From automation to CI/CD: a comparative evaluation of threat modeling tools

Dimitri Van Landuyt
DistriNet
KU Leuven, Belgium
dimitri.vanlanduyt@kuleuven.be

Laurens Sion
DistriNet
KU Leuven, Belgium
laurens.sion@kuleuven.be

Walewein Philips
DistriNet
KU Leuven, Belgium
walewein.philips@student.kuleuven.be

Wouter Joosen
DistriNet
KU Leuven, Belgium
wouter.joosen@kuleuven.be

***Abstract*—As threat modeling involves the architecture-centric identification, analysis and mitigation of security threats, it is considered an essential activity of the secure development lifecycle (SDLC). Threat modeling outcomes are used to guide the mitigation and risk management process, and to determine areas of focus in later development stages (e.g., testing and verification). Traditional threat modeling methods and techniques are driven by human experts and stakeholders and mainly conducted manually. Recently, focus has shifted towards automating (parts of) the process through improved tool support.**

In this paper, we evaluate the extent to which this trend also leads to easier development pipeline integration of threat modeling tools, i.e. whether this also enables conducting threat model-centric analysis within continuous integration and development (CI/CD) build pipelines so that they can contribute to the overall development-centric quality assurance process (SecDevOps). We first articulate the key requirements for threat modeling tools to be compatible or suited for CI/CD, and then we systematically evaluate and compare seven automated threat modeling tools against these requirements. Finally, we provide recommendations towards making threat analysis tools and enablers both more suitable for integration in a CI/CD context and hence, more compatible to modern software engineering processes.

***Index Terms*—SecDevOps, CI/CD, tool integration, threat modeling**

I. Introduction

Continuous integration and development (CI/CD) activities focus on (semi-)automating the software construction process, and orchestrating this at the fine-grained level of individual code commits. In continuous development (CD), automated integration tests which are invoked within automated build pipelines provide the developer with immediate feedback. In continuous integration (CI), this even involves the immediate deployment and live integration of the code updates into a production system. While the automated testing activities integrated in a CI/CD pipeline typically focus on the execution of functional test suites (e.g., regression tests), other forms of testing such as security and scalability testing have been applied as well [1], as well as the detection of faults, flaws and vulnerabilities [1], the detection of performance regressions [2], and robustness and dependability testing [3]. When it comes to security testing, a number of approaches involve invoking code scanning [4], [5] tools to identify issues and flaws, whereas others implement static application security testing (SAST), dynamic application security testing (DAST) [6], [7] and software composition analysis (SCA).

Threat modeling involves the systematic analytical assessment of potential threats using an architectural abstraction of the system such as a data flow diagram (DFD) representation. The threat modeling process involves (i) modeling the system, (ii) identification (elicitation) of threats, (iii) threat prioritization and (iv) threat mitigation [8]. Threat modeling has predominantly been applied as a manual activity conducted by experts and development-centric stakeholders, in a collaborative workshop setting. This is reflected in the traditional tools that mainly provide practitioners with support for creating system models, to manually document threats, etc. However, threat modeling is considered a costly and time-consuming activity, which in practice is rarely repeated over the course of development and evolution of a system. A repository mining study of large collections of GitHub repositories has found little evidence of architecture-centric threat modeling actually being applied in these projects [9].

In that light, frequently re-evaluating threats is both compelling and necessary. Systems increasingly deviate from their initial designs, constantly evolve and change –among many reasons due to the adoption of agile development, continuous development and due to increasing market and delivery pressures. Ease of adoption, automation, and repeatability of threat analysis are therefore key objectives, both for economic reasons and for increasing the overall level of scrutiny.

To alleviate this, research focus in the area of threat modeling has recently also shifted towards approaches that automate parts of this process [10]–[13]. Tools that support this focus a.o. on automated extraction of system models from different artifacts (e.g., code, deployment descriptors, etc.) [14], [15], automated generation of candidate threats at the basis of consolidated knowledge bases [16], risk assessment to prioritize threats, and guide the risk mitigation and management process.

In this paper, we evaluate the current landscape of automated threat modeling tools and enablers on their readiness for integration into CI/CD build pipelines. We articulate key requirements for effective CI/CD integration of a threat modeling tool, and then we evaluate the extent to which the current automated threat modeling tools and enablers meet these requirements.

This paper is structured as follows. Section II presents the background and motivates this work, whereas Section III discusses the related work. Section IV then outlines the study

approach, after which Section V presents the main results. Section VI discusses the main findings, and Section VII finally concludes the paper.

II. Background and motivation

Threat modeling is traditionally embedded in the early stages –requirements and architectural design– of the secure development life-cycle (SDLC). By analyzing potential security risks early on, their mitigations can be inserted into the core conceptual design of the system (i.e., its software architecture). Furthermore, they can be taken care of before they materialize, and before their mitigation becomes prohibitively costly and cumbersome.

Threat modeling consists of four main steps [8], [17]: (1) modeling the system under analysis, (ii) threat identification, (iii) threat mitigation, and (iv) aggregation of outcomes and overall process reflection. Most of the practical tools focus on the first two steps: the creation of a system model and supporting the identification, documentation and prioritization of the threats.

Applying model-based analysis activities (such as threat modeling) within high-frequency, iterative development processes such as Agile development and CI/CD has proved challenging [18], [19]. This is partly due the discrepancy of focus between source code and design, due to time and delivery pressures, but also due to absence of a stable, overarching and holistic view on the system under design, and lack of convergence towards such a stable architecture.

Knowledge management and documentation quality has been an issue in software development processes for decades. Theunissen et al. [20], [21] observe and state that in the context of CI/CD, knowledge and documentation that is often implicitly or imperfectly kept in the development process becomes embedded in the tools, scripts and descriptors of the automated build pipelines. The ability of diverse tools to interoperate reliably and repeatably in a consistent and executable manner is predicated on a common understanding of the end product, which in turn is indicative of a stable architecture.

Study motivation: Automated testing tools have been successfully integrated in continuous development pipelines, but mainly focused on code and functionality such as buildability, regression and integration testing. For example, the `dependabot`[1] dependency checker has been integrated in the GitHub CI/CD framework, to perform automated Software Composition Analysis (SCA) to signal the developer when outdated library imports or packages with known vulnerabilities have been used.

As argued by Haindl et al. [22], adopting a similar approach to address more crosscutting non-functional requirements (such as security, safety, dependability or privacy) is highly compelling yet intrinsically more difficult. In this light, the recent focus shift towards automation in threat modeling tools [13], [16] is promising for many reasons. One of the stronger motivations is the ability to perform threat analysis not just in a single shot effort, but to re-evaluate the threats and risks as the system evolves over time [23], [24].

Study goal: In this study, we evaluate the extent to which current automated threat modeling tools and frameworks are fit for adoption in continuous development pipelines.

We base the design and focus areas of this comparative evaluation study on the main requirements for practical integration[2] of development tools from literature [1], [6].

III. Related work

Many threat modeling approaches and tools have emerged over time. Threat modeling approaches are generally classified [17] as being either (i) design-centric, (ii) attack-centric, (iii) asset-centric or (iv) data-centric. Existing surveys convey the diversity in tools and approaches [25]–[27].

Threat modeling exists at a fundamental point in the traditional software development life-cycle, in the transition from requirements (identification of threats) to design (selection of countermeasures or solutions) [28], [29]. However, due to the exhaustive nature of existing approaches, cost-effectiveness and repeatability have been highlighted as key concerns. For example, a design-centric approach such as STRIDE incurs the problem of *threat explosion* in which a prohibitively large amount of potential threats have to be considered, evaluated, and possibly addressed [30], [31].

For this reason, more recently, a number of approaches and tools have started focusing on the automation of specific parts of the threat modeling process. Granata et al. [10], [16] have compared a number of open-source, automated threat modeling tools, more specifically Microsoft Threat Modeling tool (MSTMT) [32], OWASP ThreatDragon [33] and SLAGenerator [34]. Tan and Garg [13] have reviewed automated threat modeling tools with specific attention to their extensibility and applicability for privacy threat analysis, or more broadly their 'amenability to custom threats', which is one important requirement for integration into specific projects. Their review focused on CAIRIS [35], Threats Manager Studio [36], Threatspec [37], pytm [38], ThreatDragon [33], and Threagile [39]. As these studies focus on comparing (automated) features of the different frameworks, they do not evaluate the overall amenability towards the specific goal of CI/CD integration.

IV. Study design

We perform a comparative evaluation of the automated threat modeling tools in the state of the art. First, Section IV-A introduces the main research questions. Section IV-B then motivates the study approach which is structured in accordance to the Goal-Question-Metric (GQM) approach [40]. After that, Section IV-C discusses the tool selection approach and Section IV-D discusses the specific evaluation approach applied to each threat modeling tool included in the study.

[1] https://github.com/dependabot/dependabot-core

[2] We define this as the *readiness* or ease of integrating a specific software quality evaluation tool into an automated CI/CD pipeline.

A. Research questions

To assess the current threat modeling tools in terms of their readiness for integration into continuous integration or development (CI/CD) pipelines, we focus on three specific and complementary research questions.

1) RQ1. Degree of automation: Automation of the analysis activity is a key enabler for integration in CI/CD. We focus on the following questions: *A number of existing tools implement a form of automation, yet which of the four steps or phases of the threat modeling process are effectively automated by the tools, and to what degree? Does this lead to automated repeatability of the entire threat analysis process?*

In this research question, we focus on the nature and degree of automation accomplished in the threat modeling tool or framework. We distinguish between automation at the different steps of the threat modeling process [8], [17]: (i) modeling the system: model extraction or synchronization, (ii) threat identification, (iii) threat prioritization, and (iv) threat mitigation support.

2) RQ2. Customization of knowledge resources: The ability to tailor often generic and abstracted knowledge resources such as threat catalogs and countermeasures catalogs is important to effective integration into build processes [41], [42]. We focus on the following questions: *Do the threat modeling tools support customizing and tailoring the knowledge resources that are being used? To which extent can the tool effectively be customized and tailored to the development context, or application domain?*

Specific applications, application domains and technologies come with specific threat types, attention points, and countermeasures. For example, when dealing with an AI/ML application, threats such as poisoning the training data set, membership inference, model stealing, should be considered [43]. When a tool suggests or supports the threat mitigation step, it may likewise be based on knowledge of application- or domain-specific countermeasures. For example, the system may perform adversarial robustness training or include specific AI guardrails[3] to mitigate some of these AI/ML threats.

3) RQ3. Integration-readiness: Although a number of tools already provide some form of automation, this does not necessarily imply that these automated features can be accessed practically from within an automated build pipeline context. We focus on the following question: *To which extent is it feasible to effectively integrate the tool or framework in a practical* CI/CD *pipeline?*

The technical and practical integration of an automated threat modeling framework in automated build scripts and CI/CD pipelines imposes a number of requirements: the tool should be scriptable and externally invokable, for example through a command-line interface (CLI). Apart from external invocation, external instrumentation is also required (load and import models and resources, start analysis activities, obtain results, etc). In addition, developer feedback or analysis outcomes should be externally processable so that it can be integrated in the feedback provided to the developer (e.g., in the reporting of a failed build job through build logs and error outputs). This specifically means that the input and output artifacts (e.g., files) should be externally readable.

[3]Filtering and redacting inputs and outputs of an AI model.

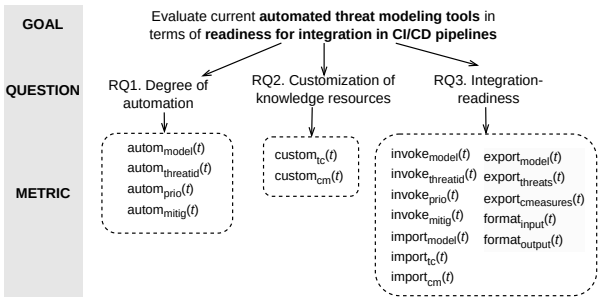

Fig. 1. The Goal-Question-Metric tree for this study.

B. GQM tree and evaluation criteria

We define a number of objectified and fine-grained criteria that each contribute to answering the research questions. These are categorical variables to distinguish between relevant cases.

*1) **Degree of automation (RQ1)**:* The following variable is defined to evaluate the extent of automation, making distinction between the different steps of the threat modeling process.

$$autom_{step}(t) = \begin{cases} \textit{Yes} & t \text{ automates threat step } i \\ \textit{Partial} & t \text{ provides partial automation of } i \\ \textit{No} & t \text{ does not automate step } i \end{cases}$$

with $step \in \{model, threatid, prio, mitig\}$, referring to one of the four threat modeling steps [8], [17].

*2) **Customization of knowledge resources (RQ2)**:* These variables express whether the tool or framework supports customization of the used knowledge resources. We distinguish between the threat catalog or threat library, and the library consisting of mitigations or countermeasures (if available), as these are play a fundamentally different role.[4] In addition, we pay attention to whether this customization can be performed externally, e.g. because this knowledge is encoded in a readable format. This additional criterion is an enabler for parallel evolution of threat and countermeasure libraries (e.g., to be able to import newer releases of such a catalog from upstream).

$$custom_{[tc|cm]}(t) = \begin{cases} \textit{NA} & t \text{ provides no resource customization support} \\ \textit{internal} & t \text{ supports customization only within the tool itself} \\ \textit{external} & t \text{ supports customization externally, e.g., config files} \end{cases}$$

The subscripts *tc* respectively *cm* refer to the ability to customize the threat library or the countermeasure library.

[4]This is related to the key distinction between the '*problem space*' and the '*solution space*', or between requirements and design.

*3) **Integration-readiness (RQ3)**:* Finally, to evaluate the ability to practically integrate a threat modeling tool t in an automated CI/CD pipeline, we define the following variable:

$$import_{[model|tc|cm]}(t) = v, \text{with } v \in \{true, false\}$$

This expresses whether an input model, threat catalog or countermeasure catalog can be loaded through external instruction (e.g., via command-line options).

A similar variable is defined for output artifacts:

$$export_{[model|threats|cmeasures]}(t) = v, \text{with } v \in \{true, false, NA\}$$

Using this variable, we evaluate whether the tools allow exporting the output model (DFD), the identified threats and the selected countermeasures through external instruction.

We furthermore pay attention to the input and export formats of these resources, more specifically whether they can be parsed and interpreted externally, making distinction between binary/proprietary outputs, or externally readable and more open formats (e.g., a standardized format, `csv`, `json`, `xml`, `yaml` or text files).

$$format_{[input|output]}(t) = v, \text{with } v \in \{binary, textual, NA\}$$

With binary formats, there is a strong dependency on the tool or framework, which is then also required to externally parse and process the outcomes of the analysis. With textual and open formats, developers can create custom parsers or scripts to process these artifacts.

We finally look at the ability to externally invoke the threat analysis:

$$invoke_{step}(t) = \begin{cases} Yes & t \text{ can be invoked externally} \\ No & t \text{ can not be invoked externally} \\ NA & \text{step not automated} \end{cases} \quad (1)$$

again with $step \in \{model, threatid, prio, mitig\}$. External invocation is an enabler for the actual integration of the threat modeling step in CI/CD build scripts.

C. Tool selection

We identify a list tools and frameworks at the basis of recent comparative survey studies [10], [16], [26]. This list is augmented with results of additional searches in both academic literature and developer resources.[5] Table I shows the resulting set of tools. We define three inclusion criteria to filter this set of tools relevant for this study.

IC1: The threat modeling tool or framework implements a form of automation in (at least one of) the main steps[6] of the overall threat modeling process.

IC2: There is evidence that the tool is actively used in practice and/or under active development.

[5]Most notably also the information shared within the `threat-modeling` channel in the OWASP Slack community (https://owasp.slack.com).

[6]We exclude tools such as Threatspec or CoreTM that only implement automation of the reporting, e.g. that extract a report based on manually-obtained analysis outcomes.

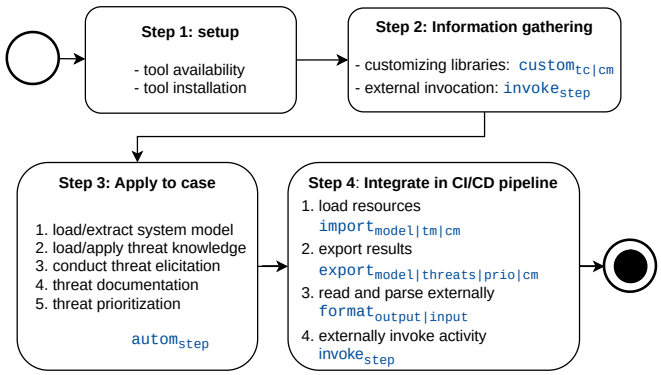

Fig. 2. Overview of the workflow executed per tool and the evaluated variables/metrics.

TABLE I
OVERVIEW OF THREAT MODELING TOOLS.

	Inactive	Active
Open source	Trike [44] CORAS [45] TAM^2 [47] OVVL [49] SLAGenerator [34]	OWASP Threat Dragon [33] Deciduous [46] CoreTM [48] Threatspec [37] CAIRIS [35] OWASP pytm [38] Threagile [39] TicTaaC [50] SPARTA [12]
Commercial (Free to use)		Microsoft Threat Modeling Tool [32] Threats Manager Studio [36]
Commercial (Paid)		IriusRisk ThreatModeler SecuriCAD SD Elements Kenna Tutamen Threat Model Automator ThreatGet itemis SECURE STRIDE-GPT
Unavailable		ThrEma [11]

tool Underlined tools do not meet criterion **IC1** for inclusion in the study.

IC3: The tool is freely usable (open source tools or tools without any licensing costs) and thus accessible for experimentation.

Applying these criteria leads to the resulting set of tools included in this study, as listed and summarized in Table II.

D. Evaluation approach

To ensure fairness and reproducibility of the study outcomes, we define a systematic workflow which is used to test the different capabilities on a per-tool basis. As depicted in Figure 2, the workflow consists of four subsequent steps,[7] and in each step, different variables, metrics and criteria are evaluated.

[7]We adopt a best-effort approach: when one step fails or is executed incompletely, findings are noted, and the subsequent steps are executed.

TABLE II
OVERVIEW OF THE INCLUDED TOOLS IN THIS COMPARISON STUDY, TOGETHER WITH THE SUMMARY OF THEIR MAIN AUTOMATION CAPABILITIES.

Abbrev.	Tool name	Description	Ver.	Ref.
MSTMT	Microsoft Threat Modeling Tool	Graphical tool to create DFDs, threat generation at the basis of templates. Threat prioritization takes into account specific attributes of elements.	7	[32]
pytm	OWASP pytm	pytm is a code-centric threat modeling framework in which system models are expressed programmatically. Pytm generates a graphical representation of both DFDs and sequence diagrams. Pytm performs threat generation at the basis of knowledge encoded in the framework.	1.2.0	[38]
CAIRIS	Computer Aided Integration of Requirements and Information Security	CAIRIS is an open-source platform aimed to harmonize and reconcile usability requirements and risk analysis. It has been developed since 2012.	2.3.8	[35]
TMS	Threats Manager Studio	Threats Manager Studio is a free tool based on the open source project entitled 'Threats Manager Platform'. TMS was released in 2020 and implements the vNext threat modeling process. The main focus is on flexibility, and integration.	2.2.1	[36]
SPARTA	Security and Privacy Threat Modeling for Automated Threat Elicitation and Risk-driven Threat Prioritization	SPARTA is an Eclipse-based threat modeling tool which implements automated threat elicitation, automated FAIR-based risk assessment based on enriched models. One research prototype of SPARTA named CTAM is integrated in CI/CD.	2022.1	[23], [24]
Threagile	Threagile: Agile Threat Modeling	Threagile is open-source toolkit for agile threat modeling. Architecture is expressed in `yaml` This model is subjected to a number of automated checks expressed as risk rules.	0.9	[39]
TicTaaC	Threat modeling-as-a-Code in a Tick	TicTaaC is an automated threat modeling toolkit which can be integrated into a SecDevOps process. It supports `yaml`-like specifications of data flows and uses externalized and customizable threat libraries to generated possible threats.	1.3.0	[50]

Step 1. Setup Information about the tool and its capabilities is gathered (e.g., development status and frequency, current availability), and the tool is installed and prepared.

Step 2. Information gathering. In this step, specific automation capabilities of the tool are specifically sought for, with specific attention to the capability to externally invoke the tool, parse the input and output artifacts, and to customize the knowledge resources.

Step 3. Apply automation in application case. In this step, the automated features are applied to a specific, realistic and consistent application case named DocProc.[8] This step allows obtaining more concrete results for $autom_{step}$.

Step 4. Integrate in CI/CD pipeline. In this phase of the experimental approach, the practical integration of the tool in a CI/CD script is tested to identify the extent of support for import, export, and external invocation of the tools.

TABLE III
OVERVIEW OF THE AUTOMATION CAPABILITIES OF EACH THREAT MODELING TOOL (RQ1).

Metric	MSTMT	pytm	CAIRIS	TMS	SPARTA	Threagile	TicTaaC
$autom_{model}(t)$	○	○	◐	○	◐ *	○	○
$autom_{threatid}(t)$	●	●	○	● *	●	●	●
$autom_{prio}(t)$	○	●	○	● *	●	●	○
$autom_{mitig}(t)$	○	○	○	○	○	○	○

● Yes
◐ Partial
○ No
* Support through non-default extension.

V. RESULTS

A. RQ1. Threat modeling automation

The overall results for RQ1 summarizing the automation capabilities are presented in Table III. We discuss these findings on a per-tool basis.

- MSTMT implements a templating mechanism which allows capturing threat knowledge specific to an application type. The specification of `GenerationFilters` supports expressing the conditions when the threat is considered applicable. For example, the Microsoft template repository on GitHub[9] provides a dedicated template for Azure Cloud applications and one specialized template for medical devices. This template is used to generate viable threats. While MSTMT does automate the threat identification step, it lacks automation in the other steps (as shown in the second column of Table III).
- The pytm framework is capable of identifying threats autonomously at the basis of the programmatically-established system model and threat definitions in the pytm threat library. Generated threats are documented in detail, including an assessment of severity and likelihood of attack. Thus, pytm automates threat identification and prioritization (third column of Table III).
- CAIRIS users can opt to define resources and assets in a relational manner, and have the tool generate an initial, incomplete DFD. As such extracted DFD lack actual data flows, this is considered partial support for the overall system modeling step. CAIRIS does not automate other steps (as shown in the fourth column of Table III).
- Threat Manager Studio (TMS) provides the '*Automatic*

[8]This case refers to a B2B document processing Software-as-a-Service application which originates from industry collaboration that has been used in a number of research activities [51]–[53]. Detailed specifications and security requirements can be consulted in [54].

[9]https://github.com/microsoft/threat-modeling-templates

Threat Generation' extension[10] which provides capabilities to automatically generate and prioritize threats (fifth column of Table III). This tool furthermore distinguishes between threat types (potential threats as encoded in a reusable knowledge base) and threat events (more concrete threat instances that may be applicable to the system model).

- SPARTA (sixth column of Table III) generates threats using threat catalogs that include model query patterns for identifying applicable threats. Based on model enrichments (e.g., assets, data subjects, existing countermeasures or mitigations), it automatically adjusts the priorities of the threats.
- Threagile loads the relevant data for its analysis from a `yaml` file that specifies the different data assets, technical assets (including their communication), and trust boundaries. From this input, Threagile automatically generates the threats based on its internal catalog (or other plugins) and performs a risk-based priority assessment which evaluates the confidentiality, integrity, and availability attributes of the assets (cf. the seventh column of Table III).
- TicTaaC loads data flow diagram specifications (`yaml` files) and applies threat library knowledge to identify threats in an automated fashion. The specification of each threat encoded in the library includes an elicitation expression, which is used in the threat generation process. For example, the logical expression "`target.type == database`"[11] is used to create `database-broad-development-team-access` threats which represent situations in which databases are too broadly accessible within the development team. TicTaaC additionally allows specifying mitigations in separate files (`mitigations.yml`) which are taken into account during threat identification. However, these are to be specified manually and lack any form of automation (cf. the final column of Table III).

RQ1 findings. The studied tools mainly automate the threat identification step, and to a lesser degree also threat prioritization, but automation in the other threat modeling steps is currently lacking.
Being able to automatically synchronize and update the system models ($model$) at the basis of source code commits, and to provide the developer with threat mitigation ($mitig$) support are highly desirable in a CI/CD context.

B. RQ2. Customization of knowledge resources

Table IV summarizes the outcomes of evaluating the tools in terms of the customization support they provide for the types of threats (threat catalog or tc), or the types of countermeasures they can express or suggest (cm).

- MSTMT provides external customization support for both threat types and mitigations (cf. second column of Table IV). The templates (`.tb7` files) are formatted in `xml` which can as such be parsed and manipulated externally, and a customized threat template can be constructed using the Template Editor which is part of the MSTMT framework. The template mechanism also provides support for expressing countermeasures, which are part of the same template files. Currently, three distinct templates are officially supported and packaged as part of MSTMT: the `default` template, one for Azure applications, and one for medical devices. Other forks and variants have also come to existence and have been shared in external repositories.[12]
- pytm loads a separate threat library, which is located in the `/threatlib/threats.json` file of the pytm installation. The library consists of a number of different threats which has grown and evolved over time: while the first version of this `json` file consisted of 54 threat types, currently it has 103. The threat library also contains information related to possible mitigations or countermeasures. As `json` is an open and externally-readable and -manipulable file format, we conclude that external customization is well-supported in pytm (cf. third column of Table IV).
- CAIRIS packages a default threat library (in the `ics_tv_types.xml` file) which is structured as an `xml` document. This library currently consists of nine example threat types. In addition, two other example threat libraries are provided: `cwecapec_tv_types.xml` and `owasp_tv_types.xml`. These are relatively limited and haven't been updated since the initial release. Given that an `xml`-based format is used, these libraries can be extended and customized.[13] As also highlighted in the fourth column of Table IV, CAIRIS does not support countermeasure or mitigation libraries.
- Threats Manager Studio (TMS) does not provide any default libraries. Knowledge about threats and mitigations is encapsulated in templates. The TMS website[14] provides one such custom template for Azure PaaS Core. To each threat type encoded in such template, a number of possible mitigations can be attributed. The list of threat types and known mitiga-

TABLE IV
OVERVIEW OF THE CUSTOMIZATION CAPABILITIES OF EACH THREAT MODELING TOOL (RQ2).

Metric	MSTMT	pytm	CAIRIS	TMS	SPARTA	Threagile	TicTaaC
$custom_{tc}(t)$	●	●	●	●	●	◐	●
$custom_{cm}(t)$	●	●	○	●	●	●	●

● External customization support (e.g., config files).
◐ Internal customization support (strong dependency on the tool).
○ No customization support (NA).

[10]https://threatsmanager.com/training/advanced/extensions/auto-threat-generation/

[11]Taken from https://github.com/rusakovichma/TicTaaC/blob/master/src/main/resources/threats-library/default-threats-library.yml.

[12]E.g., https://github.com/AzureArchitecture/threat-model-templates/ for Azure, https://github.com/tmart234/mobile-application-threat-model-template for mobile applications, https://github.com/arvindpj007/Threat-Modeling-CISCO-OpenConnect for VPNs.

[13]The XML document type definitions (DTD) for this can be found through the CAIRIS documentation site (https://cairis.readthedocs.io/en/latest/io.html).

[14]https://threatsmanager.com/downloads/templates/

tions can be adjusted and changed through a dedicated user interface feature, and thus internal customization is possible. Furthermore, as these templates (`.tmt` file) are in fact archive file that consist of one `json` (`threatmodeltemplate.json`) and one `xml` meta-data file (`[Content_Types].xml`), we conclude that external customization is also theoretically possible (cf. fifth column of Table IV).

- SPARTA's CLI only requires a single input model file (`--input`), which then has to contain the necessary references to the accompanying threat type and countermeasure catalogs that are used for the threat elicitation and the applied countermeasures. The threat type and countermeasure catalogs are structured as `xml` files which allows for external customization (cf. sixth column of Table IV).
- Threagile comes with a built-in set of risk rules that are used to generate the threats. The generation logic can be extended, but this requires the programmatic creation of risk rule plugins within the framework, which is considered a form of internal customization, as highlighted in the seventh column of Table IV. The countermeasures are partly predefined as attributes, and partly as a textual description in the `yaml` file when keeping track of the status of the identified risks.
- TicTaaC (final column of Table IV) comes with a default threat library, but developers can specify additional threats in separate files and these can be loaded using the `--threatsLibrary` command-line parameter. The `--mitigations` parameter is used to load project-specific mitigations, which as discussed above, are to be constructed manually.

RQ2 findings. Nearly all evaluated threat modeling tools support customization of both the threat and countermeasure knowledge resources. This feature allows tailoring to a specific application context in a reusable manner. However, apart from SPARTA, none of the tools maintains a clear segregation between both. For example, a number of tools introduce a templating mechanism, in which logical expressions encapsulate both information about threat types (problems) and threat mitigations (solutions). In those cases, both types of knowledge are tightly interdependent and can not independently evolve.
A standard in encoding of threat or countermeasure knowledge is currently lacking, which means that efforts made to customize resources for a specific project risk being tool-locked-in, and that the information used by these tools does not easily port or compare.

C. RQ3. Integration-readiness

Table V summarizes the results of evaluating external invocability of the automated features. Table VI summarizes the input (import) and output (export) capabilities of the different threat modeling tools. Finally, Table VII summarizes the external readability of these inputs and outputs.

- MSTMT is a graphical and user-interactive tool, and lacks support for external invocation to initiate a specific threat modeling step or activity, or to import/export resources or threat analysis outcomes (as shown in the second column of Table V). MSTMT supports manually exporting threat models to curated Microsoft Word files that use a specific template (`.dotx` file). For this support, it provides a dedicated Microsoft Word plug-in.[15] The website reports a command-line tool to perform massive migration of Microsoft Threat Modeling Tool files, but this is not available at the time of writing. Most notable is the DevOps plugin which provides the capability to report directly to the Azure DevOps service (keeping track of issues and mitigations), and provides overall project management facilities. This feature is however heavily UI-

TABLE V
OVERVIEW OF THE INTEGRATION-READINESS RESULTS FOR EACH THREAT MODELING TOOL (RQ3).

Metric	MSTMT	pytm	CAIRIS	TMS	SPARTA	Threagile	TicTaaC
$invoke_{model}(t)$	NA	NA	○	NA	NA	NA	NA
$invoke_{threatid}(t)$	○	●	NA	○	●	●	●
$invoke_{prio}(t)$	NA	●	NA	○	●	●	NA
$invoke_{mitig}(t)$	NA	NA	NA	NA	NA	NA	NA

● Yes.
○ No.
NA Not applicable since the step is not automated.

TABLE VI
OVERVIEW OF TOOL CAPABILITY TO EXTERNALLY INVOKE IMPORT OR EXPORT RELEVANT ARTIFACTS (RQ3).

Metric	MSTMT	pytm	CAIRIS	TMS	SPARTA	Threagile	TicTaaC
$import_{model}(t)$	○	○	●	○	●	●	●
$import_{tc}(t)$	○	○	●	○	●	○	●
$import_{cm}(t)$	○	○	NA	○	●	●	●
$export_{model}(t)$	○	●	○	○	○	●	●
$export_{threats}(t)$	○	●	●	○	●	●	●
$export_{cmeasures}(t)$	○	●	○	○	○	○	●

● True (supported).
○ False (not supported).

TABLE VII
SUMMARY OF THE EXTERNAL ACCESSIBILITY OF THE ARTIFACT FORMATTINGS PER THREAT MODELING TOOL (RQ3).

Metric	MSTMT	pytm	CAIRIS	TMS	SPARTA	Threagile	TicTaaC
$format_{input}(t)$	NA	NA	●	○	●	●	●
$format_{output}(t)$	NA	●	●	○	●	●	●

● Textual.
○ Binary.
NA Not applicable because input/output support is lacking.

[15] https://downloads.threatsmanager.com/latest/TMPlatform_ReportingAddIn.msi

driven, and not accessible from an external (command-line) interface, and thus less suited for reporting or management within an automated CI/CD build pipeline. The overall lack of import/export support is reflected in the second column of Tables VI and VII.

- The threat analysis and prioritization capabilities of pytm can be invoked externally through Python scripts, as shown in the third column of Table V. The import/export capabilities are summarized in the third column of Table VI. The pytm framework relies on a specific Python-encoded representation of the system model, and it does not support importing models nor does it support extraction from other artifacts or code bases. Through the `--dfd` command-line flag, the directed graph representation of the system model itself can be exported. This output can then be parsed and further processed with external tools such as `dot`.[16] The pytm framework also supports exporting sequence diagrams using the `--seq` flag. Additionally, pytm supports exporting results to a `json` file (`--json`), a sqlite database (`--sqldump`), a list of threats (`--list`), a list of elements (`--list-elements`) and a Markdown report (`--report`). It also allows querying the system model for specific information (`--describe`). These are externally-parseable file formats and data structures, as shown in the third column of Table VII.
- CAIRIS runs as a service or daemon and provides a well-documented REST API for remote authentication and invocation.[17] However, the actual DFD generation/extraction capabilities discussed above can not be invoked externally in CAIRIS, as shown in the fourth column of Table V. The `cimport.py` script supports loading a model; a complementary script (`cexport.py`) is provided to programmatically export the threats and risk outcomes (as shown in the fourth column of Table VI). CAIRIS provides draw.io integration to specify importable data flow diagrams, but only when the `cairis_dfd` library/stencils are used in the drawio.io diagramming tool. These are `json`-based representations and thus textual in nature. The `.cairis` file type stores models in a binary representation, but CAIRIS also supports exporting to `xml` documents[18] (cf., the fourth column of Table VII).
- Threat Manager Studio (TMS) supports importing MSTMT models through an extension that is available through the website.[19] It further lacks support for invoking threat modeling steps and import/export, as shown in the fifth column of Tables V and VI. TMS uses binary file formats, as indicated in the fifth column of Table VII.
- SPARTA provides a jar-file (`sparta-cli.jar`) to enable the command-line execution of its threat elicitation engine (independent of the graphical UI). This is highlighted as such in the sixth column of Table V. The engine is invoked with the input model (`--input`), the threat catalogs to which the input model refers, and the requested output format (`--outcsv`, `--outxlsx`). While the threats can be exported in multiple formats, the model itself can only be exported via the UI. (cf., the sixth column of Table VI). For all resources, `xml` encodings are used (as shown in the sixth column of Table VII).
- As indicated in the seventh column of Table V, Threagile can be started from the command line (or as a REST web service). The threat catalog is not importable but requires the plugins. The resulting model (`-generate-data-flow-diagram`, `-generate-data-asset-diagram`) and threat analysis results (`-generate-risks-json`, `-generate-risks-excel`) can be exported in multiple formats (cf. seventh column in Table VI). Threagile loads all the information to perform threat analysis and risk assessment from a `yaml` file. This file contains a textual representation of the model and the countermeasures (cf. seventh column in Table VII).
- TicTaaC supports external invocation via the command-line interface (CLI),[20] (cf., the final column of Table V). It can programmatically load system models using the `--threatModel` command-line parameters. In terms of export, the framework mainly allows exporting an all-encompassing summary report (data flow diagram with threats and possibly mitigations), as either a `html` or `json` document structure. This is done by setting the `--outFormat` parameter and specifying the output file (`--out`). These findings are presented in the final column of Table VI. All inputs (data flow diagram specification, threat library and countermeasures) are encoded as textual `yaml` files (cf. final column of Table VII). Highly relevant for the CI/CD integration requirements is the `--failOnThreatRisk` command-line parameter of TicTaaC which accepts a risk level and reports failure of the threat modeling analysis job when actual threats were found above that risk level.[21] When integrated in a CI/CD pipeline, this will then be trigger failure of the entire build job (broken build pipeline), and be reported to the developer responsible for the source code commit.

RQ3 findings. A number of the tools provide relevant import and export functionalities, but in the tools that are mainly oriented towards human users (e.g., MSTM), this support is not suited for programmatic processing in CI/CD pipelines (e.g., Markdown or Microsoft Word files).
The tools that do provide import and export capabilities all employ data formats that are externally readable and processable. This is not entirely surprising, given that we have applied a selection strategy that emphasizes open and accessible tools (inclusion criterion **IC3**).

VI. DISCUSSION

Section VI-A discusses threats to validity of this study, and Section VI-B then reflects on the expressiveness and exhaustiveness of the different approaches to model threat

[16]For example, '`tm.py --dfd | dot -Tpng -o dfd-output.png`' exports the directed graph representation and stores it in the '`dfd-output.png`' file.

[17]https://cairis.readthedocs.io/en/stable/api.html

[18]In compliance to the CAIRIS DTD, cf. https://cairis.org/dtd/cairis_model.dtd.

[19]The DocProc DFD experimentation however indicated that the current import functionality itself is flawed.

[20]https://github.com/rusakovichma/TicTaaC/wiki

[21]Specifically, TicTaaC then outputs these threats to `stderr` which is interpreted and reported by CI/CD runners as a breaking issue.

types and countermeasures. Finally, Section VI-C elaborates on interoperability requirements and data exchange formats for threat modeling.

A. Threats to Validity

This section elaborates in the different threats to validity (construct, internal, external).

A first threat to validity is in the selection of the metrics used to assess the readiness for integration of automated threat modeling in CI/CD pipelines, which may not the most appropriate for measuring integration-readiness. To alleviate this threat to validity, this study relies on the GQM framework to decompose the integration-readiness into specific questions and metrics in a structured and reasoned manner.

A second threat to validity is related to the validity of the results. These may be inaccurate because of lacking documentation or insufficient experience with these specific threat modeling tools. For each tool, a detailed motivation is provided to motivate the scores that were assigned in the result tables (Tables III to VII).

A final threat to validity involves the generalizability of the results. A set of inclusion criteria was defined, specifically to select the tools that implement automation (**IC1**), are under active use or development (**IC2**), and are available for experimentation (**IC3**). This subset of threat modeling tools however is not indicative of the complete landscape of threat modeling tools, which includes a significant amount of commercial tools. The observations made in this paper are not generalizable to tools that were excluded from this study.

Finally, we remark that it should not necessarily be within the expectations that a single tool or framework will necessarily implement all the stated and evaluated requirements: a complementary set of tools may accomplish the ideal coverage of the different steps and aspects of the overall process.

B. Expressiveness and exhaustiveness

This study mainly focused on the integration of threat modeling tools and mainly of their automation capabilities for system model specification, threat elicitation, prioritization, and mitigation. This is however a coarse-grained assessment of such features which does not convey well the variation that we observed in both expressiveness and exhaustiveness, in (1) modeling of the system and which additional information can be expressed in these models; (2) the threat type catalogs, the granularity of the modeled threats, and the level of detail in the threat type criteria; and (3) the countermeasure catalogs, the variation in types of countermeasures that can be expressed, how and how countermeasures affect threats and risks. Such differences were not directly taken into account in this study, and as such, more in-depth comparison is considered part of future work.

C. Interoperability of threat modeling data

This study (RQ2) assesses whether the different input and output representations supported by the tools are in textual or binary formats, and shows that for the studied tools, a textual and accessible format is predominantly used (`json`, `yml`, `xml`, etc). Despite external accessibility, we did observe large diversity in both input and output formats. In fact, each tool comes with its own encoding and data schema for these artifacts, and tools model threats at different levels of granularity and abstraction (e.g., threat types in MSTMT and SPARTA and CAPEC and CVE entries in pytm). This diversity negatively affects the extent to which these results can be reused or integrated in other tooling as part of the CI/CD build pipeline. The overall lack of standardization in threat model formats is an additional factor that hinders the use, integration and interchangeability of different tools in the CI/CD context. To make these different tools interoperate, complex translation, import and wrapper logic will have to be established.

One noteworthy initiative is the Open Threat Modeling Format (OTM).[22] This format has been specified by IriusRisk, and is currently only supported by their tools. Furthermore, this initivative advocates encoding a number of different element types (the system model, threats, countermeasures, assets, trust zones) into a single `json` file, which is inadvisable mainly for reasons of independent evolution of these artifacts and separation of concerns.

VII. Conclusion

This paper evaluates seven automated threat modeling tools and frameworks against the requirements for practical integration into CI/CD pipelines. We focus on the ability to (i) customize the knowledge resources (threats and countermeasures) to the specific application context at hand, and (ii) the external controllability either through remote invocation, or via a script/command-line interface (i.e. the ability to invoke, import, export, process, version inputs and outputs such as system models, threats, countermeasure suggestions, etc).

In terms of potential, we identify pytm, SPARTA, Threagile and TicTaaC as the most promising threat modeling implementations for CI/CD. Notably, for SPARTA, a complementary prototype called CTAM practically integrates its engine into GitLab CI/CD build pipelines [23], [24]. TicTaaC provides a dedicated command-line option (`--failOnThreatRisk`) which forces the overall build job to fail when certain threats are found to be above a certain threshold (e.g. High severity). This mechanism then impedes further development until these threats have been appropriately mitigated.

We subject these tools to expectations and requirements that they were not originally intended for. Notably, the GUI-based tools are meant for manual operation, and in those cases, any support for automation has been a late addition, or even an optional extension. We make the following recommendations for threat modeling tool and framework developers:

1) Broaden the automation capabilities, including the capability to extract or derive system models from code and to suggest specific mitigations and countermeasures as part of the CI/CD build reporting.

[22] https://github.com/iriusrisk/OpenThreatModel

2) Allow for separate creation and specification of knowledge resources, and provide support for customization of these towards specific context (e.g., application domains).
3) Maintain a separation between threats (problem space) and countermeasures (in separate catalogs or libraries), as it will allow for independent evolution and reuse.
4) Prepare the tools for programmatic, external access and invocation, with support for dynamically selecting and loading different and customized knowledge resources.
5) Publish open data schema specifications for encoding and parsing the different threat modeling artifact (i.e., system models and templates, threat catalogs, countermeasures, etc) and to allow for broader interoperability.
6) Collaborate towards industry standards for encoding and parsing the different threat modeling artifacts.

We particularly envision the emergence of an ecosystem of complementary and interoperable tools that each implement one relevant aspect of the threat modeling process in a modular and self-contained manner (e.g., dedicated system model extractors from code, a threat elicitation engine, etc). These utilities can then be composed and concatenated to establish the desired degree of integration in a more customized and project-specific manner. We observed this in pytm which relies on the `dot` utility to visualize DFDs, and similarly we envision that other complementary compositions of such enablers can be made in accomplishment of the broader goal of effective CI/CD support. To attain this broader vision however, emphasis on openness, interoperability and standardization, as discussed above, are considered essential prerequisites.

Acknowledgment

This research is partially funded by the Research Fund KU Leuven, and by the Cybersecurity Research Program Flanders.

References

[1] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices," *IEEE access*, vol. 5, pp. 3909–3943, 2017.

[2] F. Erculiani, L. Abeni, and L. Palopoli, "uBuild: Automated Testing and Performance Evaluation of Embedded Linux Systems," in *Architecture of Computing Systems–ARCS 2014: 27th International Conference, Lübeck, Germany, February 25-28, 2014.* Springer, 2014, pp. 123–134.

[3] L. Zhu, D. Xu, A. B. Tran, X. Xu, L. Bass, I. Weber, and S. Dwarakanathan, "Achieving reliable high-frequency releases in cloud environments," *IEEE Software*, vol. 32, no. 2, pp. 73–80, 2015.

[4] M. Marandi, A. Bertia, and S. Silas, "Implementing and automating security scanning to a devsecops ci/cd pipeline," in *2023 World Conference on Communication & Computing (WCONF)*. IEEE, 2023, pp. 1–6.

[5] G. Hao, F. Li, W. Huo, Q. Sun, W. Wang, X. Li, and W. Zou, "Constructing benchmarks for supporting explainable evaluations of static application security testing tools," in *2019 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, 2019, pp. 65–72.

[6] T. Rangnau, R. v. Buijtenen, F. Fransen, and F. Turkmen, "Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines," in *24th International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2020, pp. 145–154.

[7] A. M. Putra and H. Kabetta, "Implementation of devsecops by integrating static and dynamic security testing in ci/cd pipelines," in *2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM)*. IEEE, 2022, pp. 1–6.

[8] Z. Braiterman, A. Shostack, J. Marcil, S. de de Vries, I. Michlin, K. Wuyts, R. Hurlbut, B. S. Schoenfield, F. Scott, M. Coles, C. Romeo, A. Miller, I. Tarandach, A. Douglen, and M. French, "Threat Modeling Manifesto," https://www.threatmodelingmanifesto.org/, Nov. 2020.

[9] B. Gruner, S. T. Heckner, T. Sonnekalb, B.-E. Bouhlal, and C.-A. Brust, "Finding a needle in a haystack: Threat analysis in open-source projects," 2024.

[10] D. Granata, M. Rak, and G. Salzillo, "Automated threat modeling approaches: Comparison of open source tools," in *International Conference on the Quality of Information and Communications Technology*. Springer, 2022, pp. 250–265.

[11] F. De Rosa, N. Maunero, P. Prinetto, F. Talentino, and M. Trussoni, "ThreMA: Ontology-Based Automated Threat Modeling for ICT Infrastructures," *IEEE Access*, vol. 10, pp. 116 514–116 526, 2022.

[12] L. Sion, "Automated threat analysis for security and privacy," 2020.

[13] K. Tan and V. Garg, "An analysis of open-source automated threat modeling tools and their extensibility from security into privacy." USENIX, 2022.

[14] M. Abi-Antoun, D. Wang, and P. Torr, "Checking threat modeling data flow diagrams for implementation conformance and security," in *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, 2007, pp. 393–396.

[15] P. Benedusi, A. Cimitile, and U. De Carlini, "A reverse engineering methodology to reconstruct hierarchical data flow diagrams for software maintenance," in *Proceedings. Conference on Software Maintenance-1989*. IEEE, 1989, pp. 180–189.

[16] D. Granata and M. Rak, "Systematic analysis of automated threat modelling techniques: Comparison of open-source tools," *Software quality journal*, vol. 32, no. 1, pp. 125–161, 2024.

[17] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[18] D. S. Cruzes, M. G. Jaatun, K. Bernsmed, and I. A. Tøndel, "Challenges and experiences with applying microsoft threat modeling in agile development projects," in *2018 25th Australasian Software Engineering Conference (ASWEC)*. IEEE, 2018, pp. 111–120.

[19] M. G. Jaatun, K. Bernsmed, D. S. Cruzes, and I. A. Tøndel, "Threat modeling in agile software development," in *Exploring Security in Software Architecture and Design*. IGI Global, 2019, pp. 1–14.

[20] T. Theunissen, S. Hoppenbrouwers, S. Overbeek, J. Filipe, M. Smialek, A. Brodsky, S. Hammoudi *et al.*, "In continuous software development, tools are the message for documentation," *ICEIS*, vol. 2, pp. 153–164, 2021.

[21] T. Theunissen, U. van Heesch, and P. Avgeriou, "A mapping study on documentation in continuous software development," *Information and software technology*, vol. 142, p. 106733, 2022.

[22] P. Haindl and R. Plösch, "Towards continuous quality: measuring and evaluating feature-dependent non-functional requirements in devops," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2019, pp. 91–94.

[23] L. Sion, D. Van Landuyt, K. Yskout, S. Verreydt, and W. Joosen, "Automated threat analysis and management in a continuous integration pipeline," in *2021 IEEE Secure Development Conference (SecDev)*, 2021, pp. 30–37.

[24] ——, "CTAM: a tool for continuous threat analysis and management," in *CyberSecurity in a DevOps Environment: From Requirements to Monitoring*. Springer, 2023, pp. 195–223.

[25] N. Shevchenko, T. A. Chick, P. O'Riordan, T. P. Scanlon, and C. Woody, "Threat modeling: a summary of available methods," Tech. Rep., 2018.

[26] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, "Threat modeling tools: A taxonomy," *IEEE Security & Privacy*, vol. 20, no. 4, pp. 29–39, 2021.

[27] W. Xiong and R. Lagerström, "Threat modeling–a systematic literature review," *Computers & security*, vol. 84, pp. 53–69, 2019.

[28] S. Türpe, "The trouble with security requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 122–133.

[29] T. Okubo, N. Yoshioka, and H. Kaiya, "Security driven requirements refinement and exploration of architecture with multiple nfr points of view," in *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*. IEEE, 2014, pp. 201–205.

[30] K. Tuma, R. Scandariato, M. Widman, and C. Sandberg, "Towards security threats that matter," in *Computer Security: ESORICS 2017 International Workshops, CyberICPS 2017 and SECPRE 2017, Oslo,*

Norway, September 14-15, 2017, Revised Selected Papers 3. Springer, 2018, pp. 47–62.
[31] D. Van Landuyt and W. Joosen, "A descriptive study of assumptions in STRIDE security threat modeling," *Software and Systems Modeling*, pp. 1–18, 2021.
[32] Microsoft, "Microsoft Threat Modeling," https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling, Feb. 2024.
[33] The OWASP Foundation, "OWASP Threat Dragon," https://www.threatdragon.com/, Feb. 2024.
[34] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "A novel security-by-design methodology: Modeling and assessing security by slas with a quantitative approach," *Journal of Systems and Software*, vol. 163, p. 110537, 2020.
[35] CAIRIS, "CAIRIS: An open source platform for building security and usability into your software: Quick Start," https://cairis.readthedocs.io/en/latest/gettingstarted.html, Feb. 2024.
[36] TMS, "Threats manager studio," https://threatsmanager.com/, Feb. 2024.
[37] Threatspec, "Example report applied to a WebApp," https://github.com/threatspec/threatspec_example_report, Feb. 2024.
[38] I. Tarandach, "A pythonic framework for threat modeling," https://github.com/izar/pytm, Oct. 2023.
[39] Threagile, "Agile threat modeling toolkit," https://github.com/Threagile/threagile, May 2024.
[40] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.
[41] K. Wuyts, D. Van Landuyt, A. Hovsepyan, and W. Joosen, "Effective and efficient privacy threat modeling through domain refinements," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1175–1178.
[42] L. Sion, D. Van Landuyt, and W. Joosen, "Leveraging the domain experts: specializing privacy threat knowledge," in *The 8th International Workshop on SECurity and Privacy Requirements Engineering (SECPRE)*, 2024.
[43] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2154–2156.
[44] P. Saitta, B. Larcom, and M. Eddington, "Trike v. 1 methodology document [draft]," *URL: http://dymaxion.org/trike/Trike_v1_Methodology_Documentdraft.pdf*, 2005.
[45] M. S. Lund, B. Solhaug, and K. Stølen, *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.
[46] K. Shortridge, "Deciduous: A security decision tree generator," https://kellyshortridge.com/blog/posts/deciduous-attack-tree-app/, Jul. 2021.
[47] A. Schaad and M. Borozdin, "TAM2: Automated threat analysis," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 1103–1108.
[48] J. Von Der Assen, M. F. Franco, C. Killer, E. J. Scheid, and B. Stiller, "CoReTM: An Approach Enabling Cross-Functional Collaborative Threat Modeling," in *IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 189–196.
[49] 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 (SECRYPT)*, vol. 2, 2019, pp. 417–424.
[50] M. Rusakovich, "TicTaaC: Threat modeling-as-a-Code (TaaC)," https://github.com/rusakovichma/TicTaaC, May 2024.
[51] S. Walraven, D. Van Landuyt, F. Gey, and W. Joosen, "Service line engineering in practice: Developing an integrated document processing saas application," *CW Reports CW652, Department of Computer Science, KU Leuven*, 2014.
[52] L. Sion, D. Van Landuyt, and W. Joosen, "An overview of runtime data protection enforcement approaches," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2021, pp. 351–358.
[53] F. Gey, S. Walraven, D. Van Landuyt, and W. Joosen, "Building a customizable business-process-as-a-service application with current state-of-practice," in *Software Composition: 12th International Conference, SC 2013, Budapest, Hungary, June 19, 2013. Proceedings 12*. Springer, 2013, pp. 113–127.
[54] M. Decat, J. Bogaerts, B. Lagaisse, and W. Joosen, "The e-document case study: functional analysis and access control requirements," *CW Reports CW654, Department of Computer Science, KU Leuven*, 2014.