FlowStrider: Low-friction Continuous Threat Modeling

1st Bernd Gruner
Institute of Data Science
German Aerospace Center
Jena, Germany
bernd.gruner@dlr.de

2nd Noah Erthel
Institute of Data Science
German Aerospace Center
Jena, Germany
noah.erthel@dlr.de

3rd Clemens-Alexander Brust
Institute of Data Science
German Aerospace Center
Jena, Germany
clemens-alexander.brust@dlr.de

***Abstract*—Architectural threat modeling is a crucial technique for identifying and mitigating security threats in software systems, helping to prevent costly design flaws. While existing tools aim to reduce its resource-intensive nature through automation, they often lack key features—such as scriptability and integration capabilities—needed for practical use in development workflows.**

In this paper, we present FlowStrider, a tool that addresses these shortcomings by implementing a new, practice-oriented workflow and enabling CI/CD integration through scriptability. FlowStrider reduces the required manual effort, enhances the quality of analysis results, and eases integration into software development workflows, thereby lowering the adoption barrier for continuous threat modeling.

Screencast: https://youtu.be/iRpeU1nubHw

Repository:https://gitlab.com/dlr-dw/automated-threat-modeling/flowstrider

***Index Terms*—tool, threat elicitation, automation**

I. Introduction

Architectural threat modeling is a security-by-design approach that enables the early identification and mitigation of threats in software systems. It is a relevant topic in industry [1] and within the open-source community [2]. However, several challenges remain, including the time-intensive nature of the process, a lack of security expertise, and the absence of clear workflow descriptions [2]–[4].

Research in the field of threat modeling has primarily focused on the automation of various steps to address these challenges. However, automation alone does not fully solve the problems encountered when integrating threat modeling into the software development process [5]. Van Landuyt et al. show that while current tools implement automation to some extent, they often lack key features that support practical usability, such as scriptability for CI/CD integration and the ability to easily adapt and extend the tools to other domains.

Our tool, FlowStrider, goes a step further. It not only automates key parts of the threat modeling process, such as threat elicitation, but is also easily extendable and fully scriptable, enabling seamless integration into CI/CD pipelines. It also implements a workflow derived from practical experience that supports the iterative refinement of threat modeling inputs and provides capabilities for ongoing threat management. Furthermore, FlowStrider includes a set of example threats based on standards published by the German Federal Office for Information Security (BSI)[1], demonstrating how the tool can be used to check compliance with industry standards and legal requirements.

Field	Description	Applicable To	Allowed Values
auth_factors	If authentication is required, which factors will be needed for authentication. Examples: ['PIN', 'Chip Card', 'OTP'] or ['Digital Certificate', 'Biometric Data'].	Node: DataStore, Node: Process	PIN, OTP, Biometric Data, Digital Certificate, Chip Card, Security Token
auth_protocol	Which authentication protocol will be used to ensure integrity. Examples: 'DH_CHAP' or 'FCPAP'	Node: DataStore	DH_CHAP, FCAP, FCPAP

Fig. 1. Example of a Data Flow Graph: visualization (left), corresponding JSON snippet (middle), and excerpt of possible contextual information (right).

By focusing on the following key contributions:

- practice-oriented workflow,
- automation,
- seamless CI/CD integration,
- extensibility, and
- BSI standards threat catalog

FlowStrider sets itself apart from existing tools by positioning itself as an easily integrable solution for modern software development processes. It lowers the barrier to adopting threat modeling and enables continuous threat modeling throughout the software lifecycle.

The tool is programming-language agnostic, open source, available on GitLab[2], and comes with comprehensive documentation[3] and example use cases.

II. Threat Modeling Workflow

FlowStrider implements a custom workflow inspired by practical experience. The following sections provide a detailed description of each step in the workflow. A simple example is used to illustrate the process, showing the data flow graph in Figure 1 and how it is processed by FlowStrider in Figure 2.

A. Data Flow Modeling

The first step of the process involves manually creating an abstract model of the software system under test, resulting in a data flow graph. Since the security-relevant properties of the

[1]https://www.bsi.bund.de/dok/it-grundschutz-en
[2]https://gitlab.com/dlr-dw/automated-threat-modeling/flowstrider
[3]https://flowstrider-defe6e.gitlab.io/

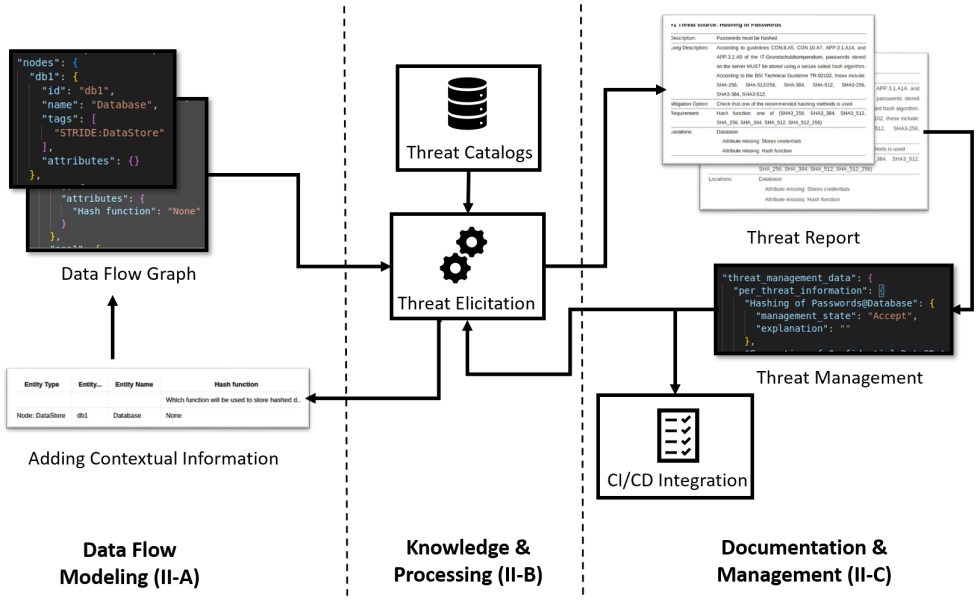

Fig. 2. The FlowStrider workflow consists of three main process steps: (II-A) modeling and refinement of the data flow, (II-B) identification of threats based on the catalog, and (II-C) evaluation of the analysis results. All steps are explained in detail in Section II.

graph elements play a crucial role in the analysis, we introduce an additional feature that enables the efficient inclusion of contextual information. This enhancement refines the graph and consequently improves the quality of the analysis results. The components of this process step are explained in more detail below.

1) Data Flow Graph: FlowStrider takes as input a software system represented by a data flow graph in JSON format and is therefore independent of the programming language of the system under test. An example of a simple data flow graph is shown in Figure 1. A data flow graph consists of five fundamental elements: processes, data stores, external entities, data flows, and trust boundaries [6]. Each of these elements can be annotated with attributes such as encryption methods, hash functions, and other security-related properties that capture relevant security characteristics. This detailed representation aims to describe the system as accurately as possible to ensure the quality of the analysis results. A complete list of supported attributes is available in the tool's documentation (see Figure 1 for an excerpt). The attribute set can also be easily extended to support additional domains and methodologies.

2) Adding Contextual Information: Manually specifying attributes for each element in a data flow graph is a time-consuming task, and it is often unclear which attributes are relevant for analysis. Based on our experience, only a few attributes are typically filled in initially, which can negatively impact the quality of the analysis results and lead to a high number of false positives. This issue arises because we perform the analysis pessimistically: if an attribute required to check for a threat is missing, we assume that the threat occurs.

To address this, FlowStrider includes a feature that analyzes the selected threat catalogs to identify which attributes are missing for each element in the data flow graph. It then generates an Excel file in which users can conveniently fill in the missing required values. This file is subsequently read back into FlowStrider, and the data is automatically merged into the original data flow graph JSON file.

This process allows users to iteratively and efficiently add important attributes, improving both usability and the quality of the analysis results. The Excel format enables easy communication between disciplines in large organizations.

3) Customizability and Extensibility: An important characteristic of FlowStrider is its adaptability and extensibility, which enables its use across a wide range of software project domains and threat modeling methodologies. Particular emphasis is placed on the underlying data model of the data flow graph, which is designed to be easily extendable. To facilitate this, each graph element contains an open attribute Python dictionary that can be populated with relevant contextual information. A detailed overview of the information currently supported by FlowStrider is available in the documentation under the section on the "data model".

In addition to the flexible data model, FlowStrider makes it easy to incorporate new threat catalogs such as legal regulations, industry standards, or project-specific policies. To support this extensibility, threat definitions are decoupled from the program logic. Instructions for adding custom threats are also provided in the documentation.

B. Knowledge & Processing

This process step forms the core of FlowStrider and is responsible for identifying threats using the data flow graph and the provided threat catalog. The two main components of this process are explained in detail below.

1) Threat Catalog: An essential component of the analysis is the threat catalog, which is evaluated against the data flow

graph. Each threat catalog specifies a set of threats, including their descriptions, potential mitigations, and the conditions under which they are applicable.

Threat catalogs to be used for the analysis can be specified via tags in the data flow graph JSON file. Currently, the tool includes two predefined threat catalogs:

STRIDE
: The STRIDE threat catalog [6] is a generic collection of threats that covers the areas of spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privileges.

BSI
: With the BSI threat catalog, we aim to demonstrate that FlowStrider can also be used to evaluate compliance with legal requirements and standards. For this purpose, we selected standards issued by the BSI. Based on these standards, we derived a set of 13 representative threats that can be automatically checked by our tool on the data flow graph. This catalog is more specific than the STRIDE catalog and leverages security-relevant contextual information from the elements of the data flow graph. In summary, the BSI catalog illustrates that FlowStrider can perform automated compliance evaluations for legal and standard-related aspects that can be modeled within a data flow graph.

Creating a new threat catalog or extending an existing one is straightforward and is explained in detail in the documentation.

2) Threat Elicitation: This is a core component of the tool, where the specified threat catalog is evaluated against the given data flow graph. STRIDE-per-Element [6] is used as the core method, evaluating threat conditions for each element in the graph. The analysis follows a pessimistic approach: if required information—such as additional attributes necessary for evaluating a threat—is missing, the threat is considered to occur. The results of the analysis can be output either directly on the command line or as a PDF report (see Section II-C1).

Additionally, this step supports an argument for integration into a CI/CD pipeline (see Section II-C3) as well as the option to include a threat management file in the analysis process (see Section II-C2).

C. Documentation & Management

This step focuses on evaluating the analysis results, which are provided as a PDF report or via the command line. Based on this information, threat management can be performed by determining how each identified threat should be handled.

Furthermore, FlowStrider can be integrated into a CI/CD pipeline, which is important for seamless and continuous integration into the software development process and lowers the barrier to adoption. The CI/CD capability also provides an incentive to keep the data flow model in version control next to the code, reducing the chance of divergence over time. In this context, the analysis results and corresponding threat management decisions are used to assess whether the configured security policy is being met.

TABLE I
THIS TABLE SHOWS THE NUMBER OF THREATS IDENTIFIED BY FLOWSTRIDER AND THE EXPECTED THREATS FOR EACH THREAT CATALOG.

	FlowStrider	Expected
STRIDE	12	12
BSI	9	9

1) Threat Report: The threat report compiles the analysis results in a structured and readable PDF document, as partially shown in Figure 2. It includes a visualization of the data flow graph, an overview of the identified threats, and a detailed breakdown of each detected threat:

- Name
- Description / Long Description (References)
- Mitigation options
- Requirements
- Locations (affected elements)

2) Threat Management: In addition to identifying threats, managing and documenting them are crucial parts of the overall process. This includes assigning a management status to each threat—such as accepted, delegated, mitigated, ignored, or undecided. For this purpose, a separate JSON file is used, which contains all identified threats along with their respective statuses and optional comment fields for documentation. This file can be updated during subsequent executions of the threat elicitation step and the decisions are incorporated into the generated report.

3) CI/CD Integration: FlowStrider is fully scriptable, enabling straightforward integration into CI/CD pipelines. Using the *fail-on-threat* argument, users can define the threat management status at which FlowStrider should raise an error and cause the pipeline to fail. This behavior is based on the analysis results, and the corresponding threat management decisions. For example, the pipeline can be configured to block a release if any threat remains unresolved or unaddressed.

III. PRELIMINARY EVALUATION

We test FlowStrider using unit tests to ensure correct functionality. Each threat is addressed by a dedicated unit test to confirm its correct behavior. This results in an overall high code coverage ($\sim 90\%$) across the program. Beyond potential implementation errors, mistakes can also occur during the interpretation of the BSI standards. To mitigate this risk, all rules were independently reviewed by another researcher.

In addition to unit testing, we perform a smoke test which executes the tool on a small example and compares the output with the expected result. The example is tested with both of our threat catalogs and includes all five data flow graph elements, along with a management file containing various decisions. In this case, all ground truth threats were correctly identified by the tool, as shown in Table I.

Furthermore, we use FlowStrider in our real-world threat modeling projects at the German Aerospace Center (DLR). While the specific outputs are confidential and cannot be

published, these projects span the domains of healthcare, public administration, and embedded systems, demonstrating the broad applicability of the tool. All results produced by the tool in these projects were manually reviewed. Any issues identified during this process were subsequently corrected and re-tested. A comprehensive evaluation of the tool would require a standardized benchmark, which currently does not exist in this field and is difficult to create due to the limited availability of open-source projects with corresponding, high-quality threat models [7].

IV. Threats to Validity

Implementation errors in the threat catalogs or the tool itself may lead to false positives or missed threats (false negatives). To identify and mitigate such errors, we conduct careful manual reviews and maintain unit tests with high code coverage ($\sim 90\%$) as well as a smoke test.

Based on a preliminary evaluation using a simple example, no conclusions can yet be drawn about the generalizability of the approach to other domains. However, through internal use of the tool in three projects, we have confirmed its applicability and correctness in the domains of healthcare, public administration, and embedded systems. Furthermore, the tool's flexible data model and configurable threat catalog make it easily adaptable to additional domains.

Meaningful threat modeling depends on an up-to-date data flow graph. Ensuring this is beyond the tool's scope, as it is a separate research challenge. It can be addressed through automated reconstruction or by validating the graph against the source code.

Not all possible threats are implemented in FlowStrider. Given the large number of potential threats and limited resources, curating a complete list is not feasible. However, our tool includes two built-in catalogs covering basic threats and supports custom threat catalogs.

V. Related Work

There are various tools available to support the threat modeling process. A recent study by Van Landuyt et al. [5] examined both commercial and open-source threat modeling tools. The study found that most existing tools place a strong focus on automation, while aspects such as CI/CD integration and scriptability have received comparatively little attention, despite being crucial for seamless integration into the software development process.

Tools such as Microsoft Threat Modeling Tool [8], OWASP PyTM [9], CAIRIS [10], Threats Manager Studio [11], SPARTA [12], Threagile [13], and TicTaaC [14] offer strong support for customizing threat catalogs and countermeasures. Most also provide automation for specific steps in the threat modeling process. However, scripting capabilities for tasks such as automated threat elicitation or integration into CI/CD pipelines are frequently lacking. For example, Microsoft Threat Modeling Tool, CAIRIS, and Threats Manager Studio do not support scripting for automation, a limitation that our tool addresses.

In addition, most of the mentioned tools do not support scripted import and export of models, threats, and mitigation measures. An exception to this is TicTaaC, which, like our tool, supports these features. However, our tool differentiates itself from TicTaaC by adopting an iterative workflow that is more closely aligned with real-world threat modeling practices and by incorporating the BSI threat catalog.

VI. Conclusion and Future Work

In this paper, we presented FlowStrider, a threat modeling tool that addresses key challenges—such as limited resources and poor integration into existing software development workflows—by automating critical steps and aligning with practical development workflows. By guiding users in efficiently adding contextual information, FlowStrider improves the quality of analysis results. Its seamless integration into CI/CD pipelines lowers the barrier to adoption and enables continuous security evaluation within the software development processes. Additionally, its extensibility allows for adaptation to software projects across various domains.

Future work includes evaluating the tool on additional projects, extending the threat catalogs, and implementing a method for prioritizing identified threats.

References

[1] K. Hermann, S. Peldszus, J.-P. Steghöfer, and T. Berger, "An exploratory study on the engineering of security features," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 721–721. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICSE55347.2025.00184

[2] H. Kaur, C. Powers, R. E. Thompson III, S. Fahl, and D. Votipka, "'threat modeling is very formal, it's very technical, and also very hard to do correctly': Investigating threat modeling practices in open-source software projects," in *34th USENIX Security Symposium (USENIX Security'25)*, 2025.

[3] K. Bernsmed and M. G. Jaatun, "Threat modelling and agile software development: Identified practice in four norwegian organisations," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1–8.

[4] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," *Journal of Systems and Software*, vol. 144, pp. 275–294, 2018.

[5] D. V. Landuyt, L. Sion, W. Philips, and W. Joosen, "From automation to ci/cd: a comparative evaluation of threat modeling tools," in *2024 IEEE Secure Development Conference (SecDev)*, 2024, pp. 35–45.

[6] A. Shostack, *Threat Modeling: Designing for Security*. John wiley & sons, 2014.

[7] 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," in *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering-Companion (SANER-C)*. IEEE, 2024, pp. 141–145.

[8] Microsoft, "Microsoft threat modeling tool," https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling, Mar. 2025.

[9] I. Tarandach, "A pythonic framework for threat modeling," https://github.com/izar/pytm, Mar. 2025.

[10] CAIRIS, "Cairis: Quick start," https://cairis.readthedocs.io/en/latest/gettingstarted.html, Feb. 2025.

[11] TMS, "Threats manager studio," https://threatsmanager.com/, Feb. 2025.

[12] L. Sion, "Automated threat analysis for security and privacy," PhD thesis, KU Leuven, Leuven, Belgium, 2020.

[13] Threagile, "Agile threat modeling toolkit," https://github.com/Threagile/threagile, May 2025.

[14] M. Rusakovich, "Threat modeling-as-a-code in a tick (tictaac)," https://github.com/rusakovichma/TicTaaC, May 2025.