by each tool. Therefore, we compare the tools in a *qualitative* manner by using the *default library and threat rules* provided by each tool.

5.1. Model form

While a diagram-based model is often more intuitive to build and understand, a text-based model also has several advantages, including: (i) version control of the model can be achieved with common tools such as Git; (ii) it allows to perform a formal and rigorous analysis; (iii) different forms of diagrams can be generated from the text-based model, for instance, DFDs and sequence diagrams.

Most tools, including MS TMT, OWASP TD, and IriusRisk, model the system in a diagram form, which makes it easier for non-experts to build the models. However, a large diagram might be difficult to maintain. As an alternative approach, some tools, such as OWASP pytm and Threagile, choose to model the system in a text form following a specific syntax, which is also

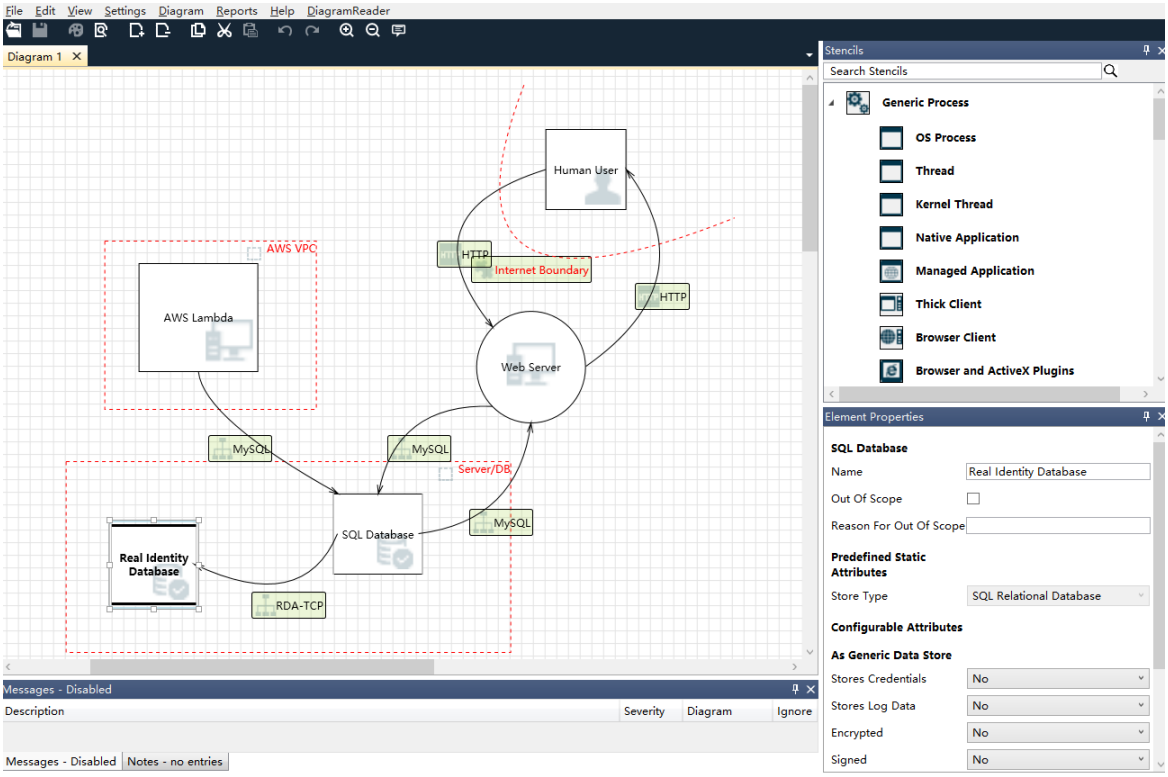


Figure 2. Draw system diagrams with Microsoft Threat Modeling Tool.

suitable for more formal analysis and deliberate extensions using text-processing scripts, as both the system model and the threat library is available in structured text formats.

5.2. Scalability

Light-weight tools might work well for simple and small systems, but fall short of handling the models in the workflow of large companies. Some tools strive for scalability, for example, by allowing users to build system models in a nested manner, or to import and export part of a diagram as a module that can be used by other models.

With MS TMT and OWASP TD, system models are usually built from scratch, and the diagrams for each system are stored in individual files, which makes it difficult to reuse parts of the existing diagrams when building new models. OWASP pytm and Threagile also lack built-in support for defining nested components or importing/exporting system modules. Nonetheless, since system components are all defined in text forms, it is possible to reuse modules of an existing model with the two text-based tools. For example, in OWASP pytm, users can import existing system modules from other files as Python objects. IriusRisk allows users to import and export parts of system diagrams as reusable modules, and supports nested components such that users can easily rearrange the diagram and define reusable modules.

5.3. Threat library

Threat modeling tools identify threats by following the associations of system components to weaknesses in their threat libraries. As a result, the threat library used by a tool decides the types and numbers of threats that can be found. In general, most of the tools allow to extend the basic library, with commercial tools often providing a much larger and sophisticated threat library.

OWASP TD only suggests six generic threats following the STRIDE category using its current threat rule engine, without indicating any specific threat. MS TMT provides more common threats. About 50 pre-defined threats are contained in its default threat library, which are categorized

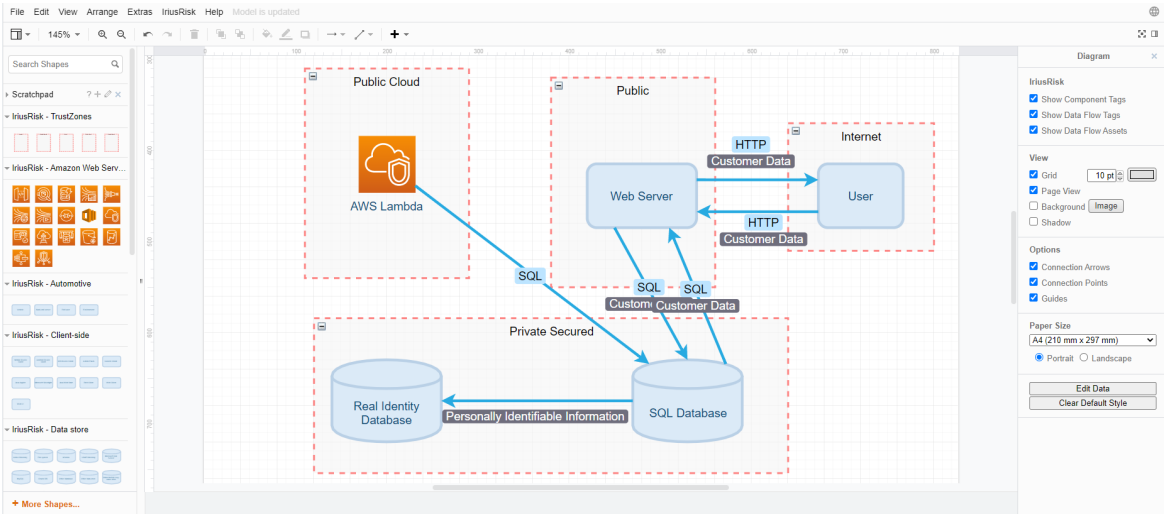


Figure 3. Draw system diagrams with IriusRisk.

by STRIDE. We have not found the sources of those threats, so the pre-defined threats were presumably proposed by the developers of the tool. Threagile covers about 40 threats from CWE, and OWASP pytm covers about 100 threats from CVE, CWE, and CAPEC. Threagile also shows the STRIDE category of each threat. IriusRisk has the largest threat library (more than 200) among the five tools, which contains not only threats in CVE, CWE, and CAPEC, but also many self-defined threats that are related to specific security standards and use cases, such as potential threats related to AWS.

5.4. Threat evaluation

Some threats are more severe or urgent than others. By evaluating threats, a tool can help users make decisions on security resource allocation more judiciously. The evaluation of a threat can be static or dynamic, where static evaluation shows pre-defined information every time the threat is identified, and dynamic evaluation involves metric calculation based on a specific model.

In MS TMT (with the default template) and OWASP TD, threats are evaluated manually by letting users set the priority or severity of an identified threat. OWASP pytm contains some pre-defined information about the likelihood of attack and threat severity in its library. Threagile not only has a pre-defined evaluation of each threat, but also uses them to calculate metrics in the context of the actual system model, specifically, relative attacker attractiveness for each component and data loss probability for each data asset are calculated. IriusRisk evaluates the identified threats and provides the information including confidentiality, integrity, availability, and ease of exploitation. The threat evaluation is based on the information stored in the threat library as well as on the optionally set priorities and criticality by the user.

5.5. Mitigation suggestion

Suggestions for threat mitigation are valuable information when users decide to handle a threat. Common threat knowledge bases such as CVE and CWE usually include some simple countermeasures to the corresponding threat, making it easier for tools to propose mitigations. Some, but not all threat libraries of the considered tools link this mitigation information to weaknesses in the library.

MS TMT (with the default template) and OWASP TD do not suggest possible countermeasures for mitigating the identified threats. OWASP pytm, Threagile, and IriusRisk provide users with simple countermeasures to the threats identified. IriusRisk also shows a general guideline to implement the mitigations as well as estimated costs of the countermeasures, which can help users decide whether to take the mitigation suggestions.

5.6. Availability of the tool

Threat modeling tools can run as desktop or web applications, or through command-line interface (CLI). Tools that run as desktop or web applications are usually more intuitive to use, while the tools with only command-line interface might be more efficient to experience users (or scripts), and consume less resources compared to desktop applications.

MS TMT is only available as a desktop application on Windows platform, while other tools are all cross-platform. MS TMT and OWASP pytm work locally, IriusRisk works as a web application, while OWASP TD and Threagile are able to work in both ways.

IriusRisk is the only commercial tool among the five tools, which makes it costly compared to other tools. Although the community edition of IriusRisk can is available for for free, its functionalities are rather limited and not suitable for real-world enterprise use. In comparison, OWASP TD, OWASP pytm, Threagile are all open-source, whereas MS TMT is free to use but not open-source. The three open-sourced tools also have the advantage to provide users with means to extend them based on their own needs.

5.7. Development stage

Some of the tools were initially released years ago, and have became relatively mature in terms of functionalities. In comparison, some newly released tools are still in development, with more feature updates planned in the future.

MS TMT was released in 2014, and IriusRisk was released in 2015. Both MS TMT and IriusRisk have been continuously updated since then, and are currently at a maintenance stage with relatively stable functionalities. Their recent updates are mostly for performance improvements and bug fixes.

OWASP TD, OWASP pytm, and Threagile are still in development. Their initial releases are relatively new, and more feature updates are planned in the future. OWASP TD has been developed since 2016 and released in 2020, with version 2.0 planned for late 2021. OWASP pytm and Threagile were both initially released in late 2020. OWASP pytm has been updated regularly since then.

5.8. SDLC integration

During the SDLC, threat modeling needs to be performed repeatedly when new changes are made to the software. Providing integration with tools in SDLC, such as issue tracking systems, can reduce the manual work and make the tool easier to use.

IriusRisk has integration with issue tracking systems like Jira, and testing frameworks like JUnit, JBehave, and Cucumber. OWASP TD supports opening models from a GitHub repository, and plans for better integration with testing frameworks. Threagile supports executing user-defined jobs (e.g. generating diagrams and reports) once the threat model file is changed on Github, it also provides editing supports in many IDEs and YAML editors to help users edit threat models.

5.9. Summary

The evaluation results of the selected threat modeling tools in this section are summarized in Table 1. From the comparison results, we note that commercial tools typically have more complete functionalities, though their commercial editions with all the features unlocked come at a higher cost. While non-commercial tools can be used for simple threat modeling, they might fall short of certain aspects when used by large companies or organizations, e.g., scalability, threat library size, or SDLC integration.

OWASP TD, OWASP pytm, and Threagile are all open-source and relatively new. Compared with closed-source tools, they are easier to extend. However, they tend to be in a relatively premature state, with certain functionalities planned but not implemented yet, as they are usually developed by a small group of people.

Most threat modeling tools leverage common threat knowledge bases, such as CWE. The knowledge bases used by OWASP pytm, Threagile, and IriusRisk not only enlarge the threat libraries, but also provide basic information about threat evaluation and mitigation, which can be valuable to users.

6. Conclusion

Our work provides a taxonomy of threat modeling tools. We categorize common threat modeling tools, and use a case study to investigate the difference between their workflows. In order to

Criterion		MS TMT	OWASP TD	OWASP pytm	Threagile	IriusRisk Community Edition
Availability	Usage	desktop app	desktop/web app	CLI	CLI/server	web app
	Cross-platform	✗	✓	✓	✓	✓
	Open-source	✗	✓	✓	✓	✗
	Free	✓	✓	✓	✓	✓
Model form		diagram	diagram	text (Python)	text (YAML)	diagram
Scalability	Nested components	✗	✗	✗	✗	✓
	Module import	✗	✗	✗	✗	✓
Threat library	Source	self-defined (STRIDE)	self-defined (STRIDE)	CAPEC, CVE, CWE	CWE	CAPEC, CVE, CWE, self-defined
	Size	~ 50	no specific threats	~ 100	~ 40	> 200
Threat evaluation		manual	manual	pre-defined	pre-defined, calculation-based	provided, but the mechanism unclear
Mitigation suggestion		✗	✗	✓	✓	✓
Development stage	Initial release	2014	Feb 2020	Sep 2020	Nov 2020	2015
	Continuously updated	✓	✓	✓	✓	✓
SDLC integration		✗	Github	✗	Github, editing support in IDEs	issue trackers, testing frameworks

Table 1. Comparison of five threat modeling tools: Microsoft Threat Modeling Tool (MS TMT), OWASP Threat Dragon (TD), OWASP pytm, Threagile, and IriusRisk Community Edition.

evaluate threat modeling tools in a standardized manner, we introduce several criteria relevant to the threat modeling process. We further compare five representative tools following these criteria, and summarize the results in a table. We expect our taxanomy to be useful for researchers or practitioners in need to choose the most suitable threat modeling tool.

A major issue found by our taxonomy is the relative immaturity of open-source threat modeling tools. Extending the functionalities of these tools would be highly valuable for the community, as only open-source tools can properly be scrutinized and vetted. Another important area left for future work is the development of standardized benchmarks, which could assist in performing quantitative evaluation of different threat modeling tools.

REFERENCES

1. R. Stevens, D. Votipka, E. M. Redmiles, C. Ahern, P. Sweeney, and M. L. Mazurek, "The battle for new york: a case study of applied digital threat modeling at the enterprise level," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 621–637.
2. R. Scandariato, K. Wuyts, and W. Joosen, "A descriptive study of microsoft's threat modeling technique," *Requirements Engineering*, vol. 20, no. 2, pp. 163–180, 2015.
3. V. Saini, Q. Duan, and V. Paruchuri, "Threat modeling using attack trees," *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 124–131, 2008.
4. K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *2009 Annual Computer Security Applications Conference*. IEEE, 2009, pp. 117–126.
5. "Threat modeling," 2021. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling
6. A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
7. A. Schaad and T. Reski, "Open Weakness and Vulnerability Modeler (OVVL) – an updated approach to threat modeling," in *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, Prague, Czech Republic - Volume 2: SECRYPT*, vol. 2, 2019, pp. 417 – 424.
8. A. Shaked and Y. Reich, "Model-based threat and risk assessment for systems design," in *Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP)*, 2021, pp. 331–338.