Summary

This thesis explores the applicability and reliability of AI-driven threat modelling tools for software development projects, focusing on their potential to support "shift-left" security practices. Traditional threat modelling remains a vital, but highly manual and expertise-dependent process, which makes it inconsistent, and difficult to scale. This thesis investigates whether AI, particularly large language models (LLMs), can effectively support and automate aspects of this process, and if so, to what extent their output is valid, and sound.

For the evaluation, the thesis introduced a structured assessment framework based on the Goal-Question-Metric framework. Two goals are defined:

G1: Inspect the features of AI-driven threat modelling tools to be used by a software project team

G2: Verify the soundness of a generated threat model from an AI-driven threat modelling tool to establish the value it brings to a software project

In G1, a structured review was conducted on six AI-driven tools: PILLAR, STRIDEgpt, ThreatCanvas, IriusRisk, Aribot, and ThreatModeler. These tools were chosen based on defined inclusion criteria (e.g., use of LLMs, documented AI capabilities, and relevance to core threat modelling phases). Each tool was evaluated across a set of functional, AI-related and general metrics defined for goal G1. The review found that most tools can support system modelling and basic threat elicitation, though only a few offer full AI integration across all modelling phases. Many of the tools rely on rule-based engines for threat elicitation, mitigation and risk scoring, with AI primarily being used for visualisation of the system. The review underscores the current fragmentation and lack of true AI-driven threat modelling tools in this domain, despite their potential to assist novice users and scale modelling efforts.

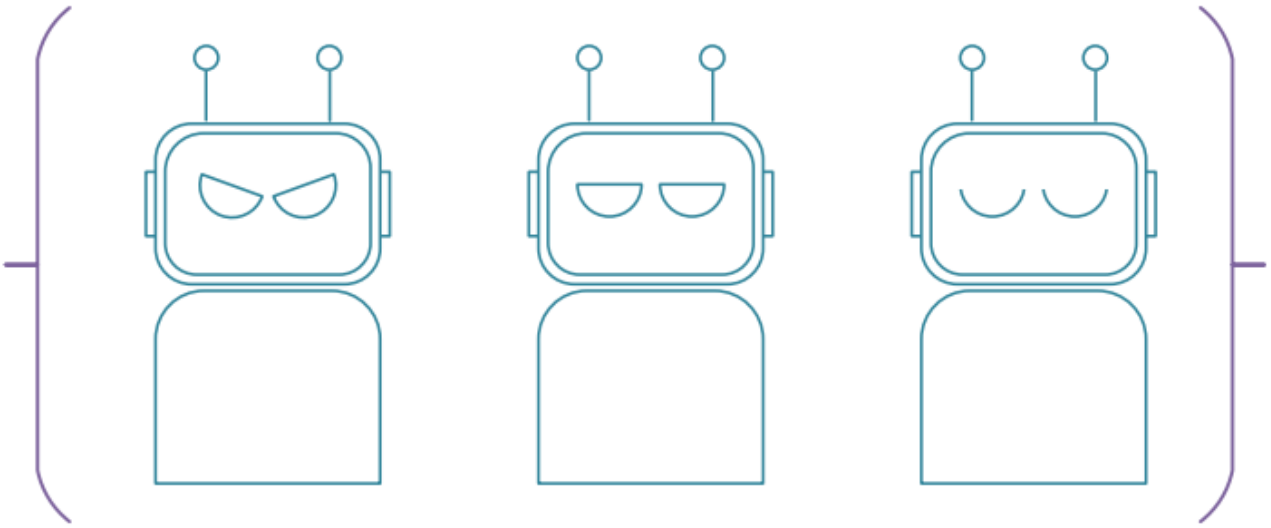
The research includes a practical case study for G2 involving the application of three selected tools: IriusRisk, ThreatCanvas, and STRIDEgpt on predefined systems. The tools' models are compared against manually developed example threat models using a set of defined metrics for G2. The case study seeks to verify the validity and soundness of the tools' threat models. The metrics include evaluating the tools' susceptibility to producing hallucinated assets in the system, and threats in the model.

Key findings from the case study reveal that the AI-driven tools are somewhat able to discover the same threats as the examples, and generally the tools' threat models provide comparable results as the example models. Some of the tools did hallucinate assets which were not found in the actual system, showing that they still have some way to go.

The thesis concludes that AI-driven threat modelling tools are not yet suitable as standalone solutions, but they hold value as assistive technologies that can streamline threat modelling. Furthermore, the thesis contributes a set of practical metrics, a critical review of existing tools, and evidence from applied evaluations that can guide future tool development and adoption in the cybersecurity and software engineering communities.

Applicability and Validity of AI-Driven Threat Modelling Tools

From Method to Measurement



Josephine Marie Bakka
Distributed Systems
Computer Science

Aalborg University, Denmark
Katholieke Universiteit Leuven, Belgium

May 30, 2025

Computer Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Applicability and Validity of AI-Driven Threat Modelling Tools -
From Method to Measurement

Theme:
Computer Science - Distributed Systems

Project Period:
Spring Semester 2025

Project Group:
cs-25-ds-10-06

Participant(s):
Josephine Marie Bakka

Supervisor(s):
Andreas Kjeldgaard Brandhøj
Tobias Worm Bøgedal
René Rydhof Hansen

Page Numbers: 78

Date of Completion:
May 30, 2025

Abstract:

This thesis investigates AI-driven threat modelling tools and their ability to assist and automate threat modelling. Through developed metrics and case studies, it evaluates a set of tool's features and output validity. Findings show that while these tools enhance efficiency, they require human oversight. The work proposes practical evaluation metrics and supports a hybrid human-AI approach to secure software development.

Contents

Preface

This thesis was completed as part of the Master's programme in Computer Science (IT) at Aalborg University, and a semester abroad following the Advanced Master's in Cyber Security at KU Leuven. The research was conducted with support from the CLAAUDIA AI Lab at Aalborg University, which provided access to compute resources and infrastructure necessary for experimentation with AI-driven threat modelling tools.

Additionally, ChatGPT was used to assist with minor code generation tasks, such as formatting scripts and generating boilerplate code for the experiment setup. All critical analysis, design decisions, and conclusions presented in this thesis are my own.

1 Introduction

The cyber security landscape has been predominantly right-shifted with companies mainly focusing on handling incidents and closing gaps in their deployed applications. This means delayed penetration testing, heavier incident response efforts, and generally putting out fires, rather than stopping them by limiting the attack surface from the start, thereby shifting security left [30, 49].

Even though, a company can never fully rely on "left" measures, combining the two approaches have proved to be beneficial, and often less expensive. The "left" upfront cost might be more, however incident handling quickly exceeds the cost in both man hours and loss in revenue [28]. This was witnessed with the CrowdStrike incident (Summer 2024) [9], in which their EDR (Endpoint Detection and Response) tool crashed millions of Windows systems all over the world, causing disruptions in critical sectors like healthcare, finance and transportation. The malfunction could only be fixed through manual intervention, and it took several weeks before all systems were back to normal. CrowdStrike is a leading cyber security company, and this incident proved that even the most trusted companies can make mistakes in their software.

The incident is an example of a costly mistake that could have been avoided had more efforts been put into the design and implementation phases, and it also demonstrates the importance of thoroughly testing your software to verify that they meet the requirements. Whilst this particular incident has more to do with development best practices, it still points at the underlying issue of focusing more on "right" measures to save the day, and how if the "left" ones are not in place, the "right" measures can fail.

Furthermore, the human-driven "right" measures are falling behind, with new exploitable zero-days discovered daily and adversaries getting more creative, it is hard for a human to keep up. As a solution, researchers have started investigating the possibility of replacing humans with AI, because of its ability to absorb vast amounts of data and discover patterns much faster than human capable. So far, AI has proven valuable for early discovery and reaction and it is paving the way for adaptive and self-learning cyber security systems. This is furthered by the addition of reinforcement, and the recent realisation of zero-shot learning[1] (currently seen in GPT and BERT models) making it possible to discover threats without prior data about an attack [42, 5].

Zero-shot learning for automated threat detection was tested by Srivastava[1] and Sanghavi [42], and whilst it seemed promising, it lacked reliability and suffered from a high false-positive rate. They attribute to poor calibration, training data quality and variety, which can also lead to an unintended bias in the model. Given the added manual labour for human detectors, having to sift through the many reported threats, they conclude that the technology shows promise, but for now it is too immature to find applicability in the detection domain.

At the same time, AI usage raises concerns about transparency and accountability, because of its natural black-box behaviour, and vulnerabilities in the AI, itself [5]. Therefore relying solely on AI should not be the main practice, but instead a collaboration between humans and AIs would be best, with the AI taking over the simpler mundane tasks, leaving the humans free to focus on the strategic element [56].

For left shifted security, we have seen very little research investigating the use of AI to create threat models, establish security requirements, and assist in maintaining a secure software development life-cycle (SSDLC). Focusing on mundane tasks, an AI could elicit threats and suggest mitigations for a threat model leaving the human experts to prioritise and develop requirements from what the AI has discovered. In threat modelling, rule-based automation tools already exist, and several articles have reviewed them:[53, 49, 14, 41], showing that these tools speed up a cumbersome and knowledge

[1] A training method that enables a model to make predictions about data that was not present in the training data [5].

heavy task, which is also heavily influenced by the team performing the threat modelling. Based on this, investigating what AI-driven threat modelling tools can offer a threat modelling team could be interesting, as AI is a less rigid technology than rule-engines. For example applying zero-shot models would enable the threat modelling team to find threats for a system based on trends in different domains, which they might not have found on their own.

A preliminary study by Obioha-Val et al. [30] on the feasibility of using AI and OSINT[2] to predict and create threat models. They used the Common Crawl Dataset with a logistic regression model to classify threats, and their model achieved an accuracy of 94%, proving its ability to correctly distinguish between threats and non-threats. Whilst their findings prove the efficacy of AI for predictive threat modelling, they still found several risks, which can obfuscate the results. They found poor data quality to be a key factor, but the biggest impact is adversarial manipulation rendering the model completely useless. They urge that any further work in this field should address these risks before the AI model can be truly useful.

AI seems to immature to be a valuable technology in cyber security, but restricting its application to mundane tasks, like furthering current automation tools in threat modelling, could be within reason. Therefore, the project will focus on the "left" method: threat modelling, seeking to uncover the use of AI and to what extend the methods overcome the aforementioned obstacles. Limiting ourselves to investigation the state of AI-driven threat modelling tools, and measuring their performance through a case study. The main contributions of this project are:

- A set of metrics to assess AI-driven tools and their generated threat model
- A high-level review of current AI-driven threat modelling tools
- An applicative study of the validity of AI created threat models

The rest of the project is organised as follows: Section 2 discusses related threat modelling literature, followed by Section 3 with an overview of the threat modelling domain. Sections 4 and 5 describe the current state of AI models and usage. Section 6 details the methodology used for selecting and studying the tools. Then in Section 7, the selection and review of the tools is carried out, leading to the applicative case study in Section 8. The project will wrap up with a discussion and conclusion in Sections 9 and 10.

2 Related Work

We focus on AI-driven threat modelling tools, meaning that we need to understand the current scope of three key areas: (i) The threat modelling landscape, (ii) Automated threat modelling tools and (iii) AI-driven threat modelling tools. Thereby, we can reason about the AI tools role in the broader domain, how others have measured similar threat modelling tools, and highlight the absence of a comprehensive AI-focused review of threat modelling tools.

2.1 Threat Modelling

The first comparative threat modelling literature review was performed by Xiong and Lagerström [53], in which they sought to uncover directions for further research. They analysed 54 threat modelling articles and divided them into three clusters, applicative studies, methods and process descriptions, which are then further divided into sub clusters. Their review shows that the majority of articles is

[2]Open-Source Intelligence dataset created by gathering public information from various sources [30].

still focused on manual threat modelling, but newer research is trending towards automation. Five future research directions are proposed: Automation, Validation, Tools for threat modelling, Combining defence with threat modelling, and Expansion of attack categories, of which automation and tools are aspects that this project will investigate, however the scope is narrowed to AI automation.

A later review from 2023 by Granata and Rak [14] confirm the expected future directions of Xiong and Lagerström, seeing a growth in automated methods, classification techniques, validation approaches and tools. Their review showed that most researchers use data flow diagrams (dfd), STRIDE threat categories and label-based assessments for automated threat modelling. Granata and Rak also performed a review of three open-source automation tools (Microsoft TMT, OWASP Threat Dragon, SLA-generator), in which OWASP's Threat Dragon missed some of the threats found by the other tools, and the SLA-generator was better at comprehending distributed systems with databases, virtual machines and networks compared to Microsoft TMT. Their review did not seek to prove, which tool is better, but to investigate the properties of the tools, and this project shall do the same, seeking to investigate AI-driven tools, not rank them based on performance.

2.2 Threat Modelling Tools

Threat modelling tools must exhibit certain qualities in order to be valuable for an organisation, the same is true for AI-driven tools, and in order to evaluate their capabilities some criteria need to be developed.

Shi et al. [41] established a taxonomy for categorisation of threat modelling tools, determining the seminal features of the tools. They focus on creating objective metrics, which are qualitatively evaluated, emphasising that they do not seek to find the best tool, but to create metrics that can assist organisations in choosing a suitable tool, based on their experience and use case. The taxonomy includes functional metrics, like input type, threat library, threat evaluation, and mitigation suggestions, along with contextual metrics, like software-development lifecycle integration, open/closed source and maturity level of the tool. They test their taxonomy on six different threat modelling tools: Microsoft TMT, OWASP Threat Dragon, OVVL, OWASP pytm, Threagile and IriusRisk, concluding that open-source tools seem to fall short compared to commercial tools, with smaller threat libraries, less options for reusability of previously modelled components, and all except one do not give any mitigation suggestions. As for the taxonomy, they do not directly evaluate the usefulness of the different metrics, but state that it will be able to assist researchers and organisations in selecting the correct tool. The taxonomy developed could be of interest to this project, as it could form a basis for the metrics to evaluate on, however the metrics do not address automation, AI integration, or tool effectiveness, which will be needed in this project.

Other studies by Tan and Garg [44] and Van Landuyt et al. [49] used similar metrics, where Tan and Garg added a security/privacy metric and Van Landuyt et al. added automation degree and Continuous Integration Continuous Delivery integration metrics. These metrics could further improve the taxonomy from Shit et al. to discover the automation aspects of AI-driven threat modelling tools.

All of the studies apply their criteria differently: Shi et al. provided a simple table of the metrics' value, Tan and Garg used a Pugh matrix and Van Landuyt et al. the Goal-Question-Metric (GQM) framework. The table and GQM highlight the differences between the tools, and both methods do not supply any numerical scoring, whereas this would be the main purpose of the Pugh Matrix. Neither of the studies mention any downsides to their chosen method, only that the outcomes are influenced greatly by their chosen criteria. Any of these methods could be applied in this project with our own criteria.

2.3 AI Threat Modelling Tools

In recent years, AI-driven tools have started to emerge with researchers leveraging a threat enhanced Large Language Models (LLM) to discover threats. Some proof of concepts are: ThreatModeler-LLM by Yang et al. [54], Cyber Sentinel by Kaheh et al. [20] and (LLM)-based by Elsharef et al. [12]. Both (LLM)-based and ThreatModeler-LLM use the open-source llama model, which is then adjusted to enhance its contextual capabilities and vulnerability knowledge. (LLM)-based is trained on the National Vulnerability Database[3], and they feed the LLM the system's documentation, so it can understand, which system to threat model. Whereas, the ThreatModeler-LLM is fine-tuned using existing threat models from Microsoft TMT combined with prompt engineering to optimise results. Unlike the others, Cyber Sentinel is a conversational agent, which assists in querying the data found in various cyber threat intelligence feeds.

In all three articles, the researchers find AI can be useful in speeding up the threat modelling process, however, the models all suffer from dataset biases, inadequate understanding of the system and lack of real world knowledge. The issues cause a skew in the predictions and a large amount of false-positives. The researchers also raise the question of preserving the privacy of the application, as the AI is also vulnerable to attacks, which could lead to leaking the discovered threats to attackers. Thereby, concluding that further work is needed before the models will be able to provide reliable results. These articles highlight some issues that the AI-driven threat modelling tools will need to address, and they can be used as part of the evaluation criteria in this project to address the AI part of the threat modelling tools.

So far, only one other study has compared AI-driven threat modelling tools. Sędkowski [43] compared a pre-prompted GPT 4.0, STRIDEgpt and ChatGPT 4.0 to a human's threat model. The pre-prompted GPT 4.0 is given some initial prompts to enhance its contextual understanding prior to the actual threat modelling. The models are then tested on network scanning data extracted from Nmap, where STRIDEgpt and the human is provided with a description of the application, and the pre-prompted GPT 4.0 and ChatGPT 4.0 is given the actual scanning data from Nmap. The models supplied with the pure Nmap data create better threat models, uncovering more of the threats than the human and STRIDEgpt. This way of feeding data is rather unconventional, and threat modelling tools mainly use diagrams or textual descriptions of the system, as output data cannot be obtained before the system is implemented [41]. Compared to the study performed by Sędkowski, this project will apply a more traditional use case, whilst also contributing an investigation into the functionalities of the tools, thereby not only looking at effectiveness of the tools.

In reviewing related articles, a set of possible criteria and methods have been discovered, and it showed that no study has investigated both the functionalities and effectiveness of AI-driven threat modelling tools. Before, we can proceed with creating our criteria, it is necessary to understand three key areas: the threat modelling domain, AIs and how the limitations of AI can impact threat modelling.

3 Threat Modelling

Threat modelling is a structured way of discovering vulnerabilities, before they can become threats. A threat can formally be defined as "*a potential or actual event that can adversely affect IT infrastructures, applications and data, which can have negative consequences for individuals or organisations*" [28], and it can be caused by internal/external, accidental/intentional and computer/human failures. A threat model contains a set of threats addressing the vulnerabilities in a system, and a set of suggested

[3] A repository of software vulnerabilities and configuration settings used by the U.S. government to create an open standard for reliable and interoperable vulnerability impact metrics, and assessment method [10]

corrective measure to fix them. Usually the process starts by defining the system, and then applying one or several threat elicitation methods to the model. The overall threat modelling approach can roughly be divided into four phases [28, 16]:

System Description, document the system's functionalities, assets, and architecture, including its context and technologies. The outcome can be user requirements, an asset list, adversarial descriptions, and/or a data flow diagram (DFD), and the format will depend on the precise threat modelling approach (attacker-, software- or asset-centric).

Threat Elicitation, discover all possible threats, which pose a risk to the system, its owners, and assets. The analysis can rely on mnemonics of threat categories, and catalogues to facilitate the brainstorming process. The results will be an overview of possible threats, which components in the system they affect, and how the component can be affected.

Risk/Threat Assessment, discuss the risk of the threats by analysing the likelihood of realisation and the severity of the repercussions. A risk can impact anything from the system's normal behaviour to an organisation's performance, reputation and finances. Calculation of risk can be based on quantitative or qualitative measures, with most risk assessments relying more or less only on qualitative measures.

Mitigation Suggestion, find fixes for the most severe threats to the system a.k.a. the ones with the highest risk. Possible mitigation strategies are to avoid, transfer, mitigate, or accept the risk. It is preferred to fully avoid the risk, and if not possible then implement mitigations to relinquish or lower the risk. Acceptance should only be used as a last resort, however accepting residual risks after initial mitigation is common.

Threat modelling is an iterative process, and it should be repeated whenever changes are made to the system to ensure it always reflects the threats and mitigations of the system [49]. The threat modelling team should not only consist of security engineers, but also developers and product architects, who will assist in explaining the different components of the system and its context. The security engineers need to possess the knowledge to translate abstract threats into specific threats for the particular system. This is no easy task, and it requires years of experience to create effective threat models, as they need to keep up with the ever-evolving threat landscape and be able to model very complex systems [12].

3.1 Threat Modelling Approaches

In order to guide the threat modelling team in their discovery of threats, several threat modelling approaches have been created, each presenting a different focus for the analysis. The approach chosen by the team will greatly impact threat model and care must be taken to select one that best suits the team and objective of the treat modelling [28]. The approaches are divided into [16]:

- Attacker-Centric, *defining what to protect against.*
- Software-Centric, *defining the system to protect.*
- Asset-Centric, *defining the elements to protect.*

Each type focuses on different aspects of a system, i.e. the attacker, software or assets of the system. An overview of specific approaches for each type is seen in Table 1. Some of the approaches only focus on threats within either security (STRIDE), or privacy (LINDDUN), whereas other methods are more risk focused, typically inherent for asset-centric approaches. Therefore, the asset-centric are commonly referred to as risk-centric, and they all specify a way of prioritising the threats according to a calculated risk [16].

Type	Type	Focus Area
Attacker Centric	Attack Trees	Attack Patterns
Attacker Centric	VAST	Security Threats & Attack Patterns
Software Centric	STRIDE	Security Threats
Software Centric	LINDDUN	Privacy Threats
Asset Centric	Trike	Risk Assessment
Asset Centric	PASTA	Security Threats & Risk Assessment
Asset Centric	OCTAVE	Security Threats & Risk Assessment

Table 1: Overview of Threat Modelling Methods

3.1.1 Attacker-Centric Approaches

These approaches focus on the potential attacker, defining attacker categories as the basis for discovering threat scenarios. In this approach, the team iterates over the list of attacker categories, trying to understand their mindset, objectives, and available resources, and then defines threat scenarios according to the obtained knowledge [16].

Attack Trees
These are tree-shaped diagrams showing how an attack could be carried out on the system, the root is the goal of the attack and the leaves are ways of achieving it. Each tree represents a different attack goal, and a complete threat model will contain a set of trees. After decomposing the goals, the likelihood is assigned to each of the leaves, to assist in assessing the risk of the attack [40]. Attack trees is one way of visually displaying attacks, and it is possible to use graphs or matrices as well.

This is a very simple and comprehensible approach, and it is possible to use it even for larger systems by decomposing it into smaller components. Thereby, modelling attack trees specific to each component instead of the entire system at once. However, the method does not specify any guidelines for assessing goals, attacks or risks, and it should only be used by organisations with highly expertised cybersecurity personnel. This method is often used in conjunction with other threat modelling methods, like STRIDE and PASTA, to visualise the threats and possible attacks [40].

VAST
The Visual, Agile, and Simple Threat approach is developed for scalability and usability allowing it to be adopted by large organisations. It bridges the gap between development and infrastructure teams, and the result is two threat models focusing on each aspect. The application threat model represents the architectural view, and the operational the attacker's view, however, both are based on an initial DFD [40].

VAST can be integrated into the organisation's development lifecycle, and it was build with automation in mind, thereby making it easier to adopt. It has also shown to be consistent upon repetition, and it can contribute to risk management. Even though, it is easy to integrate, the documentation is vague and poor, which can make it hard for organisations to adopt the approach [40].

3.1.2 Software-Centric Approaches

These approaches look at the system architecture, where the team tries to fully comprehending the system at hand, creating fine grained architectural diagrams/DFDs, and then adding trust barriers. The team will then iterate over the created diagrams, particularly the barriers between the components, and come up with possible threat scenarios [16]. Typically, the system is modelled as a DFD, which

focuses on how data flows between the processes, data stores and external entities, and it defines the trust boundary for the system [41].

Examples of software-centric approaches are STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) and LINDDUN (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, and Non-compliance). Both are mnemonic approaches for discovering threats within the given threat categories, where STRIDE is used for security and LINDDUN for privacy [40].

For STRIDE two variations exists: STRIDE-by-element and STRIDE-by-interaction, they both iterate over the threat categories, but differ in the analysis of the DFD. LINDDUN also has different flavours: Go, PRO and MAESTRO. LINDDUN GO is a card-deck detailing the most common privacy threats, not relying on a DFD, but on the team's own understanding and discussion of the system. Whereas, LINDDUN PRO is a systematic interaction-by-interaction iteration over the components and flows in the DFD, searching for privacy threats for each category. LINDDUN PRO is supported by pre-made threat types, trees and mapping tables. MAESTRO is similar to PRO, but it uses a threat-specific system abstraction instead of a DFD to support a more in-depth analysis for the threat categories [27].

The mnemonic approaches are quite simple, and they can only discover threats for the predefined threat categories, whilst being somewhat labour intensive and time consuming. At the same time, only relying on the DFD can be limiting, as it does not provide any contextual information, which could lead to security related issues. Both STRIDE and LINDDUN are best for smaller systems, and while they have a low false-positive rate, they do give many false-negatives, meaning there is a high likelihood of missing threats [40].

3.1.3 Asset-Centric Approaches

These approaches describe the resources/assets of the system and organisation that the team wishes to protect. A resource is something the attackers want, things you want to protect or something connected to either of the two, examples of these are software, hardware, data, intellectual properties and human actors. In these kinds of approaches, the team must decide, which assets should make the cut, and the list can easily grow exponentially. When the asset list has been created, the team will move on with finding threat scenarios asset-by-asset [16].

Trike
It focuses on risk management and it takes a defence perspective on threat modelling, trying to understand the system and its context, instead of the capabilities of attackers. The team need to explain the impact of threats, reasoning behind mitigations, and why some risks were accepted in the eyes of the organisation [29]. Instead of starting with a DFD, Trike uses an actor-asset-action matrix to investigate what Create, Read, Update, Delete (CRUD) actions should be allowed, disallowed or allowed with restrictions. The matrix will be mapped onto a DFD, and the team will iterate over the diagram to find elevation of privilege and denial of service threats. Finally, actors will be rated on a five-point scale to obtain the risk scores for each threat [40].

For Trike to work, the team needs to have a complete understanding of the entire system, which can be hard to obtain for very large systems. Therefore, it is best used for threat modelling on smaller less comprehensive systems. Trike's scoring system is too vaguely defined, which is not helped by the poor documentation of the method, adding an extra hurdle of applying the approach [29].

OCTAVE
It uses the organisation's own security practices to perform the Operationally Critical Threat, Asset, and Vulnerability Evaluation. The outcome is used to find areas of improvement for the current security practices to make sure that it can address the operational risks. The risks and threats are captured in

attack trees, and they suffer from the same issues as the attack tree approach, with linear expansion of the system's size, making it unclear which paths represent which threats [29].

OCTAVE only addresses organisational risks and not technological risks, therefore another method will be needed to assess these types of threats. This focus also means that OCTAVE is primarily to be used in larger organisations, with other less comprehensive variations, like OCTAVE-S, being more suitable for smaller organisations [40]. At the same time, OCTAVE is a highly complex method, and learning the process can be quite time consuming, however, it is one of the more robust threat modelling methods making learning it worthwhile [29].

PASTA
It is the Process for Attack Simulation and Threat Analysis, bringing business objectives and technical requirements together. It describes seven stages, initially identifying the objectives and business, then analysing the system for threats, weaknesses and attack patterns, finalising the process with a risk assessment and impact analysis [40].

This method should be used by organisations that wish to align threat modelling with their strategic objectives, as business impact analysis is an intrinsic part of the PASTA method. Therefore, it will often require spending time on training and educating the key stakeholders in the organisation, as they will need to participate in the threat modelling process [29].

The presented threat modelling approaches show that it is important to choose one that suits the maturity and size of your organisation and system, where approaches like OCTAVE and VAST are better suited for larger organisations, compared to the others. However, OCTAVE, VAST, and also Trike suffer from vague and poor documentation, which could make organisations think twice about using them. Compared to these, STRIDE and LINDDUN offer simplicity, but they lack risk assessment and mitigation capabilities. For any of the approaches to be successful, they require highly skilled cybersecurity professionals, who have an in depth knowledge of possible attacks, vulnerabilities and mitigations, which is not always possible. To assist the threat modelling team's knowledge, they can adopt one or several threat knowledge bases (libraries and catalogues) containing information about currently discovered vulnerabilities, how to mitigate them, and how an attack can exploit them.

3.2 Threat Knowledge Bases

Previously discovered vulnerabilities and attacks are recorded in several knowledge bases, some focus on mapping the vulnerabilities and possible mitigations, whilst others look at possible attack techniques and tactics. These knowledge bases capture some of the expertise needed by the threat modelling team and it is a reusable source of information for eliciting threats [52].

When looking at threat knowledge bases, we distinguish between does modelling the problem space and the solution space. The problem space is commonly referred to as threat catalogues, and they contain threat and attack explanations for elicitation. Whereas the solution space are threat libraries describing mitigations and countermeasures for threats. Some knowledge bases capture both the problem and solution space, and even provide a risk score for the threats [49].

The STRIDE and LINDDUN methods contain a threat catalogue presented by threat trees for each of the threat categories and DFD element types (processes, data stores, external entities). Whilst MITRE, a U.S. government sponsored non-profit-organisation, has developed [52, 41]:

- CAPEC, *Common Attack Pattern Enumeration and Classification*
- ATT&CK, *Adversarial Tactics, Techniques and Common Knowledge*
- CVE, *Common Vulnerabilities and Exposure*
- CWE, *Common Weakness Enumeration*

CAPEC and ATT&CK enumerate attack patterns, looking at exploitation of weaknesses, and techniques of attackers in the real world. ATT&CK is limited to network defences, and it focuses on documenting advanced persistent threats in networks, dividing them into the matrices: Enterprise, Mobile and PRE-ATT&CK [41].

On the other hand, CVE and CWE catalogues contain publicly recognised vulnerabilities and weaknesses, providing a description, mitigation options and risk score for each catalogued entity. CVE and CWE, each have their own scoring system, CVSS for CVE and CWSS for CWE, these are described further in Section 3.3 [41].

Another contributor to the threat modelling landscape is OWASP, who regularly publishes several domain specific top 10s of security risks. The top 10s provide best practices, describe threats and propose mitigations, and even though it is not a full-fledged knowledge base, it can used as one [52].

In automated threat modelling tools, these knowledge bases are used as input for the threat elicitation phase, in which rule-based engines can filter threw them and decide which are applicable to the system [49]. As for AI-driven tools, it would be possible to fine-tune the AI model on some specific knowledge bases, for example to enhance its ability to identify network attacks. However, it might be more prevalent to fine-tune the model in other ways, like prompting it about the task at hand.

3.3 Threat/Risk Assessment

An essential step of threat modelling is performing a risk assessment in order to prioritise the threats, thereby allowing the organisation to put efforts towards the riskier threats. A risk assessment looks at the likelihood of realisation and the severity of the threat's impact on the organisation. This comes with its own challenges, first and foremost it is hard to properly quantify risks and usually the team will have to rely on qualitative measures to assess the risk. The qualitative measures makes it hard to reproduce the risk evaluation as different people can evaluate the risks differently [28].

One qualitative approach is DREAD, which calculates the risk of each identified threat according to the Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability. A score is given for each category, and then the collective score is converted to a qualitative risk rating: Critical, High, Medium, Low, if any of the categories have a high risk, the threat will always be evaluated as high. This approach is commonly combined with other mnemonic threat modelling approaches, like STRIDE, to identify the likelihood of the discovered threats. DREAD has shown to suffer from too much subjectivity to create reliable and repeatable results. It is still used in industry, but Microsoft (the creator) discontinued the use of DREAD in their own software development life-cycle in 2010, because of its instability [29].

Compared to DREAD, Factor Analysis of Information Risk (FAIR) presents itself as a quantitative risk analysis, based on Loss Event Frequency (LEF) and Loss Magnitude (LM) within a given time-frame,

typically annual. LEF is a measure of how often loss is likely to happen, and it can be decomposed into threat frequency and amount of vulnerabilities. LEF will evaluate to a distribution over the time-frame i.e. 5-25 incidents for a year. LM determines the tangible expected loss for a threat, and it considers the primary and secondary stakeholder's losses for each threat event [13].

The aforementioned risk calculation methods are done by each individual team, but public scoring systems, like the Common Vulnerability Scoring System (CVSS) and the Common Weakness Scoring System (CWSS), have been created to try and sustain the calculation. The two systems use different metrics to calculate the severity, as an example the CVSS score is calculated based on the characteristics that never change (the base), the characteristics that changes over time (the time) and the characteristics that change with the user environment (the environmental) [16]. These systems also suffer from inconsistent risk scores based on the assessment team, and it is not transparent, how the score is actually calculated based on the metrics. So while these systems can help calculate the risks, they still suffer from the human factored issues as with other methods [40].

The current way of threat modelling poses several issues, as the result is dependent on the approach and the people involved in the process. The approach will guide the threat modelling team towards a specific target, be it security, privacy, or risks, which can cause the results to not properly reflect the threat landscape for a particular system. At the same time, prioritisation of the threats needs not only consider the system's inner mechanisms, but also the context of which it exists, like the organisation, broader infrastructure and regulations/compliance. So even if a team is able to discover all threats, the threat model can still suffer if they fail to prioritise them appropriately. Furthermore, once the initial threat model is created, it will need to be maintained as the system and the organisation grows. Threat modelling an enterprise level system can easily become a cumbersome task and many methods are not scalable enough to comprehend these types of systems, leaving extra overhead for the maintainers of the threat model [28].

However, the biggest factor impacting all phases of threat modelling is the team, their knowledge, subjectivity and understanding will influence the end result, and impact the reproducibility of the model. Some of these impacts have already been mitigated with the use of automated tools, along with improvement of scalability and less time spend, but these tools still rely heavily on human interaction, thereby inheriting the subjectivity of the team [12, 54].

For a more autonomous approach AI-driven threat modelling tools could be a solution to remove human subjectivity and bias from a threat model, however the AI might have its own bias, and therefore the treat model will depend on the exact tool that was used. The extend of the bias could be less, but other limitations of AI could interfere with the results as well.

4 AI Models

In 2022, we saw the release of OpenAI's ChatGPT, a general purpose AI assistant, followed by GitHub Copilot specifically targeting developers. These models have been extensively trained to emulate human conversation, effectively identifying contextual cues, and providing fitting responses[24, 43].

In 2024, the Stack Overflow Developer Survey[31] showed that 76% of software professionals use or plan to use AI tools, and 62% are currently using them. This is an increase compared to the 44% in the survey from 2023. Diving deeper into the exact use cases 81% use it for documenting code, 80% for testing code, and 76% to write code. Most of the participants agreed that AI tools will become a more integral part of their jobs to improve productivity and efficiency. The survey attests to the software development industry's adoption of AI in their every day tasks, and it is expected to increase as AI models improve.

In cyber security, AI assistants/models can replace more expensive static analysis security scanners, create threat models, and generate security requirements. In threat modelling, AI assistants lessen the manual load by defining an initial threat model, and they can provide suggestions for possible attacks on the system, acting as a sparring partner rather than the main contributor. Thereby, AI assistants can diminish the need for highly experienced security expert, as it can help identify threats that the more novice security worker might otherwise have overlooked. However, the quality of threat models created by general purpose AI assistants depend heavily on the user's prompting capabilities, as the AI assistant can easily be confused by inaccurate information or lack of context of the system[24].

This observation applies broadly to general-purpose AIs, not only to threat modelling. They find little use for domain specific tasks, and instead domain optimised AIs would provide better results[24]. An example of this is Cyber Sentinel[20], also mentioned briefly in Section 2.3, it is originally one of OpenAI's GPT-models, which they fine-tuned for security. This allowed it to better answer and assist with security tasks compared with the original GPT-model. The GPT-model is a large language model, and so are the AI models used in other threat enhanced and modelling tools[27, 2, 54].

4.1 Large Language Models

Large Language Models (LLM) act like conversational assistants, helping users with information gathering, problem-solving and explaining concepts. These models undergo extensive training, enabling them to[55]:

- Comprehend natural language context.
- Generate human-like text.
- Be context aware and consider factors, like domain expertise.
- Excel at problem-solving and decision-making.

The models require an extensive training corpora, where pre-training and post-training techniques are applied in order to fine-tune their hundreds of billions of parameters. Training these models is time-consuming and demands great computational power, therefore it is preferred to use a pre-trained model, and then tune it for your specific domain, like threat modelling[55]. An example of this is OpenAI's Codex, a general GPT-model trained on a smaller, domain specific dataset to tune it for programming tasks[57].

A high level description of LLM concepts is given below, these will be used in the subsequent sections to describe current LLM models, challenges and fixes:

- Corpora, *the data used to train the LLMs, the term is used for both pre- and post-training data. The data is typically preprocessed into tokens by cleaning the data points and restructuring it into a machine understandable format[22].*
- Parameters, *the values learned by the model during pre-training, they will be the main contributor in the LLM's decision making process, and their values directly reflect trends in the training data[55].*
- Pre-training, *initial training of the LLM on a massive corpora with trillions of tokens to instantiate its parameters[22].*
- Fine-tuning, *readjusting the parameters for specific tasks (coding, security, ...) and optimisation of the interaction with users. It is also referred to as post-training, because it often entails training the model on a different more specialised corpora[22].*

- Prompting, *the way that a user communicates with the LLM, it involves crafting clear and context-rich instructions that elicit optimal responses. Often, the user will need to prepare pre-prompts to ensure that the LLM has the correct knowledge and mindset before asking the actual question[33].*

4.2 Assistants & Models

It is important to note the difference between the AI models (GPT, Llama, Gemini) and the AI assistants (ChatGPT and GitHub Copilot) as these serve different purposes. The AI models can be implemented, fine-tuned and become part of systems owned by you, whereas AI assistants are complete systems that are ready for use, but owned by the organisation publishing them, and thereby all interaction-related data is owned by them as well[55, 24].

As you do not own an AI assistant extra care should be taken to not share private information with it, as it can be used to retrain the models. Therefore, threat modelling tools should always implement AI models into the tools, and they should declare, if they plan to use the shared information to fine-tune the model.

There are a variety of models to choose from: OpenAI's GPT, Meta's Llama, Google's Gemini, BERT and more, dating back to 2018 with the initial BERT model, followed by GPT-3 in 2020, and now GPT-4 released in late 2023. The models that have gained most traction are Gemini, Llama and GPT. These models excel in science, logical reasoning, interaction, and they are able to learn from their surroundings, thereby advancing their initial capabilities[55].

A brief overview of the models is seen in Table 2, however the Gemini model from Google is replaced by their open-source version called Gemma. It is not possible to gain insights into Gemini or use it outside of Google's own eco-system, and therefore it is not relevant for AI-driven threat modelling tools[34].

	AI Models		
	Llama	Gemma	GPT
First release	2023	2022	2020
Current version	Llama 3	Gemma 3	GPT-4
Provider	Meta	Google	OpenAI
Main Tasks	Reasoning	Searching	Conversation
Open Source	Yes	Yes	No
Parameters	405 B	27 B	1.7 T
Corpora	15 T	14 T	–
Context Window	128 K	128 K	32 K

Table 2: Overview of Llama, Gemma and GPT AI Models [55, 34, 15, 21, 1]

The models excel at different tasks, Gemma can provide precise and informative answers, utilising Google's extensive knowledge base, and it enhances transparency by effectively citing its sources in responses. Gemma focuses on factual correctness over creativity and personality, whereas this is an integral part of GPT, making it a better conversationalist[34]. At the same time, both Gemma and Llama have much larger context windows than GPT, the larger context window will better enable them to retain their contextual awareness doing prolonged conversations. In turn, GPT can start providing conflicting responses, and it will need the user to repeat and rephrase their prompts multiple times to regain its contextual understanding.

4.3 Limitations

Each LLM is specialised in a subset of tasks, and their responses will depend on the training and fine-tuning, thereby adding a level of subjectivity to the responses they provide. Discovering the extend of subjectivity and possible false threats in an AI-produced threat model is impeded by their black-box nature, thereby lessening the value the AI could bring. For each limitation, we will discuss the impact on threat modelling, and the extend of the current solutions to better understand the value that AI models can bring to threat modelling.

4.3.1 Transparency & Interpretability

The model's structural openness and accessibility, and the degree to which you can explain, how it has reached certain conclusions[17].

Cause:
LLMs and other AI models suffer from the inherent black box problem, in which its decision making process lacks the needed opacity to enable the user to understand, how it has reached certain conclusions. This does not affect accuracy, as a model can predict with high accuracy without the need to explain its decision[17].

Effect:
A lack of transparency disallows for external scrutiny of the model and its data sources, and it can be a contributing factor to other issues like bias, fairness and obscurity of responses. Whereas, interpretability inhibits the trust and reliability in the model and its outcome, and without it users need to verify the results further before applying them[17].

In threat modelling, non-interpretable results would limit the usability, as the threat modelling team would need to manually validate the threats before taking action. The team would then spend the same effort or more than if they had not applied the tool at all.

Furthermore, non-explainable models cannot be used for any elements involving the General Data Protection Regulation (GDPR) and similar regulatory guidelines, as these require the "right to explanation" of automated decisions. Therefore, before sharing any sensitive information with an AI, the organisation would need to make sure that they can explain their decision making process from start to finish[17].

Solution:
The use of Explainable AI (XAI) methods can help improve the transparency and interpretability of the AI, by providing insights into the AI's decision making process. Explainability focuses on "why" a particular decision was made, and with these methods the organisation would know exactly, how the conclusion was drawn. The organisation can then make sure that no bias or inaccurate data was used, and they can see, which features contributed to a threat being discovered[17].

In both ChatGPT and Gemini, the transparency is rather limited, but they are able to provide the user with the sources they used to create their responses, which the user can verify manually. However, it does not actually provide a direct insight into the actual decision making process, but assists in verification towards accuracy of the results[34]. These models can also use Chain-of-Thought (CoT) to provide a textual explanation of their thought process, this method will provide arguments for reaching certain conclusions similar to those of a human. However, it cannot provide detail into the parameters that contributed to the result, for this XAI methods are needed. It is possible to apply XAI methods post training, like LIME (Local Interpretable Model-agnostic Explanations) and SHAP

(Shapley Additive Explanations), and developers could add these methods to their AI model when implementing them in their systems[17].

XAI is, like CoT, not a perfect solution, as LLMs have millions or even billions of parameters affecting its decision making, and fully comprehending how each parameter contributed to a conclusion is next to impossible. For both XAI and CoT, the explanations would need to be tailored to the reviewer, as we cannot ensure that all reviewers have the same semantic understanding of textual explanations[17]. There is no single solution that fully solves transparency and interpretability issues, and therefore this is left at the discretion of the organisation using the AI-driven threat modelling tool. The organisation would have to add this as a risk of applying the tool, and then develop a mitigation strategy that suits their use case.

4.3.2 Fairness and Bias

The unequal treatment of individuals and groups stemming from historical and structural imbalances in the training dataset[17].

Cause:
In LLMs, the bias is caused by trends in the training data, and the data will often model the minds of the humans, who created it. It is not possible to train on all available data, because of limited resources, and therefore the LLM creators must select a subset of the full dataset. The selection can suffer from[22]:

- Unbalanced domain distribution.
- Old outdated information.
- Biases of the creators.
- Cultural skews.

Both domain and cultural skews stem from a lack of available data, and it dates back to the content of the original data sources. An example is Wikipedia, which is heavily weighted towards sports, music, and geography, and therefore LLMs trained on this source will have trouble discussing subjects, like literature, economics and history. The same is seen for language, as over 50% of Wikipedia's articles are written in English by English speakers, skewing the LLM towards English speaking cultures[17, 22].

Effect:
Biased datasets can lead the LLM to perpetuate societal biases posing harm on individuals or groups. This can impact critical domains, like hiring, mortgaging and healthcare, where their decisions would be able to impact people's lives. General examples of biases are: derogative language ("whore"), individual/group exclusions ("only two genders"), stereotyping ("immigrants are terrorists") and toxicity ("I hate ...")[22].

In threat modelling, biases can make the LLM perpetuate unfair profiling of certain attackers causing an obscurity in threat prioritisation. Thereby, the objectivity of the LLM will be diminished, and it will suffer the same issues as seen in manual threat modelling. As here, human biases can influence the analysis, and it can lead them to potentially overlook or underestimate certain threats.

Solution:
So far no method has been discovered to fully eliminate biases and sustain fairness in LLMs, the current solution strategies include[22, 12]:

- Data Augmentation, *diversifying the original dataset by randomly changing attributes, like changing gender pronouns to their counterparts, thereby balancing out any skews.*
- Data Filtering, *selecting subsets of the under-represented groups from the original training data to reuse during the fine-tuning of the model. This can enhance the LLM's understanding of the under-represented groups in the original data.*
- Prompt tuning, *modifying the user's prompts to add textual triggers making it possible to freeze the original pre-trained model's parameters to prevent it from skewing later on.*
- Guardrails, *structural safeguards embedded in the model ensuring that it adheres with preset boundaries for what prompts it is allowed to respond to.*

These strategies can be used in combination, and each one of them cover a different aspect, which the previous method missed. Guardrails have been the most prevalent strategy, and models with guardrails are less susceptible to bias from poor data quality, and they are less likely to be skewed during prolonged prompting. Gemma, Llama and GPT all use guardrails to modify the behaviour of their models, however non of them are completely free of bias, and the creators stress that the guardrails are not a complete fix and issues can occur over time[34, 15].

Meta have improved their corpora extraction for Llama 3 compared to their earlier models. They start by extracting high quality text, then apply a heuristic filter to remove missed low quality data, and finish with a model-based classifier to select the best tokens. The model-based classifier will tailor the tokens to the specific model, and therefore it will select different tokens for the Llama 2 versus Llama 3 model[15].

As the solutions are not yet complete, it is best to monitor the models to make sure that they maintain performance over time against fairness indicators. Fairness indicators would be errors across demographics of cultural and language groups, and analysing the feedback loop through interactions with the LLM[22].

4.3.3 Hallucinations

Responses generated by the AI that might be correctly formatted and seemingly coherent, but they lack factual correctness or they diverge from the actual intent of the source[22].

Cause:
Hallucinations are commonly caused by poor data quality, and in general AIs are only as good as the data, they were trained on, and it will happily propagate any observations found in the dataset. The observations can be biases and inaccuracies, which will cause the AI to skew its results, and they might lack essential knowledge to fully comprehend the dataset[22].

Other issues arise from the training method and overconfidence in their own responses, as is witnessed in ChatGPT. The sequential generation strategy applied in LLMs, i.e. reinforcement learning and sampling-based randomness, can cause hallucinations to grow by magnifying early errors[22].

Effect:
Hallucinations affect the accuracy and dependability of the model, and inherently limits the trust of its responses. They can be categorised into[22]:

- Input-Conflicting, *there is a discrepancy between the user's prompt and the AI's response.*
- Context-Conflicting, *the AI generates internally inconsistent responses in prolonged conversations.*
- Fact-Conflicting, *responses are factually incorrect.*

Hallucinations are already inherent in ChatGPT, and it struggles with tasks that involve common sense and logical reasoning that is not covered by its training data. For threat modelling, analytical thinking, decision making, and problem solving is necessary to achieve better results. The team relies significantly on reasoning, which means that lack of awareness and the ability to reason about the relationship between concepts, can cause the AI to generate false information[43].

Solutions:
Careful selection of the model can be the initial step towards less hallucinations, like choosing a Retrieval-Augmented Generation (RAG) LLM. RAGs have proven to perform better than other LLMs, like Generative Pre-training Transformer models (GPT)[12]. RAG models combine information retrieval with text generation, which results in more factually correct responses, and this is the type of model behind Google's Gemma. Therefore, Gemma excels at providing precise and informative answers, and it utilises Google's extensive knowledge base allowing it to perform proficiently on a variety of tasks[34].

The next step is moderation of the pre- and post-training data to ensure that models are only trained on high-quality data, where noise has been filtered out. This strategy is used in OpenAI's GPT-4 model, where they automatically clean the pre-training data based on similarities in a reference corpora[22].

After deploying the model, continuous evaluation and monitoring shall verify that the model does not degrade over time. Methods for this include iterative testing of the responses and using human-driven reinforcement training, where the latter is a more time consuming task. For iterative testing, it is possible to create a reward model that can automatically evaluate the LLM responses, and then alert when the responses start to decay. The evaluation from the reward model can be used to correct the LLMs behaviour, through the application of other algorithms[54, 22].

As for the user, they can pre-emptively lessen the hallucinations by pre-prompting the LLM to make sure that it fully comprehends the context and meaning of the actual prompt before asking the actual question[33]. The user can also use CoT to improve the results of the models, as they become more aware of their decision making process, possibly limiting incorrect answers[55].

For most of these limitations no complete solution exist, but it is a matter of combining current strategies with ongoing monitoring of the model to ensure it does not start skewing its responses. Gemma, GPT and Llama models are currently using guardrails and various data augmentation methods to limit bad behaviour, and for each iteration the models are improving. The limitations need to be considered when applying AI models in threat modelling, as it can impact the quality and reliability of the threat model. In essence, a badly implemented AI could produce threat models that serve no value and leave more work for the threat modelling team than previously, as they will need to redo the work, and spend time tracing the conclusions of the AI in the tool.

5 Reflections of AI Usage

The last section only discussed the limitations inherent in the models, whereas this section will look at limitations stemming from the context threat modelling tools operate in. An AI-driven threat modelling tool will need to work with humans, and it is important that it maintains the security and privacy of the threat model itself.

5.1 Privacy Concerns

Primary privacy concerns stem from users sharing sensitive information, either Personally Identifiable Information (PII) or organisational secrets. When this data is used for retraining, it could make the model reproduce the conversation, unintentionally sharing the sensitive information with other users [24].

A recent example of this was the reveal of 2,000 hardcoded credentials in code generated with GitHub Copilot, making it easy for an attacker to steal and reuse them to gain access to secret data and manage subscriptions. The issue stems from the training data being public GitHub repositories, which contained hardcoded credentials that Copilot was merely replicating [26].

The most effective approach currently available to control the shared information is for the organisation to self-host their own version of the AI model. Self-hosting would allow them to be the data owners, and thereby be in complete control of what happens to the data. Some organisations do not have the money or ability to self-host, and instead they rely on policies, privacy guarantees and awareness training to ensure their employees know, how to properly use the AI assistants. This is not a complete solution as it relies on employees following guidelines, which can not be ensured [24, 33].

In addition to concerns about sharing their own secrets, the organisations need to consider possible infringement of privacy rights, when using AI assistants. The organisations will need to comply with regulations in their country of operations, in the EU this would entail GDPR and the AI Act. These laws set restrictions on the use cases, and data, they are allowed to share with an AI, and they might need explicit consent from an individual before sharing anything with an AI assistant [55].

5.2 Security Concerns

As for security, the responses from the AI can lack proper security measures, and often assistants will need to be specifically asked to leverage security in their answers [24]. The exact security impact of using Github CoPilot for code generation was investigated by Pearce et al. in 2022 [32] and then replicated in 2023 by Majdinasab et al. [26]. In both studies, they had Github CoPilot complete a piece of code to see, if it would introduce vulnerabilities in previously secure code. Their results showed that CoPilot does introduce more vulnerabilities, however, the newer study found that some of the previously discovered vulnerabilities did not persist. Even though, Github has managed to remove some of the vulnerabilities, the amount had not changed over time, indicating that the old ones had merely been replaced. The studies highlight the need for scepticism and proper review of AI generated code, which can be generalised for applying AI to solve any given task, in our case a threat model.

Besides the already inherent security issues in AI responses, attackers can further corrupt the AI assistants with poisoning attacks through crafted user inputs, or corruption of training data to make the assistants purposefully suggest insecure responses. An example of this is package hallucinations, where the AI will use fake or old packages containing attacker-written code, adding a backdoor to your system [33]. Again, these attacks rely on users' negligence, in which they choose to blindly trust the AI responses, and choose to copy the responses straight into their systems without thinking twice.

5.3 Human and AI Concerns

As witnessed, responsible AI usage falls on the users, which relies on individual judgement to safeguard security and privacy of the organisation. Several studies have been completed to discover the users' ability to scrutinise AI assistants, some testing them through coding tasks and others purely through interviews.

One of the initial studies was by Perry et al. [33] modelling users' security outcomes with and without AI assistants. In their study, they divided 47 participants into two groups, with and without AI access, and they had them solve five programming tasks. Their finding was that participants with access to AI assistants produced less secure code, whilst they believed that their code was more secure, because of the AI. Upon further investigation, it seemed that most of the participants blindly accepted the AIs solution out of negligence or lack of knowledge for solving the task. Thereby, leading them to accept the AI-generated solution without proper validation.

As a side, the study found that participants, who invested more time in creating prompts, providing helper functions, and adjusting parameters, discovered a secure solution. This indicates that deriving correct responses from AIs is possible, but relies on the user doing there due diligence [33].

Later in 2024, Klemmer et al. [24] conducted 27 semi-structured interviews with software engineers, team leads, and security testers, and they reviewed 190 Reddit posts to uncover the level of trust and usage of AI, specifically for security related tasks. All participants used AI on a daily basis, mostly for generating smaller code pieces or fixing buggy code, however the non-security experts would also seek its advice on security related topics, as they found themselves falling short on correctly assessing these elements. As underlined by the comment [24]:

"For the security point, there are a lot of checks that maybe, as a developer, I couldn't be aware of. Security is exactly one of those points that are not for humans, because I believe a lot in machine solutions."

While this sounds very trusting, all participants displayed a great distrust in the answers provided by the AIs, in which they have to manually vet the answers, and rewrite most of the generated code the AI has provided. The distrust is heaviest among the security experts, who view it more as a hindrance than a help, however they still use it on a daily basis [24].

Compared to the study by Perry et al., the participants display great awareness of AI, and they seem to be taking the necessary precautions. The varying results could indicate that users are becoming more aware, as time has passed, and they are now more familiar, not only with the technology, but also its limitations. In order to prove this theory, however, the participants of the newer study should be tested on their actual AI usage, because as proven by the earlier study, users can have a wildly different perception of their abilities compared to the truth.

Similar to Perry et al., Klemmer et al. concludes that the responsibility of AI usage is solely on the user, and the organisations should implement proper measures to ensure the quality is kept. These measures are similar to the processes already seen in software development through peer-programming, code reviews and static analysis checks. At the same time, AI is not valued as a mature enough technology to work on its own merits, and it should be supervised by a human for now. In the future, as the technology gains more experience, they expect it to completely handle mundane tasks, leaving the complex thinking for the humans [24, 33].

In threat modelling, the threat elicitation and mitigation suggesting process can be perceived as a mundane task of sifting through possible threats and mitigations in a pre-made knowledge base. The decision of which threats are actually relevant and how to prioritise them, is less straight forward and perhaps this part of the process is better handled by a human.

6 Methodology

AI-driven threat modelling tools need to automate the mundane phases of threat modelling, and leave the thinking to the humans, airing on the side of caution to not overstep in what they can actually achieve. We have seen the limitations they bring and how these can be partially solved by the owners of the tools, however none of the solutions manage to truly fix the issues. Thereby, the question remains, if AI-driven threat modelling tools have any merit, and can add value to a more novice and experienced threat modelling team. The AI-driven tools need to create reproducible and sound threat models matching those of a human threat modelling team. Beforehand, the tool must be applicable to a system, and perform the necessary phases to produce a threat model.

Our review will establish the general features of the tools, and then evaluate a threat model generated by each tool. The review will be organised as:

1. Select a set of AI-driven threat modelling tools
2. Gather information about each tool, like documentation, technical papers, patents and repositories
3. Measure general features of each tool in the set
4. Collect a set of threat models for a case study
5. Apply the tools on the system defined in the threat model set
6. Measure each generated threat model against the original threat model

In reviewing the AI-driven threat modelling tools, we recognise that tool suitability varies depending on system complexity and organisational context. Therefore, it is not a comparative review, but an informational one, focusing on highlighting the features of each tool.

6.1 Framework

The measurements are structured using the Goal-Question-Metric Framework (GQM). It is a top-down approach to evaluate quality of software, dividing the problem into a conceptual, operational and quantitative level. We saw the framework used in [49], where they applied it to discover the extend of automation and CICD integration readiness for a set of threat modelling tools. Other possible framings would be a Pugh Matrix, as used in [44], and a table structure, like [41], however we find these do not suit our review and reason for using a framework. The Pugh Matrix is a comparative approach to finding the most suitable entity in a given situation, thereby not informationally oriented, and a table does not actually provide any structure, but is merely a way of displaying the results. GQM is specifically developed for assessing software tools, and it imposes no constraints on the types of metrics that can be applied through the review. Each goal is defined according to the structure shown in Table 3, and then decomposed into questions and metrics. The relationships between goals, questions, and metrics form a tree structure, where the metrics answer the questions, and when all questions are answered the goal is fulfilled [50, 25].

For the purpose of	Understanding, controlling, improving the object
With respect to	The quality focus of the object that the measurement focuses on
From the viewpoint of	The people that measure the object
In the context of	The environment in which measurement takes place

Table 3: Structure for defining goals [23]

6.2 Research Goals, Questions & Metrics

Following the review's structure, the area under investigation is divided into two parts: *G1:* Threat Modelling Tool and *G2:* Generated Threat Model. G1 corresponds to the third step, focusing on tool features, and G2 applies to step six comparing the generated threat models to an original threat model. We use the structure devised in Table 3, to create our goals as seen in Table 4. The following subsections elaborate on the questions and metrics for each of the two goals.

Area of Investigation			
Threat Modelling Tool		Generated Threat Model	
For the purpose of	Inspecting	*For the purpose of*	Verifying
With respect to	Features	*With respect to*	Soundness
From the viewpoint of	Project Management	*From the viewpoint of*	Project Management
In the context of	Software Development	*In the context of*	Software Development

Table 4: Research Goals

6.2.1 Threat Modelling Tool

G1: Inspect the features of AI-driven threat modelling tools to be used by a software project team

In previous studies: [49], [14], [41] and [44], they found that it is not possible to quantitatively evaluate the features of the tools, and they developed qualitative evaluation metrics to assess them instead. We will use their metrics as the offset for creating our own, in order to establish the needed basic threat modelling features and then extend the tool criteria with some AI specific ones. Shi et al. [41] devised a taxonomy for gathering information about threat modelling tools, and through their study, they found that their criteria managed to highlight important differences between the tools. Therefore, we will transform their criteria into metrics that will fit AI-driven threat modelling tools. In their taxonomy, they divided the features of a threat modelling tool into two sets: functionality and general aspects criteria, and we will do the same. The questions to measure goal *G1* are therefore:

- *Q1:* Which functionalities do AI-driven threat modelling tools have?
- *Q2:* Can AI-driven threat modelling tools be applied in a software project?
- *Q3:* What AI is applied in AI-driven threat modelling tools?

Starting with *Q1*, Shi et al. developed a separate criteria capturing information about each of the threat modelling phases, i.e. the input format, threat catalogue, mitigation library and evaluation method. Instead of capturing information, we want to measure who can perform the phases, to see

how much control the user has over the tool. As contrary to other threat modelling tools, an AI could fully replace a threat modelling team, however given the previously discussed limitations, this might not produce the best threat model. For the software project viewpoint in *G1*, we add metrics for the input, output and overall threat approach. The team choosing between the tools will then know what information they need to provide, what information they can get out of the tool, and what the threat model will focus on. Thereby, our metrics will cover the same scope as Shi et al., but from a broader point of view, ensuring that we still cover the functionalities of a threat modelling tool. For question *Q1* the metrics will be:

Question & Metrics

Q1: *Which functionalities do AI-driven threat modelling tools have?*

M1: Input Type (DFD, Text, Code,...)
M2: Threat Approach (STRIDE, LINDDUN, ...)
M3: Modelling Phases (Describe, Elicit, Assess, Mitigate)
M4: Output Format (Report, Compliance, Tests, ...)

The general aspects chosen by Shi et al. details the development progress, open/closed source and how the user can access the tool, these remain equally relevant for AI-driven threat modelling tools. In their study, they emphasize trust as an important factor for selecting an appropriate tool, and state "only open source tools can properly be scrutinized and vetted" [41]. We can see value in both open and closed source tools, as you either trust the organisation who's revenue stream is tied to the success of the tool, or you trust the community and yourself to ensure the tool lives up to your standards. This means that the open versus closed source discussion is more nuanced than presented by Shi et al., in which we instead focus on the control and insight a user can have in the development of a tool.

For the other criteria, we do not make any adjustments, however the possible values for them will be slightly different. For the development progress, we track the original creation date, and when the AI was added to the tool. Whereas, the access to the tool will not detail if it is a web application, installable software and so on, but it will measure the editions, like enterprise and community. Thereby, the metrics for *Q2* becomes:

Question & Metrics

Q2: *Can AI-driven threat modelling tools be applied in a software project?*

M5: Release (First release, AI release)
M6: Editions (Enterprise, Community, Private, ...)
M7: Controllability (Forkable, Changeable, Viewable, ...)

For *Q3*, we need to develop our own set of metrics to assess the AI model in the tools. The metrics will cover the basic elements of AI models as discussed in Section 4.1: pre-training, fine-tuning and data processing. We expect that the tools use pre-trained models from one or more AI providers i.e. Google's Gemini, Meta's Llama, OpenAI's GPT models. The model provider enables us to further investigate the AI's capabilities through benchmarks available online, and thereby understand the impact the different models could have on the threat model. This then leads to the next metric for the number of AI models applied in the tool, as combining models with different skill sets could make sense for a threat modelling tool, for example using Gemini to search to correctly identify the threats,

and then GPT could formulate it in a user-friendly way. Thereby, benefitting from the knowledge of Gemini, and the text generation skills of GPT.

Fine-tuning is restricted to pre-prompting of the AI models, even though [20], [54] and [12] all used a mix of post-training datasets and pre-prompting. The articles did not show significantly better results than [43], which only used pre-prompting, whilst the latter study saw a significant change in outcome after pre-prompting a GPT model. The data processing relates to the privacy concerns expressed in Section 5.1, an organisation needs to know how their data is processed by the tool and AI, even though most of the information shared when threat modelling would not contain sensitive information. An organisation could have other reasons for not wanting to share their system architecture with others, like competitors or attackers, and letting an AI use their system for retraining could have others replicating their system. The metrics for *Q3* will be:

Question & Metrics

Q3: *Which AI is applied in the tool?*

M8: AI Model (Google, Local, Meta, ...)
M9: Number of Models (1, 2, ...)
M10: Pre-prompting (Persona, Context, ...)
M11: Data Processing (Disclosed, Not-Disclosed, ...)

GQM Tree

The GQM tree for goal *G1* is displayed in Figure 1. The figure shows the hierarchical connection between each of the 11 metrics, 3 questions and the goal.

```
G1: Inspect the features of AI-driven threat modelling tools to be used by a
software project team
|
|__ Q1: Which functionalities do AI-driven threat modelling tools have?
|   |__ M1: Input Type
|   |__ M2: Threat Approach
|   |__ M3: Modelling Phases
|   |__ M4: Output Format
|__ Q2: Can AI-driven threat modelling tools be applied in a software project?
|   |__ M5: Release
|   |__ M6: Editions
|   |__ M7: Controllability
|__ Q3: Which AI is applied in AI-driven threat modelling tools?
    |__ M8: AI Model
    |__ M9: Number of Models
    |__ M10: Pre-prompting
    |__ M11: Data Processing
```

Figure 1: GQM tree for Goal G1

6.2.2 Generated Threat Model

G2: Verify the soundness of a generated threat model from an AI-driven threat modelling tool to establish the value it brings to a software project team

The soundness of a generated threat model is not defined in literature, and generally measuring the threats of a threat model in relation to its system is not possible. Threat modelling is a subjective activity and one cannot define a true threat model for a system. Instead, you can draw general assumptions about the validity of a threat model, and verify its soundness in relation to another threat model on the same system, thereby objectively defining validity and soundness. Before we proceed with defining validity and soundness, we decide to confine ourselves to threat modelling tools applying a mnemonic threat approach (STRIDE, LINDDUN, ...), which allows us to make assumptions about the relation between generated threats and threat categories in the threat approach. From this, we can observe that a threat is confined to the categories in the threat approach, and it should pertain to an asset in the system. Then for a specific system and threat approach, we can define a valid threat model as:

Validity: *A threat model is valid iff every threat is mapped to a category in the threat approach and asset in the modelled system.*

Building on this, we then proceed with defining a sound relation between two threat models for the same system, where one will be the generated threat model. According to *G2.* the other threat model should capture the viewpoint of a software project team, and the relation should capture the benefit of applying a tool. A team would benefit from the tool, if it can recreate their threat model, and suggest valid threats that they have missed. Then, the team would be able to trust the threat model the tool created, leaving less work for them to review the threat model. The relation needs us to define a minimal set, threats a team has found, and a maximal set, all valid threats. The minimal set would be a threat model created by the team selecting between the tools, whereas the maximal set can be derived from the validity definition of a threat model. Using the validity definition will allow for subjectively irrelevant threats to be present in a sound threat model relation, however objectively the threats are all equally relevant if they are valid. We define the soundness relation between two threat models according to the minimal set as:

Soundness: *A sound relation exists between two threat models iff they are valid threat models for the same system and threat approach, and one threat model is equal to or a subset of the other threat model.*

The definition allows either threat model to be contained in the other, but in our case the generated threat model should be the superset. We then proceed with the questions for *G2* which should capture the validity and soundness of the threat models:

- *Q4:* Can AI-driven threat modelling tools produce valid threat models for a software system and mnemonic threat approach?
- *Q5:* How sound is the relation between a team's and an AI-driven tool's threat model for the same system and threat approach?

For *Q4*, the validity is bound by the threat categories and assets, therefore we need to verify the extend of these in the generated threat model. An AI can hallucinate both assets and categories, and tools that allow the AI to have full control over the elicitation phase will be vulnerable to this. Some tools

might use other techniques for elicitation, more similar to what is seen in automatic tools using rule engines, or they leave it in the hands of the user, thereby we would expect that no hallucinations will appear in the measurements. For the validity the assets and categories are measured separately, however for the threat model to be valid, all the threats need to adhere with both. The metrics for *Q3* are:

Question & Metrics

Q4: *Can AI-driven threat modelling tools produce valid threat models for a software system and mnemonic threat approach?*

M12: Valid Categories
M13: Valid Assets

The soundness is measured between a team's threat model and an AI-driven tool's threat model, where we expect that each threat will contain a category, asset, description, mitigations and risk score. For two threats to be exactly the same, they should align on each element, however given the subset relation between them, the fields containing a set, like mitigations, only need to be a subset of one another. The soundness is then divided to have one metric for the category, asset and description, and separate metrics for mitigations and risk score. The grouping of category, asset and description stems from validity, where these are perceived as the elements of a valid threat, and therefore they should remain as such for the soundness. For mitigations and risk score, these should only be evaluated on threats sharing category, assets and description, as we cannot expect that threats where these differ share the same mitigations and risk score. Thereby, the metrics for *Q5* becomes:

Question & Metrics

Q5: *How sound is the relation between a team's and an AI-driven tool's threat model for the same system and threat approach?*

M14: Same Threats
M15: Same Mitigations
M16: Same Risk Score

GQM Tree

The GQM tree for goal *G2* is displayed in Figure 2. The figure shows the hierarchical connection between each of the 5 metrics, 2 questions and the goal.

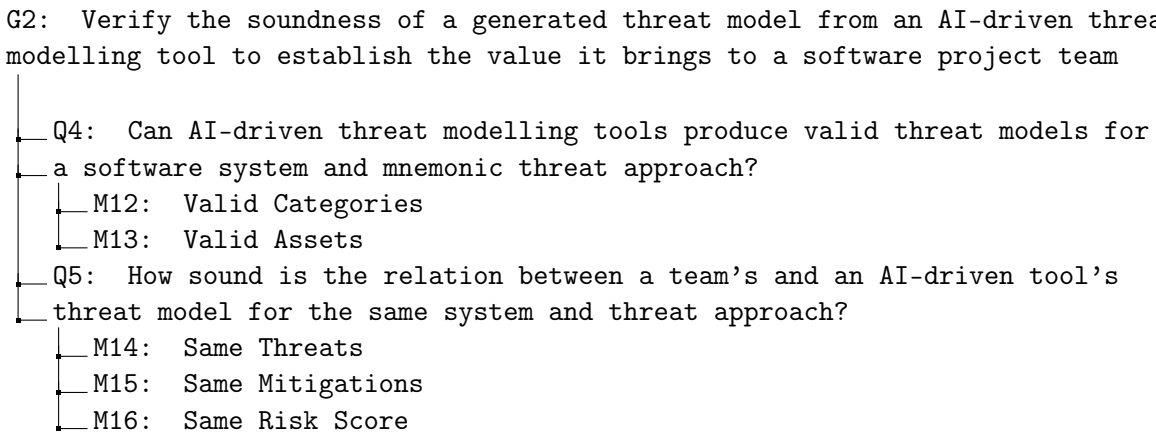


Figure 2: GQM tree for Goal G2

6.2.3 Evaluation Metrics

G1 has a total of 11 metrics, and *G2* has 5, as described in the previous section. For each metric, a function or set is created to measure them against a specific tool t or threat model TM. The metrics are not defined for a specific set of tools, but they should be applicable to all AI-driven threat modelling tools. The actual set of tools will be defined in Sections 7 and 8.

M1: Input Type

$$input_{type}(t) = \begin{cases} + & t \text{ can combine input } type \text{ with other possible inputs} \\ | & t \text{ can at most take input } type \text{ at any time} \\ - & t \text{ cannot use input } type \end{cases}$$

where $type \in \{diagram, template, description, questionnaire, asset, code, prompting\}$

The $input_{type}$ function specifies the possible inputs of the tool as a whole, not the AI, and it captures the tool's ability to combine multiple input types at once. The possible inputs are given by the *type* set, and we define each of the input types as:

- Diagram, *the user creates a visualisation of the system, typically a DFD or architectural diagram.*
- Template, *the user chooses a diagram from predefined architectural templates in the tool.*
- Description, *a loosely defined text written by the user.*
- Questionnaire, *a set of questions given by the tool for the user to answer.*
- Asset, *a separate list of important components in the system, like specifying data assets and flows.*
- Code, *a link to a repository for the code of the system.*
- Prompting, *a user interactively discusses the system with the AI prior to threat elicitation.*

The inputs above capture the effort needed from the user, before they can begin using the tool. Some of the tools allow the user to draw the diagram directly in the tool, however this will not be counted as input, but instead as part of the define phase. For the prompting, it is possible for the creators to pre-prompt the AI model, but this is measured in the $M10 : Pre - prompting$ metric instead.

M2: Threat Approach

$$approach(t) = \begin{cases} ta & t \text{ uses threat approach } ta \\ ? & t \text{ does not disclose which threat approach is used} \end{cases}$$

The *approach* function gives a textual output either the name of the threat approach the tool uses and if the approach is unknown a ? is displayed instead. We previously explained threat approaches in Section 3.1.

M3: Modelling Phases

$$modelling_{phase}(t) = \begin{cases} + & t \text{ allows the entities to assist each other in completing } phase \\ | & t \text{ allows either entity to complete } phase \\ A & t \text{ only allows the AI to complete } phase \\ U & t \text{ only allows the user to complete } phase \\ R & t \text{ only allows non-AI automation to complete } phase \\ - & t \text{ does not complete } phase \end{cases}$$

where $phase \in \{describe, elicit, assess, mitigate\}$

The $modelling_{phase}$ function highlights who/what can affect the completion of each threat modelling phase. The *phase* set aligns with the threat modelling process described in Section 3. Completion only counts, if the tool calculates or allows for adjustments by either of the possible entities: user, AI or non-AI automation. For each of the phases completion is defined as:

- Describe, *the tool automates diagram creation, have a build in canvas or questionnaire for the user to fill out.*
- Elicit, *the tool has a page/button that elicit threats.*
- Assess, *the tool has a page/button that calculates/defines a risk score or prioritises the elicited threats.*
- Mitigate, *the tool has a page/button that shows possible mitigations for a threat.*

M4: Output Format

$$output_{format}(t) = \begin{cases} + & t \text{ provides } format \text{ as output} \\ - & \text{otherwise} \end{cases}$$

$$\text{where } format \in \{report, compliance, tests, integration\}$$

The $output_{format}$ function describes how the user can extract or integrate the generated threat model. The possible output in $format$ are:

- Report, *a PDF, CSV or similar exportable overview of the threats, mitigations and assessment.*
- Compliance, *an audit log describing the system's adherence to specific security and privacy standards.*
- Tests, *runnable format that can validate the existence of a threat or extend of a mitigation in the implemented system.*
- Integration, *creating issues in a separate project management tool, like Jira or Azure DevOps.*

The above formats are applicable after the tool has finished creating the threat model, and they can assist the user with documentation, compliance and ease integration into current processes. Other than the above formats, we expect all tools to provide an in-tool overview of the threats, mitigations and assessment, therefore, this is not tracked as part of this metric.

M5: Release

$$release(t) = \begin{cases} (d1, d2) & t \text{ was first released on date } d1, \text{ and then AI was/will be added on date } d2 \\ d & t \text{ was/will be released on date } d \\ - & \text{otherwise} \end{cases}$$

The *release* function specifies the initial release date of the tool, and for tools that have previously been released without AI, it provides the release of the AI addition as well. Some tools might not have a release date, and to indicate their unavailability we mark them with "−".

M6: Editions

$$edition_{type}(t) = \begin{cases} + & t \text{ is provided as } type \\ - & \text{otherwise} \end{cases}$$

$$\text{where } type \in \{private, community, enterprise\}$$

The $edition_{type}$ function outlines the possible options for obtaining access to the tool. The editions are captured in *type*, and they are defined as:

- Private, *a user-only edition of the tool with AI.*
- Community, *a non-paid edition of the tool with AI.*
- Enterprise, *a paid edition of the tool with AI.*

M7: Controllability

$$controllability(t) = \begin{cases} +3 & \text{Anyone can legally fork } t\text{'s source code} \\ +2 & \text{Anyone can make changes to } t\text{'s source code} \\ +1 & \text{Anyone can view and ask for changes to } t\text{'s source code} \\ 0 & \text{No-one can view } t\text{'s source code} \end{cases}$$

The *controllability* function measures the amount of knowledge and control a user can have for a tool. A +3 is awarded, if the tool's source code can legally be forked, allowing a user to fully control the tool's implementation, and thereby ensure that it fits their specific use case. Whereas, a 0 is for tools that do not disclose any information of their inner workings to the users, and the users will need to fully trust the creators. For +3 and +2, forking is scored higher, as the user will be in full control, whereas changes to publicly shared source code can be changed by the current user or other organisations using the tool. Therefore, the user cannot necessarily tailor it to their needs, and others could make changes which conflict with the users needs.

The numerical scale is not meant to fort any tool as better, as the need for controllability is situation dependent. Some organisations prefer to completely host their own tools in-house, and others would rather not have the extra overhead of a self-hosted solution.

M8: AI Model

$$AI_{model}(t) = \begin{cases} + & t \text{ can combine } model \text{ with other models to produce a threat model} \\ | & t \text{ can use one or more of the same } model \\ - & \text{otherwise} \end{cases}$$

where $model \in \{google, local, mistral, openAI\}$

The AI_{model} function will determine the AI model used in the tool, and it is evaluated for each of the possible model providers as discussed in Section 4. The tools can allow the user to choose between different models, and even use multiple models at a time, this behaviour is reflected in the cases of the function. The *model* set contains a small subset of all model providers, and more can be added to show more variance of models in the tools. If a tool does not disclose, which model they use, the entire metric will be "−".

M9: Number of AIs

min_{ai} the least amount of AIs needed for a tool t to produce a threat model
max_{ai} the maximum amount of AIs for a tool t to produce a threat model

Some of the tools can use more than one AI model at a time, and to present the range, we define to variables. min_{ai} is the minimum number of AI models, and max_{ai} is the maximum number of AI models the tool uses at once to produce a threat model. It is possible for min_{ai} to be zero, if the tool is able to generate a threat model without using any AI.

M10: Pre-prompting

$$prompt_{theme}(t) = \begin{cases} + & t\text{'s AI is informed about } theme \\ - & \text{otherwise} \end{cases}$$

$$\text{where } theme \in \{persona, context, architecture, security, privacy\}$$

The $prompt_{theme}$ function elicits pre-prompting information fed directly to the AI model prior to threat elicitation. The content of the prompts are grouped into overall themes, based on words mentioned in the prompt. The *theme* set contains the groupings:

- Persona, *mentions a specific organisational role to subsume.*
- Context, *describe external inflictions on systems, like the organisation itself.*
- Architecture, *specify to obtain knowledge about cloud, servers, file systems and so on.*
- Security, *ask to suggest solutions with security in mind.*
- Privacy, *ask to suggest solutions with privacy in mind.*

It is possible to supply this information in one or multiple prompts, but we choose not to concern ourselves with this. The functionality is not interactive, and therefore the prompts will not change depending on the AI's response, if multiple prompts are given.

M11: Data Processing

$$processing(t) = \begin{cases} +2 & t \text{ has a legal policy stating how the data is processed by the tool and AI} \\ +1 & t \text{ discloses how data is processed by the tool and AI} \\ 0 & t \text{ discloses how data is processed in the tool} \\ -1 & t \text{ discloses how data is processed in the tool, but state they cannot control the AI} \\ -2 & t \text{ discloses no information about how data is processed by the tool and AI} \end{cases}$$

The *processing* function uses a scale, and the baseline is set to tools that disclose data processing procedures for the overall tool. Lower grades are tools that partially disclose the usage or allow for other services to freely control the information in unknown ways, mainly AI creators. Higher grades are rewarded for tools that fully disclose data processing, and impose a legally binding data processing policy for the tool and AI.

M12: Valid Category

Let $TM = \{th_1, \ldots, th_j\}$ where th_j is the j^{th} threat in a threat model,
and $th_j = (c_j, a_j, t_j, m_j, r_j)$
Let $TA = \{ta_1, \ldots, ta_k\}$ where ta_k is the k^{th} threat category in a mnemonic threat approach
Let $f_c : (th_i) \mapsto \{0, 1\}$ be defined for $th_i = (c_i, a_i, t_i, m_i, r_i)$ such that

$$f_c(th_i) = \begin{cases} 1 & \text{if } c_i \notin TA \\ 0 & \text{otherwise} \end{cases}$$

$$C = \{th_n \in TM \mid f_c(th_n) = 0\}$$

The set TM contains all the threats in a threat model, and every threat contains: category c, asset a, threat description t, mitigations m and risk score r. The TA set contains the threat categories of a mnemonic threat approach, i.e. for STRIDE the set would be $\{"Spoofing", "Tampering", ..., "Elevation\ of\ Privilege"\}$. We then define a function f_c mapping the correlation between the categories in TM and TA, and we use it to create the set C. The set C will contain all the threats, where the threat's category is not a member of TA. In this case, f_c will evaluate to zero. For a threat model to have valid categories C should be the empty set.

M13: Valid Asset

Let $TM = \{th_1, \ldots, th_j\}$ where th_j is the j^{th} threat in a threat model,
and $th_j = (c_j, a_j, t_j, m_j, r_j)$
Let $S = \{sa_1, \ldots, sa_k\}$ where sa_k is the k^{th} asset of the modelled system
Let $f_a : (th_i) \mapsto \{0, 1\}$ be defined for $th_i = (c_i, a_i, t_i, m_i, r_i)$ such that

$$f_a(th_i) = \begin{cases} 1 & \text{if } a_i \notin S \\ 0 & \text{otherwise} \end{cases}$$

$$A = \{th_n \in TM \mid f_a(th_n) = 0\}$$

We define TM as the set containing all threats in a threat model, similar to *M12*, and the set S containing all the assets of the system. S can contain elements like "Web Server", "Database" and "User", thereby S is not confined to components in the system, but also accounts for external entities, which should be protected. The function f_a defines the correlation between the assets in the threats in TM and the assets in S, and it is applied when creating the set A. A contains all the threats where the asset is not in S, thereby the cases where f_a evaluates to zero. The set A should be the empty set, if all the threats pertain to valid assets.

M14: Same Threats

Let $TM_{te} = \{th_1^{(te)}, \ldots, th_j^{(te)}\}$ where $th_j^{(te)}$ is the j^{th} threat in a team's threat model
and $th_j = (c_j, a_j, t_j, m_j, r_j)$
Let $TM_{ai} = \{th_1^{(ai)}, \ldots, th_k^{(ai)}\}$ where $th_k^{(ai)}$ is the k^{th} threat in a tool's threat model,
and $th_k = (c_k, a_k, t_k, m_k, r_k)$

$\sim_t$:= describe the same malicious objective and impact in t
Let $f_t : (th_i, th_l) \mapsto \{0, 1\}$ be defined for $th_i = (c_i, a_i, t_i, m_i, r_i), th_l = (c_l, a_l, t_l, m_l, r_l)$ such that

$$f(th_i, th_l) = \begin{cases} 1 & \text{if } a_i = a_l \text{ and } c_i = c_l \text{ and } t_i \sim_t t_l \\ 0 & \text{otherwise} \end{cases}$$

$$T_{all} = TM_{te} \times TM_{ai}$$
$$T = \left\{ \left(th_i^{(te)}, th_l^{(ai)} \right) \in T_{all} \mid f_t \left(th_i^{(te)}, th_l^{(ai)} \right) = 1 \right\}$$

TM_{te} and TM_{ai} contain the threats of two different threat models on the same system, and each element in the sets is denoted with ^{te} or ^{a}i to make it clearer which set the element originates from. f_t is defined for two threats, and it compares the category, asset and description of the threats,

the category and asset should be the same, and the description should describe the same malicious objective. An example of this would be *"Flooding the server with messages"* and *"Denial of service attack on the server"*. We perform a full comparison between the two threat models, and the pairs are collected in TM_{all}. The set T is build from TM_{all}, and it only contains the threats from the two threat models, where f_t evaluates to one. Two threat models only have sound threats if for all threats in TM_{te} their exists at least one pair in T containing each of the threats.

M15: Same Mitigations

$\sim_m$:= describe the same or overlapping method(s)

Let $f_m : (th_i, th_l) \mapsto \{0, 1\}$ be defined for $th_i = (c_i, a_i, t_i, m_i, r_i), th_l = (c_l, a_l, t_l, m_l, r_l)$ such that

$$f(th_i, th_l) = \begin{cases} 1 & \text{if } m_i \sim_m m_l \\ 0 & \text{otherwise} \end{cases}$$

$$M = \left\{ \left(th_i^{(te)}, th_l^{(ai)} \right) \in T \mid f_m \left(th_i^{(te)}, th_l^{(ai)} \right) = 1 \right\}$$

We reuse the set T from *M14*, as the mitigations should only be the same if the threats are. f_m is defined for two equal threats, and it compares the mitigation suggestions of the threats. It is possible for the mitigations to contain more than one suggestion, and we accept if the suggested methods are somewhat overlapping, similar to the subset relation between the two threat models. The set M will contain all of the threat pairs where the mitigations are the same, meaning that f_m evaluates to one. Combining the T and M set, then if all threats in TM_{te} are present in at least on threat pair in both sets, then the threat models contain only sound threats and mitigations.

M16: Same Risk Score

Let $f_r : (th_i, th_l) \mapsto \{0, 1\}$ be defined for $th_i = (c_i, a_i, t_i, m_i, r_i), th_l = (c_l, a_l, t_l, m_l, r_l)$ such that

$$f(th_i, th_l) = \begin{cases} -1 & \text{if } r_i < r_l \\ 0 & \text{if } r_i = r_l \\ 1 & \text{otherwise} \end{cases}$$

$$R_{less} = \left\{ \left(th_i^{(te)}, th_l^{(ai)} \right) \in T \mid f_r \left(th_i^{(te)}, th_l^{(ai)} \right) = -1 \right\}$$
$$R_{same} = \left\{ \left(th_i^{(te)}, th_l^{(ai)} \right) \in T \mid f_r \left(th_i^{(te)}, th_l^{(ai)} \right) = 0 \right\}$$
$$R_{more} = \left\{ \left(th_i^{(te)}, th_l^{(ai)} \right) \in T \mid f_r \left(th_i^{(te)}, th_l^{(ai)} \right) = 1 \right\}$$

We again reuse the set T from *M14* to only evaluate the risk score on threats that are the same. f_r is defined for two equal threats, and it compares the risk score of the threats. The function enforces an ordering of the two threats, and we create a set for each output of the function. R_{less} are all the threats in the team's threat model, where the risk score is lower than in the tool's threat model, R_{same} all the threats where it is the same, and R_{more} where the risk score is bigger in the team's threat model. We distinguish between same, more and less, because we want to see if the tool is more aggressive or passive in its risk assessment of the threats, compared to a threat modelling team. For the two threat models to be completely sound, T, M and R_{same} all need to contain at least one threat pair for all threats in TM_{te}.

6.3 Case Study Design

For *G2*, the tools shall be applied to an example system, and then compared to the threat model of the team selecting between the tools. This generalisation is achieved by selecting a set of example threat models to stand in for a team's threat model. The selected examples might not be created by the same team, and they may vary in quality, however it will illustrate how the analysis would work if a real threat modelling team were to test our metrics against their threat model. From this point on, we refer to the team's threat model as the *baseline*, and the model produced by a tool as the *generated* threat model.

To apply the metrics, we propose the approach shown in Figure 3, and by following this, other researchers would be able to reproduce our results. The approach is only viable for AI-driven threat modelling tools, as it requires full automation of all four threat modelling phases: describe, elicit, assess, and mitigate. The approach consists of four steps: Define, Apply, Judge, and Measure, and prior to applying the approach, the tools, a baseline model and a set of judges need to have been selected. It is best to select three or more judges, as this will improve reliability of the assessment. The approach is applied to a set of tools on a single baseline threat model, and after the Define phase, it will iterate over the list of tools, to create, judge and measure the generated threat model. It is important that the Define and Apply phases follow the same procedures for all tools, thereby controlling variance of the results, making them equally biased and objective. This follows from the tools being controlled by different users, and for the approach, we should mimic the same user applying the tools on a given system. For the approach, we emphasise that the purpose is not to judge how good a threat model is, or compare the tools with each other. Instead the goal is to gain insight into how well the tool can capture some baseline, in our case that of a threat modelling team.

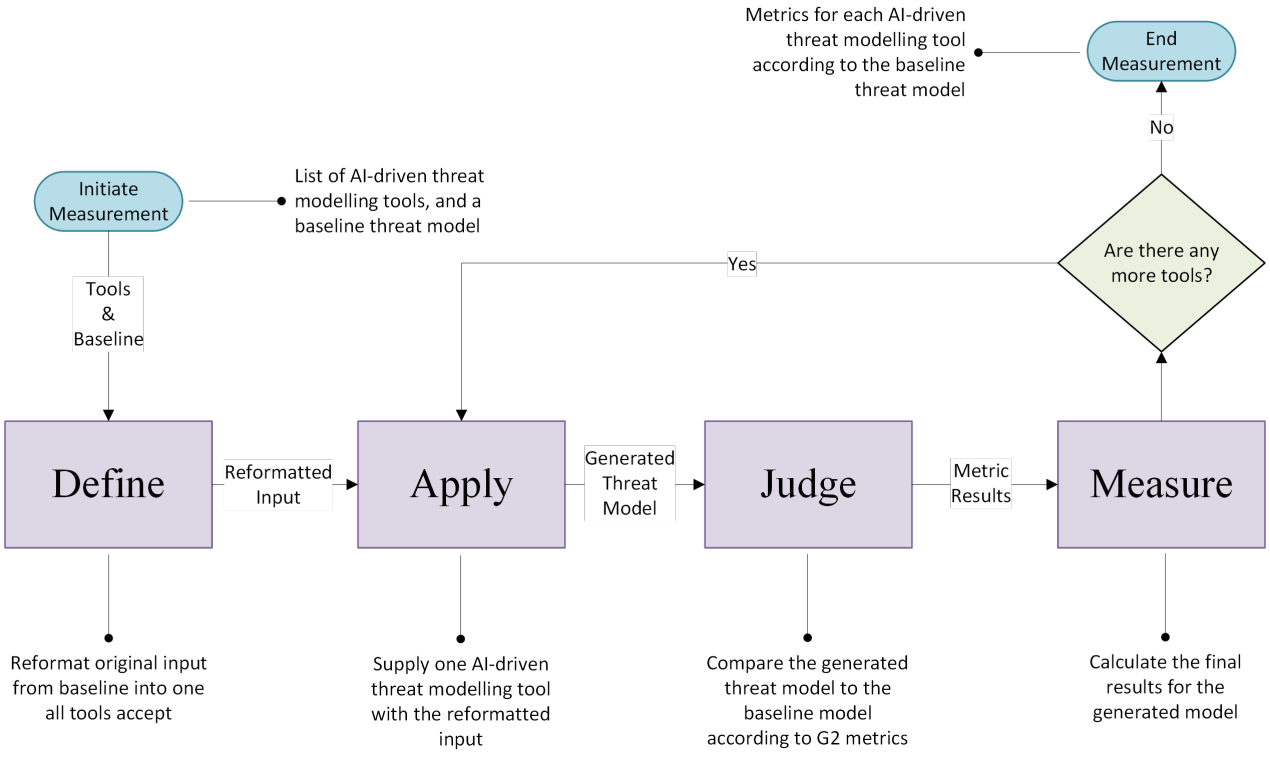


Figure 3: Approach for Case Study

The Define phase gathers the relevant input from the baseline model and reformats it into one or more forms that all the tools can accept. Examples of possible inputs for the tools would be DFD, text description, questionnaire, code and more. The reformatted input needs to capture the same details as the original one, and failing to do so can directly affect the results. Then, in the Apply phase a tool from the tool list is asked to generate a threat model based on the reformatted input, and the user should only supply the input, and not otherwise interfere with the tool's generation process. Once the threat model is generated the pre-selected set of judges assess each metric *M12-M16* for the generated model, and in comparison with the baseline model. The assessments from the judges is then correlated based on the inter-rater reliability, this will tell us the degree of agreement between the judges.

The inter-rater reliability between three or more judges is measured by Fleiss' Kappa. This is a renowned measure of inter-rater reliability, and it is perceived as more correct than purely measuring the accuracy between judges. Fleiss' Kappa does not only account for the actual overlap between the judges assessments, but it accounts for chance agreement between the judges. Chance agreement is not used when calculating accuracy, and therefore if the judges guess a lot on the assessments, the inter-rater reliability would be skewed. Fleiss' Kappa is not a perfect measurement, and it is sensitive to outliers in the assessment, in which one judge has provided wildly different answers from the others. In this case, Fleiss' Kappa would be 0, because the chance agreement would be greater than the observed agreement.

In Fleiss' Kappa the actual agreement p_o and chance agreement p_e are calculated in two steps. p_o is the number of judges, who have assigned a specific outcome to a specific subject, and p_e the number of times the judges assigned a specific outcome. The calculation of the chance and actual agreement for Fleiss' Kappa is [19]:

Let s be the subjects, and t the evaluation outcomes,
Let M the number of judges and m the number of subjects
Then z_{st} is the number of judges who assigned outcome t to subject s

$$p_s = \sum_t z_{st}^2 - M * N \qquad p_t = \frac{1}{m * M} \sum_s z_{st}$$

$$p_o = \frac{1}{m * M * (M-1)} \sum_s p_s \qquad p_e = \sum_t p_t^2$$

In our case, the subjects will either be the threats in a generated model, or the threat pairs for a generated and baseline model. After the actual and chance agreement is calculated, the kappa value can be calculated [11, 19]:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

p_o is the actual agreement
p_e is the chance agreement

Fleiss' Kappa will be in the range -1 to 1, and a general interpretation of the scores is seen in Table 5. The Kappa score indicates the reliability of the assessments, not their correctness. For a negative or close to zero Kappa score the judges do not agree at all, and repeating the experiment with more judges, and/or reevaluating the assessment function to make it more stringent could improve the results.

After the assessment, we measure how valid and sound a single generated threat model and baseline is in the Measure phase. The phases Apply, Judge and Measure are then repeated for the rest of the tools. The approach is applied in Section 8, and the following subsections will detail the needed pre-requisites for applying the approach: baseline model(s), tool inclusion criteria and judge selection.

Kappa	Level of Agreement
> 0.8	Almost Perfect
> 0.6	Substantial
> 0.4	Moderate
> 0.2	Fair
> 0	Slight
< 0	No Agreement

Table 5: Level of Agreement according to Kappa score [11, 19]

6.3.1 Baseline Threat Models

The approach only requires one baseline model, but to showcase the tools more, a set of baseline models is chosen. For the set, we want to test the tools on different systems and teams to generalise our results more. Another option would have been to use various baseline models on a single system to only measure how different types of teams compare to the tools. However, this would not allow us to measure, if the tools are resilient on size and complexity, which when looking at AI and threat modelling limitations, seem to be of bigger relevance, than the subjectivity that a team brings. Furthermore, the metrics require a mnemonic-based threat modelling approach, for which STRIDE is selected, because it seems to be the most commonly adopted one. The selected baseline models should adhere with the following criteria:

BIC1: The threat model should be created by one or more humans.

BIC2: The threat modellers can apply a none AI tool as part of the threat modelling.

BIC3: The threat model should map the threats to STRIDE.

BIC4: For each threat, a description, category, mitigation, and risk score shall be present.

BIC5: The threat model should include a DFD and a short description of the system.

There exists several GitHub repositories[4][5] for publicly available threat models, and some companies also provide it on their websites. During the search, it was decided to slightly adjust criterion *BIC3*, as it was not possible to find more than one threat model, which adhered with all criteria. We can infer the STRIDE category based on our own familiarity with threat modelling, and this will not cause any implications with the other criteria. As we can be equally subjective in our threat categorisation as the original threat modellers. This decision may affect metric *M14*, which compares the category, asset and threat description, because it will depend on our ability to choose the correct threat category. The selected baseline models are summed up in Table 6 and their DFD's can be found in Appendix A. This is also noted in the DFD column in the table. The set is rather small, and in the future repeating the review with a bigger set would be preferable.

[4] https://github.com/TalEliyahu/Threat_Model_Examples
[5] https://github.com/OWASP/threat-model-cookbook

System	Description	DFD
Message Queue Application	A web application running in the user's browser, with a decoupled background process that retrieves messages from a queue and stores it in a database.	A.1
Web Application	A three tier web application with a web UI in the user's browser, which communicates with a web service and PostgreSQL database running in public cloud.	A.2
IoT Edge Devices	An application, which collects video frames from a video camera and processes the frames on a set of IoT Edge Devices. The processed frames are send to an Azure Cognitive Service to get the audio output. The processed data is then stored in Azure Blob storage.	A.3

Table 6: Description of systems to apply the threat modelling tools on

6.3.2 Tool Inclusion

The extend of AI-driven threat modelling tools is not yet fully known, and therefore a set of tool inclusion criteria (TIC) is created for later selection of the tools. Upon selecting the the baseline models, it was decided that STRIDE should be used, and that they have a DFD and textual description of the system. Thereby, the TICs are:

TIC1: The tool should accept a textual description as input.

TIC2: The tool should map the threats to the STRIDE categories.

Even though, the baseline models contain both a text description and DFD, it is reformatted into a single textual description by systematically describing the components, trust boundaries and flows in the DFD. The reformation is a source of error for our study, as we cannot completely ensure that it truly captures the same information as the original input. We can verify the extend of the error, by checking a tools DFD against the DFD in the baseline model, if they match the error should be slight or non-existent. The reformatted input for each of the baseline models is found in Appendix B.

6.3.3 Judges

For efficiency and reliability of our results, we choose to use LLM-as-a-judge, as it will allow us to have multiple judges assess the threat models in a timely manner. Otherwise, our assessment would be performed by a single human judge, for which the reliability cannot be measured, and repeating the review could easily provide very different results. The LLM-as-a-judge method automates the slow human assessment, and it aims to produce outcomes that match those of a human judge. The method's viability was shown in the study by Zheng et al. [58], in which they found that GPT-4 was able to achieve the same or higher inter-rater reliability as human-human evaluators. They ran their experiment for 3000 votes given by a group of experts, a separate set of human judges and GPT-4, in which GPT-4 achieved an agreement rate above 80% with the human evaluators. They did not use Fleiss' Kappa for their inter-rater reliability, but had their own method for randomly sampling the assessments of the different judges.

In the LLM-as-a-judge method, one or more LLMs are provided with a prompt asking them to assess one or more subjects against a set of criteria, and then assign a numerical score, category, ranking or extract a selection. The prompt can be point-wise, asking the judge to provide a score for a single

subject, or it can be pair-/list-wise asking the judge to compare two or more subjects [58]. In our case, we will perform point-wise assessments for validating a single threat model and pair-wise when comparing a generated model with a baseline.

Fine tuning & Examples

Prior to prompting the LLMs, it is possible to fine tune them on a pre-labelled or synthesised data set to enhance their understanding of the task at hand. Through the fine tuning, the LLMs can learn the subtle difference that leads to a specific result, and this should enhance the inter-rater reliability for a specific human judge. It can also be equally effective to provide a few examples of the subjects and correct outcome directly in the prompt, making the LLM learn on-the-go. Both methods depend on the quality and completeness of the dataset/examples. We choose to create a set of examples for each prompt showing how each possible outcome can be reached. The examples should be formatted exactly like the expected input and output, for us this would be JSON. An example for metric *M14* showing two threats that have the same category, asset and threat description is given below:

Same Threat Example

```
[Example]
Threat A:
{
    "Category": "Denial of Service",
    "Asset": "System",
    "Threat": "The System is flooded with requests,
    overwhelming system resources and rendering it unresponsive."
}

Threat B:
{
    "Category": "Denial of Service",
    "Asset": "System"
    "Threat": "An attacker performs a Denial of Service attack on the system."
}

output:
{ "answer": 1 }
```

Prompts

The judges are presented with a prompt containing the threat(s), criteria and a persona, and we define a separate prompt for each metric. The prompts are structured in the same manner for all the metrics, and the persona is the same as well. The persona states that the LLM should act as a "Impartial Judge", which should assist in them being more objective when assessing the threats. The criteria is as described for each metric, i.e. the mitigation metric's criteria would be *"describe the same or overlapping method(s)"*. We also request that the judge disregards spelling and grammar, because it is unlikely that the tool and team writes their text in the exact same way. The final prompts can be found in Appendix C, the examples have not been added, because these should depend on the exact team's own assessment of the metrics, and not ours. The prompts are created with simplicity in mind, by mapping the metrics as a binary decision (0 or 1), except for the risk metric, which can be -1, 0 or 1. This is the suggested method for creating prompts, to limit them to one question, for one/two subjects and then providing a binary decision [36].

Judging Setup

The LLMs should assess each subject in vacuum, like we would expect from a human judge, and not be affected by prior outcomes. Compared to a human judge, we are actually able to programmatically ensure that a vacuum is upheld, by using a stateless LLM inference engine. For a stateless engine, each prompt is handled separately out-of-the-box, and the implementer would have to ask for the history to be upheld.

Furthermore, the LLMs should have a temperature of 0 to produce more consistent answers for the same subject(s), and it should be seeded if possible. Setting the seed and low temperature will enhance the possibility to reproduce the outcomes, however it is not possible to guarantee that the LLM's outcome will remain exactly the same over several iterations [36]. This stems from the inner-workings of the LLMs, leading back to the issue of providing solid explanations for their decision making process as explained in Section 4.3.

Our implementation of LLM-as-a-judge can be found in this GitHub repository: https://github.com/pannaAwesome/ai-threat-model-evaluation, and it uses Ollama[6] as the inference engine. Ollama is chosen, because it is free of charge, and it is not restricted by a max number of requests, because it runs locally. However, running bigger LLMs locally is not feasible, and therefore, the assessments have been run on the AI Labs platform[7] provided by the CLAAUDIA team at Aalborg University.

LLM Selection

The Ollama engine (version 0.6.2) offers a variety of LLM models, and we wish to select five or more LLMs as our judges. The judges are in place of a human judge, and they should aim at assessing the threat models to the same standard. To select the judges, we perform an experiment on a fake baseline and generated threat model, and compare the judges assessments with a human judge. For the inter-rater reliability we cannot use Fleiss' Kappa, as the reliability is measured for two judges at a time, one LLM and one human. Instead, we calculate Cohen's Kappa, which is a similar measurement but the calculations of the actual and chance agreement is slightly different. The complete formula for Cohen's Kappa can be found in Appendix D. The fake threat models can be found in Appendix E.1, the baseline model contains three threats, and the generated five, and they are meticulously curated to provide a positive and negative sample for each of the metrics.

For the experiment, a subset of ten LLMs are chosen based on their performance on a range of LLM knowledge and reasoning benchmarks. The set also includes models from different providers, because an LLM provider can cause a bias in their models, and therefore using more than one from the same provider could skew the inter-rater reliability measure for the judges. The results of the LLM-vs-human experiment are shown in Table 7, and the best performing LLMs are marked in blue. The experiment shows that most of the LLMs are able to achieve a perfect inter-rater reliability with the human judge on at least one of the metrics. phi3:14b is the highest scoring with perfect agreement on four out of seven metrics, with deepseek-r1:14b and qwen2.5:7b seconded with perfect agreement on two out of seven metrics. yi:9b is not able to agree with the human evaluator, and therefore choosing this LLM would defeat the purpose of using LLM-as-a-judge.

[6] https://ollama.com/
[7] https://hpc.aau.dk/ai-lab/

LLM Models	Average Cohen's Kappa						
	Category	Asset	Threats	Mitigations	Same Risk Level	Higher Risk Level	Less Risk Level
command-r7b:7b-12-2024	0.00	0.00	0.75	0.66	0.50	0.35	0.00
deepseek-r1:7b	-0.08	0.60	0.95	0.85	0.68	0.68	0.82
deepseek-r1:14b-qwen-distill	1.00	1.00	0.99	1.00	0.77	0.68	0.69
gemma3:4b	0.80	1.00	0.86	0.80	0.44	0.00	0.46
llama3.1:8b-instruct	0.00	-0.04	0.89	0.81	0.23	0.26	0.55
phi3:3.8b-mini	0.18	0.00	0.26	0.16	0.10	0.08	0.20
phi3:14b-medium	1.00	1.00	1.00	1.00	0.50	0.44	0.50
qwen2.5:7b-instruct	0.48	0.85	1.00	1.00	0.56	0.30	0.78
yi:9b-v1.5	0.07	-0.06	0.04	0.06	0.00	0.00	0.00
yi:9b-chat-v1.5	0.00	1.00	0.86	0.80	0.66	0.00	0.37

Table 7: Human and AI inter-rater reliability observed for possible LLMs

For the experiment, we also calculated the:

- Repetition Stability, *how consistent the assessment is for multiple iterations.*
- Position-bias, *how much the outcome changes, when the order of the subjects is switched.*
- Timings, *how long it takes for the LLM to assess each metric.*

These measurements show other aspects of the LLMs, which are important to consider, but they are not the main motivation for selecting an LLM judge. The measurements are found in Appendix E.4, and they show that the LLMs are repetition stable, producing very reproducible results, and the LLMs experience somewhat of a position bias. According to the results in Table 7, the two deepseek models are both among the five highest scoring, so we look at the other measurements in the appendix to select a LLM from a different provider. Based on the experiment, we choose to use:

- deepseek-r1:14b-qwen-distill
- gemma3:4b
- phi3:14b-medium
- qwen2.5:7b-instruct
- yi:9b-chat-v1.5

7 G1: Threat Modelling Tools

In this section, we evaluate a set of AI-driven threat modelling tools based on the metrics discussed in Section 6.2.1. The tools to evaluate are selected based on a set of defined inclusion criteria, and subsequently evaluated against the questions defined for goal *G1*.

7.1 Tool Selection

The initial list of available tools is based on the related works articles from Section 2, and additional searches with academic and general search engines. Additional searches were conducted using Google and Google Scholar. The search terms are a combination of {"threat modelling tool*", "STRIDE tool*", "LINDDUN tool*", "threat tool*"} and {"AI", "LLM", "machine learning"}, and on Google Scholar, the results are sorted first by *relevance* and then by *date*. The search has been repeated through Autumn 2024, and the last search was performed on the 1st of February 2025. Tools that are created or have adopted AI after this date will therefore not be included in this review. We do not include tools that have been decommissioned as of 1st of February 2025. It should be noted that AI-driven threat modelling tools can also be found on Github, but these tools are deemed too premature for inclusion, and perhaps a later review can focus solely on these tools.

The discovered tools in Table 8 are devised into:

- Frameworks & Modelling Languages, *tools that are not an installable software application, but instead comes as a package or library in another application.*
- Software Applications, *tools that can be installed as an application on your computer, or work as a web application in your browser.*
- AI-integrated Applications, *tools where the AI is not available to the user, but part of the inner mechanisms.*
- AI-assistant Applications, *tools where the AI is a separate AI assistant the user can communicate with through the tool.*

Software Applications			
AutSec	CAIRIS	CoReTM	itemis SECURE
TAM^2	OVVL	OWASP Threat Dragon	SD Elements
SPARTA	Deciduous	Microsoft TMT	TAMELESS
TicTaaC	ThreatGet	Threats Manager Studio	Tutamen Threat Model Automator
Frameworks & Modelling Languages		*AI-integrated Applications*	*AI-assistants Applications*
CORAS	MetaGME	PILLAR	Jeff (IriusRisk)
STS Tool	Threagile	STRIDEgpt	Cyber Sentinel
ThreatSpec	Trike	Aribot	WingMan (ThreatModeler)
OWASP PyTM			ThreatCanvas

Table 8: Overview of threat modelling tools

We develop three inclusion criteria to ensure that we focus on threat modelling tools with AI, but do not limit ourselves to models or assistants. However, threat or security enhanced AI assistants, like Cyber Sentinel[5], will be omitted, as they do not follow or use the processes that we have chosen to measure in our metrics. The criteria also limits the type of AI to LLMs, however we are aware that there potentially could be other AI models in use. The criteria is formalised as:

IC1: The creators of the tool classify it as a "threat modelling tool".

IC2: The tool uses an LLM that can complete one or more of the threat modelling phases described in Section 3.

IC3: The tool has a publicly available documentation, technical paper or similar that describes the LLMs role in the tool.

After applying these criteria, we are left with the six tools described in Table 9, and for the subsequent measurements we use the references listed in the "Ref." column. Out of the six, IriusRisk, ThreatCanvas, Aribot and ThreatModeler are all commercial tools provided by threat or security organisations, whereas PILLAR and STRIDEgpt are by independent creators.

Tool	Description	Ref.
PILLAR	Privacy risk Identification with LINDDUN and LLM Analysis Report (PILLAR) is a privacy only threat modelling tool. It can provide a preliminary threat model with either LINDDUN GO, PRO or MAESTRO, and it works by mimicking the discussion of a threat modelling team.	[27] [3]
STRIDEgpt	The tool is security focused, with limited threat discovery for privacy, and it can create a preliminary threat model with STRIDE. It has an LLM, which automates every phase of the threat modelling process.	[2] [43] [4]
ThreatCanvas	The tool can use a variety of threat and risk approaches, and it is backed by a threat modelling training platform. The tool has a build-in AI assistant that can model the system, and explain parts of the threat model upon the user's request.	[39] [38] [37]
IriusRisk	The tool is a mix of automation and AI-driven STRIDE threat modelling, and it is possible to completely forgo using the AI. The AI can model the system for the user, but for the subsequent phases the tool uses a rule-based engined.	[18]
Aribot	The tool applies to cloud systems, aiming at providing traceability and compliance mapping for cloud security policies. It leverages two LLMs to model and elicit threats for the system.	[6] [7] [45]
ThreatModeler	The tool is similar to IriusRisk, a mix between an automated and AI-driven threat modelling tool. The AI can help model the system, and then a rule-based engine will perform the rest of the phases.	[48] [47]

Table 9: AI-driven threat modelling tools

7.2 Q1: Which functionalities do AI-driven threat modelling tool have?

	approach	$input_{diagram}$	$input_{template}$	$input_{description}$	$input_{questionnaire}$	$input_{asset}$	$input_{code}$	$input_{prompting}$	$modelling_{describe}$	$modelling_{elicit}$	$modelling_{assess}$	$modelling_{mitigate}$	$output_{report}$	$output_{compliance}$	$output_{tests}$	$output_{integration}$
PILLAR	LINDDUN	+	−	+	+	+	−	−	+	+	+	+	+	−	−	−
STRIDEgpt	STRIDE	−	−	+	+	−	+	−	−	*A*	*A*	*A*	−	−	+	−
ThreatCanvas	Risk Templates	−	+	−	−	−	−	+	+	*A*	*A*	*A*	+	+	−	+
IriusRisk	Four-Question	|	|	|	|	−	|	|	+	*R*	+	*R*	+	+	−	+
Aribot	?	+	−	−	−	−	−	−	*A*	+	*U*	+	+	+	−	+
ThreatModeler	?	+	−	−	−	−	−	−	*A*	*R*	+	*R*	+	+	−	+

Table 10: Overview of the metrics of Q1 for the tools

7.2.1 M1: Input Type

Most tools use a mix of textual input and a visualisation, either a diagram and/or template, allowing the users to specify their system in a format that suits their needs. These types of inputs are static, in that they cannot be changed by the user later on, and therefore the user will have a higher chance of fully explaining the system, if they can supply multiple static inputs. Aribot, IriusRisk, and ThreatModeler can only apply one type of input, but IriusRisk has a big variety in input types, allowing the user to choose the most suitable one. Aribot and ThreatModeler can only use an image of a diagram, and the user should be able to model all the important aspects solely through a diagram to use these tools. This means that all assets need to be present in the diagram, and the user will need to find a way of describing any extra information, like *"We assume that the database is fully encrypted"*, in the diagram. In terms of textual input, ThreatCanvas and STRIDEgpt have no support for visual representations, and they mainly rely on a textual description of the system. However, both tools can use multiple inputs, and it should be possible for the user to somewhat describe a visual representation of the system through text.

ThreatCanvas and IriusRisk are the only tools, which let the user interactively provide the input to their AI assistant, thereby the input is no longer static, and the user can specify further detailed upon request. The assistant will try to create a visual representation of the system, which the user should then approve before proceeding with elicitation. This allows the user to ensure the system is modelled correctly by the tool, which will limit the chance of failure.

7.2.2 M2: Threat Approach

Three of the tools apply a generic threat approach, while the other tools each focus on separate aspect: PILLAR for privacy, STRIDEgpt for security, and ThreatCanvas for risk. IriusRisk applies a general approach, but the tool will map the threats to STRIDE, thereby insinuating a stronger focus on security. All the tools follow the same four phases (mentioned in Section 3), so more or less experienced users would not need to adjust to a new threat modelling flow.

PILLAR offers a variety of LINDDUN sub-approaches, in which the user applies either LINDDUN GO, LINDDUN PRO or simply has the tool iterate over the LINDDUN categories. This offers a more

varied experience for the user, where they can fully control the process in PRO, by iteratively going through each component/asset in the system, and select the threats and mitigations they think applies. Whereas, the other options are fully automated.

Similarly, ThreatCanvas can use a range of different risk templates, and even though the approach remains the same, the focus will depend on the user's chosen risk template. Examples of risk templates in ThreatCanvas would be STRIDE, LINDDUN, and OWASP Web Top10. The variations will allow the user to model the same system, with a different focus each time, i.e. the user could create separate threat models for security and privacy.

7.2.3 M3: Modelling Phase

The tools have focused on automating the elicitation and mitigation phases, whereas the input and risk can be more or less user dependent. The automation will be useful for less experienced users, whilst they have the option to adjust the risk scores for elicited threats, if necessary. Except for Aribot, the user can choose to let the AI complete the assessment, letting the threat model purely depend on the AI's judgement.

The automated elicitation and mitigation does not provide any traceability for the decision-making processes of the AI. If the user needs transparency, then IriusRisk and ThreatModeler would be a more suitable choice. They use rule-based automation instead of AI, which connects each threat directly to a rule explaining why it was discovered. IriusRisk have based their rule-engine on Drools[8], which is a business rules management system developed by IBM and Red Hat. ThreatModeler do not disclose how their engine works or what it might be based on. The rule engines are manually updated to keep up with new patterns, components and knowledge bases, which can cause them to be slightly outdated as they are awaiting updates. The same issue can present with the AIs, but it would be possible to automatically update them through prompting. Both methods do require some knowledge base to cover new threats or mitigations, before they are capable of suggesting them.

IriusRisk, PILLAR and ThreatCanvas allow the user to model their system directly in the tool, PILLAR uses a table format and the others have a canvas similar to LucidChart, DrawIO, and Visio. Users that are already familiar with any of these diagramming tools could benefit from using either of these two tools.

7.2.4 M4: Output Format

In this metric there is a clear divide between the commercially operated and the independently created tools. The four commercial tools, Aribot, IriusRisk, ThreatCanvas and ThreatModeler, have a more organisation oriented output. Whereas, PILLAR and STRIDEgpt focus more on providing a quick overview of the threats, which should be further evaluated by a threat modelling team to establish all needed details.

The commercial tools can create risk-, compliance and technical reports, allowing the user to document their threat model based on topic and recipient. The risk and technical reports contain information, which is already accessible through the tool interfaces, and the compliance is created by mapping the threats to known standards, like NIST-800-53[9]. For audits, they provide change management information directly in the tool, to track and explain changes and the current state of the threat

[8] https://www.drools.org/

[9] A catalogue of privacy and security controls an organisation can implement to assist in addressing threats and risks, the full publication is available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

model. For some organisations the audit feature can be necessary as they need the added traceability of their decision making, and for others this may be perceived as unnecessary overhead.

For organisations that wish to mend the gap between threat modelling and development, the commercial tools can integrate with project management tools i.e. Jira and Azure DevOps. This can assist in ensuring continuous assessment and ongoing threat modelling throughout the software development life-cycle. IriusRisk and ThreatModeler can create issues and update them from the tool, whereas ThreatCanvas and Aribot can run directly in the project management tools. This means that developers can develop threat models for new features without leaving the Jira and Azure DevOps environment.

STRIDEgpt can create actual test cases to verify the implementation of mitigations. The tests are written in Gherkin in a *given-when-then* structure, which can be executed with the Cucumber testing framework. Gherkin resembles natural language, and it is not specific to a given programming language, instead the user will need to define *step-definitions* translating the Cucumber test cases into executable code [46]. Compared to the integration of the commercial tools, the tests provide a verifiable method for ensuring the proper implementation of agreed upon mitigations. The tests can also serve as a method for verifying that changes do not reintroduce previously mitigated threats in the system.

Q1 findings

The tools automate all phases of threat modelling, and STRIDEgpt, they allow the user to verify the AI's findings directly in the tool. The phases of the tools do not differ, but three of the tools only concern themselves with a specific aspect of the system, privacy, security or risk.

The commercial tools provide the user with better transparency than the independent ones, and they offer integration with project management platforms. The transparency stems from the ability to either map rule-based or user-driven decision processes during threat elicitation.

The commercial tools seem to be more suitable for bigger organisation needing the added audit features. Whereas the independent tools would be better suited for providing a starting point for further discussion by an experienced team wanting to keep the main process more user-driven.

7.3 Q2: Can AI-driven threat modelling tools be applied in a software project?

	release	$edition_{private}$	$edition_{community}$	$edition_{enterprise}$	*controllability*
PILLAR	11-10-24	+	+	−	+3
STRIDEgpt	21-11-23	+	+	+	+3
ThreatCanvas	06-11-23	−	+	+	0
IriusRisk	(01-12-2021,22-10-2024)	−	+	+	+1
Aribot	30-09-2022	−	−	−	0
ThreatModeler	(26-05-2020,14-11-2023)	−	−	+	0

Table 11: Overview of the metrics of Q2 for the tools

7.3.1 M5: Release

IriusRisk and ThreatModeler have previously been released without AI, and it is possible to use them as pure automatic threat modelling tools. This is also evident in their use of a rule-based elicitation strategy instead of an AI, as this was a part of the non-AI automation tool. These tools are also the oldest, and therefore they might be more mature and have a bigger community supporting them than the others. The other tools are based around AI, potentially resulting in a more integrated workflow.

STRIDEgpt and PILLAR have some future developments planned, but PILLAR no longer appears to be maintained. Their Github [3] states that the tool was purely developed as part of a research project in Autumn 2024, and there is uncertainty regarding the tool's future availability once the research project finishes. The other tools seem to run on a regular release cycle, however further details on their release schedule cannot be found.

7.3.2 M6: Editions

The tools are all available to organisations, either through gratis or/and paid versions. We use the term gratis instead of free, as this only refers to the price of something. The paid versions have more organisational management options than the community version, which works on a per-user basis. Aribot should be available, but it has not been possible to ensure this through their website[10], therefore we do not know which versions it comes in. With the independent tools, the user can quickly get started by accessing them directly through a hosting site, where the user is provided with a short explanation of the workflow on the landing page. Whereas the commercial ones have a bit more overhead, and the user will need to setup an account and payment plan before they can access the tools. For these tools, it is also possible to request a demo before deciding to on which tool to use.

7.3.3 M7: Controllability

As expected the commercial tools are closed source, with IriusRisk exposing parts of their code on Github. For full control of the tool's implementation the organisation would need to use either PILLAR or STRIDEgpt, however they would still need to rely on an AI provider to get the tool running. STRIDEgpt and PILLAR are available for forking, and they are both licensed to allow others to continue development on their own.

IriusRisk has a public organisation on Github[11], which seems to contain an old version of their community tool, along with various APIs and their template library. Through this, a user would be able to somewhat explore the implementation of the tool, but given the age of the last commits, any insights may be outdated and of limited relevance. For the commercial tools, the user would need to rely on the tool providers' implementation and internal processes for mitigating threats in their own systems.

[10] https://aribot.aristiun.com/
[11] https://github.com/iriusrisk

Q2 findings

The tools are available for organisations to use, except for Aribot. The commercial tools have more intricate onboarding processes, whereas the independent tools can be accessed freely without prior requisites.

If an organisation wants full control of their tool's implementation, they would need to use one of the independent tools, however these tools do not provide integration options or auditable transparency as seen in question Q1. The organisations would need to implement these functionalities themselves.

7.4 Q3: Which AI is applied in AI-driven threat modelling tools?

	AI_{google}	AI_{local}	$AI_{mistral}$	AI_{openAI}	min_{ai}	max_{ai}	$prompt_{persona}$	$prompt_{context}$	$prompt_{architecture}$	$prompt_{security}$	$prompt_{privacy}$	$processing$
PILLAR	+	−	+	+	1	3	+	+	+	−	+	−1
STRIDEgpt	|	|	|	|	1	1	+	+	+	+	−	−1
ThreatCanvas	−	−	−	−	1	1	−	−	−	−	−	0
IriusRisk	−	−	−	+	1	1	−	−	−	−	−	+1
Aribot	+	−	−	+	1	2	−	−	−	−	−	−2
ThreatModeler	−	−	−	−	1	1	−	−	−	−	−	0

Table 12: Overview of the metrics of Q3 for the tools

7.4.1 M8: AI Model

Two of the tools implement proprietary AI models, while the remaining tools rely on models from established providers. The independent tools support a variety of distributors, and the user is able to choose any of the models they provide. For these tools, it is possible to obtain gratis access to some models, such as Mistral and locally hosted ones, though most models require payment, adding a cost to otherwise gratis tools.

Aribot uses Aristiun's own AI called Ayurak AI, in their patent they disclose that the AI component combines Google Visual AI and OpenAI models. Visual AI is used to interpret the user's diagram image, and thereafter OpenAI collects the threats and mitigations for each system component based on their threat knowledge base. Google's Visual AI encoder can only take square images of 896x896 pixels, and larger or non-square images will be cropped and resized before encoding. Resizing may cause text to become unreadable and smaller details to be lost. This is especially problematic for architectural diagrams, such as those used in threat modelling [21]. A similar concern can be raised for ThreatModeler, as they also use some type of visual AI to "scan" a diagram image from the user, however we cannot determine the extent, as they have not disclosed the exact AI they use. Neither of the tools allow the user to assess the AI-generated diagram, and any misinterpretation of the diagram image may result in an accurate or incomplete threat model.

For PILLAR and STRIDEgpt, the user can pick any model available to them, and we would expect larger models to output better threat models. The larger models have more parameters, enhancing general performance, and they would have a bigger context window, enabling them to process larger amounts of contextual information simultaneously. After the user has selected an AI, they can adjust the temperature setting, thereby influencing the creativity of the generated threat models.

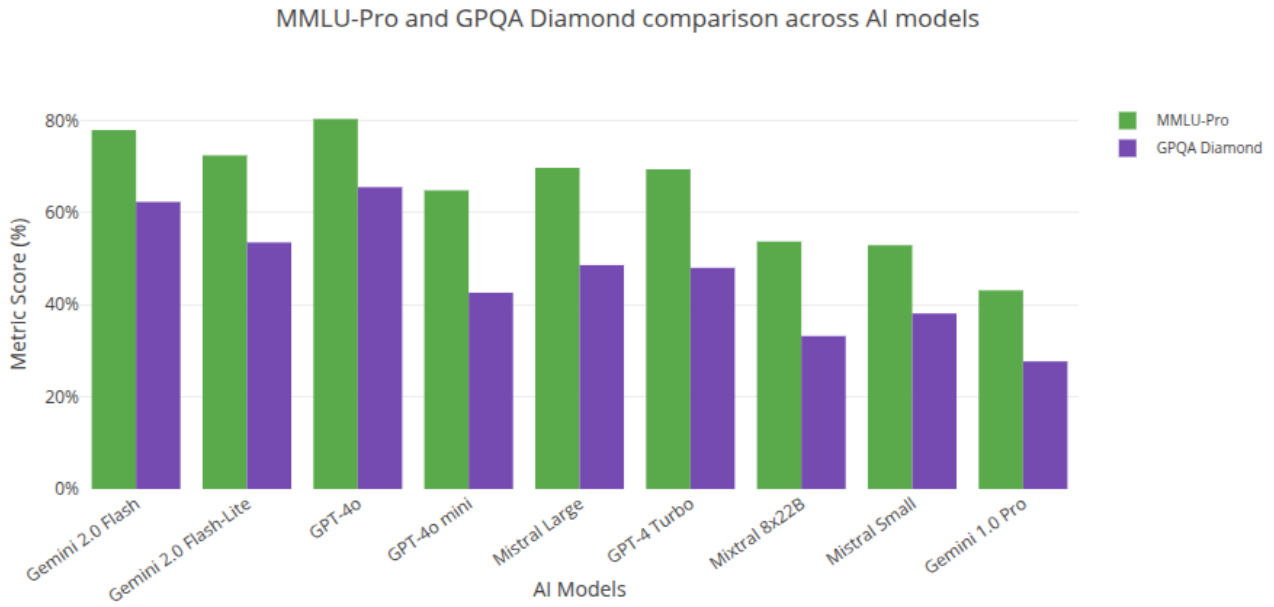


Figure 4: Comparison of AI Model Benchmarks for PILLAR and STRIDEgpt [8]

In terms of knowledge and text generation, Figure 4 displays benchmarks for question answering, and the models are presented in order of context window size. Both the MMLU-Pro[12] and GPQA Diamond[13] benchmarks measure the model's language understanding capabilities on complex, and reasoning-intensive tasks. MMLU-Pro has no baseline, but for GPQA Diamond they measured an accuracy of 65% for human experts, and 34% for non-experts [51, 35]. Compared with human accuracy scores, only GPT-4o and Gemini 2.0 Flash comes close to what human experts can achieve, while the other models lie somewhere in between the experts, and non-experts. These metrics do not say anything specifically towards threat modelling, but we would ponder that models, which are generally better at solving complex tasks, will create better threat models.

7.4.2 M9: Number of AIs

PILLAR uses multiple AIs for LINDDUN GO, in which a discussion forum is created, and each AI is assigned a specific persona. This allows the tool to elicit, assess and mitigate threats from different perspectives, similar to how a human threat modelling team would work. This can provide a more nuanced insight into the threats, offering a more nuanced perspective than tools that simulate only a single persona. Aribot uses two AI models, but they work separately in sequence with very different purposes. These will not be able to discuss their ideas, and thus the tool functions similarly to a system with only one AI.

[12] An extension of the Massive Multitask Language Understanding Benchmark (MMLU) with 12,032 questions spread across 14 different domains.

[13] A data set of 448 multiple choice questions created by domain experts in biology, physics and chemistry.

7.4.3 M10: Prompting

Prompting is used to provide the AIs with perspective, typically specifying a persona, such as "a security engineer with 20 years of experience" or "data architect specialised in privacy and protection of sensitive data". The prompts set the scene for the AI, and they can apply it when eliciting and reasoning about the threats. It is not possible to obtain information about prompts for the commercial tools, though they may not require it, since the AI may already be fine-tuned for threat modelling tasks. Whereas, the independent tools would need further setup as they use generic AI models. The extent of prompting used in PILLAR and STRIDEgpt is not mentioned in any articles, but it was identified thorugh an analysis of their publicly available source code. For PILLAR, the prompting is intended to specify the AI's thought pattern in hopes of them acting like a specific human character. Whereas, for STRIDEgpt the prompting is purely for setting the scene to ensure that the AI actually suggests secure solutions.

7.4.4 M11: Data Processing

The data processing information is not always clear, with Aribot scoring lowest and IriusRisk the highest. The tools describe how they process the user's data, but most leave out how the AI processes the information. Because the AI's processing is not defined, the tools (or AI owners) can freely use the user's information for training purposes. If this is the case, the AI's abilities will be affected by other user's threat models, and the AI could start exhibiting issues similar to the ones experienced with code generation, as discussed in Section 5.3.

Aribot does not disclose any information about how they process the user's information, and the organisation lacks a privacy policy. The user should therefore be wary of the information they share with this tool.

The other commercial tools all have an official privacy policy, but it does not detail how the AI processes the information. Their privacy policies will allow them to use the information as training data in their model. IriusRisk is given a higher score, because they disclose that their AI does not save any information upon leaving the conversation. Furthermore, they use the enterprise version of OpenAI, which has a privacy policy stating that OpenAI is not permitted to use the shared information for training purposes. This is mentioned in their documentation and not privacy policy, so they might not be legally obliged to ensure this is upheld.

PILLAR and STRIDEgpt both disclose that the information in the tool is deleted upon exit, though they caution that the AI's internal data processing mechanisms remain unknown. The tools use publicly available AIs, and it is possible for the providers to save the information for later retraining, and users should keep this in mind when sharing data with the tools. It is possible to obtain data processing information directly from the AI model providers, and they commonly address privacy implications in the model's technical report.

Q3 findings

The commercial tools do not disclose information about the specific AI model or how the model processes the user's information. The user will have to rely on the owners to correctly implement and maintain the AI components and ensure their ongoing viability for threat modelling.

On the other hand, the independent tools allow the user to choose between a set of supported AI models. The user needs to provide an API key, and they are typically required to pay for access. Whilst the tools do not save any of the user's data, the AI providers might, and similarly to the commercial tools, the user will need to trust AI providers to handle their data appropriately and in accordance with relevant policies.

8 G2: Generated Threat Model

From the original six threat modelling tools, a subset will be selected and used to generate threat models for each baseline model presented in Section 6.3.1. We have previously described a set of tool inclusion criteria (TIC) in Section 6.3.2, and applying these to the original six tools, we get:

- IriusRisk
- STRIDEgpt
- ThreatCanvas

Out of these tools, we can see that STRIDEgpt does not actually provide a separate declaration of the asset for a given threat, but the assets are instead mentioned in the threat description. The asset can be identified from the description as the component with the vulnerability to exploit, a.k.a the entry point or target of the attack, and they are written with capital letters. An example from one of the generated threat models: *"An attacker compromises the Message Queue and injects fraudulent messages, which are then processed by the Background Worker as legitimate requests."*, in which the entry point can easily be identified as the "Message Queue" based on the wordings in the descriptions. Therefore, we make a slight change to metrics *M13* and *M14*, allowing them to also consider assets mentioned in the threat descriptions, not just the mapped assets. It cannot be assumed that all descriptions clearly define the entry point, and therefore this could obscure our results.

The steps to generate threat models for each tool is described in Appendix F, and the raw results of the assessments and the generated threat models are available in the "results" folder of the GitHub repository: https://github.com/pannaAwesome/ai-threat-model-evaluation.

8.1 Q4: Can AI-driven threat modelling tools produce valid threat models for a software system and mnemonic threat approach?

The metrics for this question are measured against the generated threat models for each tool. In Table 13, the number of the generated threats for each tool and threat model is seen, these serve as the basis for the results.

Tools	*message queue*	*web app*	*iot edge device*
IriusRisk	57	19	87
STRIDEgpt	17	17	17
ThreatCanvas	33	14	54

Table 13: Number of threats in each generated threat model

8.1.1 M12: Valid Categories

Upon applying the judges on metric *M12*, they reported some invalid categories for IriusRisk, however after reviewing the models, this could not be confirmed. Therefore, we do not include the judges results for this metric, but instead that of a human judge, who did not find any discrepancies. None of the tools suffer from hallucinations related to the threat categories, and the threat models are categorically valid.

8.1.2 M13: Valid Assets

	IriusRisk			STRIDEgpt			ThreatCanvas		
Models	*message queue*	*web app*	*iot edge device*	*message queue*	*web app*	*iot edge device*	*message queue*	*web app*	*iot edge device*
deepseek-r1:14b	10	0	15	0	3	3	0	0	2
gemma3:4b	5	0	3	0	0	0	0	0	0
phi3:14b	8	0	5	0	0	13	0	0	0
qwen2.5:7b	12	0	19	0	3	3	0	0	0
yi:9b-chat	17	0	20	0	7	0	0	0	0

Table 14: Overview of results for valid assets

According to the judges ThreatCanvas does not produce any invalid assets, neither in the directly linked asset nor in the threat description. There is one outlier for IoT Devices with deepseek, which found two invalid assets, however given the remaining judges, this must be an error or misjudgement.

Conversely, IriusRisk has produced many invalid assets for both the message queue and IoT edge device applications, with approximately 10 threats per model containing invalid assets. By investigating the threats closer, it does not seem that the listed asset appointed by IriusRisk is invalid for any of the threats, but the issue is caused by the threat description. An illustrative threat description from the message queue threat model is:

```
Attackers may exploit vulnerabilities in the Secrets Manager API,
such as insufficient input validation or authentication weaknesses,
to gain unauthorized access to secrets.
```

This threat description clearly points at a *"Secrets Manager API"*, which is not part of the original system in the baseline model, likely introduced by IriusRisk, either during AI-based system visualisation or by its rule engine.

A similar trend is seen for STRIDEgpt, however the scope of invalid assets seem to be smaller, and for STRIDEgpt the invalid assets are directly caused by the AI performing the threat elicitation. An example of a threat with an invalid asset in the message queue is:

```
A user performs unauthorized actions and then denies having performed them,
as there is no logging mechanism to track user actions.
```

It is a repudiation threat, and these types of threats typically detail a user or attacker, who cannot be held accountable for their actions, like in the example threat description. In the baseline model, neither of these is a part of the assets, and therefore this threat will seemingly contain invalid assets. From a threat modelling point of view, this should not be classified as an invalid asset, but it actually is caused by poor formulation of assets in the baseline model. Therefore, STRIDEgpt has actually identified some threats for assets that the team otherwise would have missed.

Q4 findings

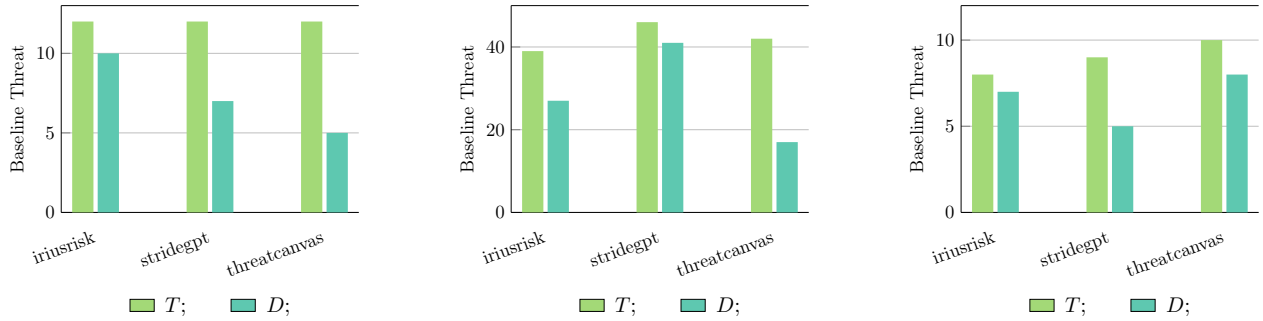
None of the tools create invalid categories, however STRIDEgpt and IriusRisk can create invalid assets for a subset of the threats. For each invalid asset, the threat modelling team will need to judge if they find value in the new asset, and then add it to the system model.

ThreatCanvas has only produced valid threat models, as no invalid categories or assets are present, so a team seeking to avoid validation effort might prefer ThreatCanvas.

8.2 Q5: How sound is the relation between a team's and AI-driven tool's threat model?

For this question, each pair of threats between the generated models and baseline models are assessed by the LLM judges. During the assessment, we chose to save only the baseline threats, excluding the threat pairs, which were defined in the actual metrics. This choice was made, because we only want to focus on the coverage of the baseline model, and not how the threats are mapped across the two models. For each metric, we have averaged over the judges' assessment per baseline and generated threat model, and the full results can be found in Appendix G along with the Fleiss' Kappa score between the judges.

8.2.1 M14: Same Threats



(a) Message Queue Application (b) Web Application (c) IoT Edge Devices

Figure 5: Overview of results for same threats metric for the tools

All the tools have a high coverage of the threats in the baseline models, reaching between 75% and 95% coverage across the applications. At the same time, every generated threat model seems to contain duplicates of the threats in the baseline, adding extra overhead for the team applying the tools. There is no correlation between the number of duplicates, system size and complexity, but it seems that if the generated models contain a larger number of threats than the baseline, there will be a high number of duplicates. IriusRisk produced the biggest models, with more than five times more threats for the message queue application and iot edge devices, which also contain the highest number of duplicates.

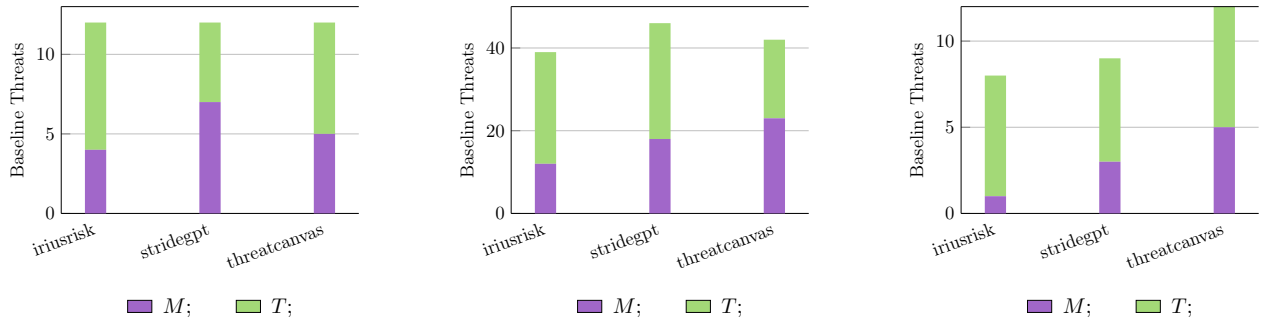
Regardless of the number of threats in the baseline model, STRIDEgpt consistently produces 17 threats. For the web application it seems that it has produced a high number of duplicates, but in the two other applications, only half of the covered threats are duplicated in the generated model. Therefore, the web application could be more challenging, or perhaps the simplicity of the system, has caused duplicates in the 17 generated threats, even though the baseline suggests that there should be many more.

ThreatCanvas scores persistently across all the baseline models, and by looking at the generated threat models, it appears that it could be using a rule-based engine and not AI for eliciting the threats, similar to IriusRisk. An example of a threat from ThreatCanvas would be:

```
Threat Description: The node is susceptible to identity spoofing,
                    where an attacker may impersonate another user or entity.
Mitigation:         Enforce Authorization: Ensure that the node uses strict
                    access policies against unauthorized access.
Risk Score:         High
```

The description, mitigation and risk score are all tied to a specific threat category, and the text is not changed to match any asset in the system, for which the tool has decided that the threat category applies. This is a very stringent approach to threat modelling, but based on the threat results, it seems that it aligns with what a human would be able to find.

8.2.2 M15: Same Mitigations



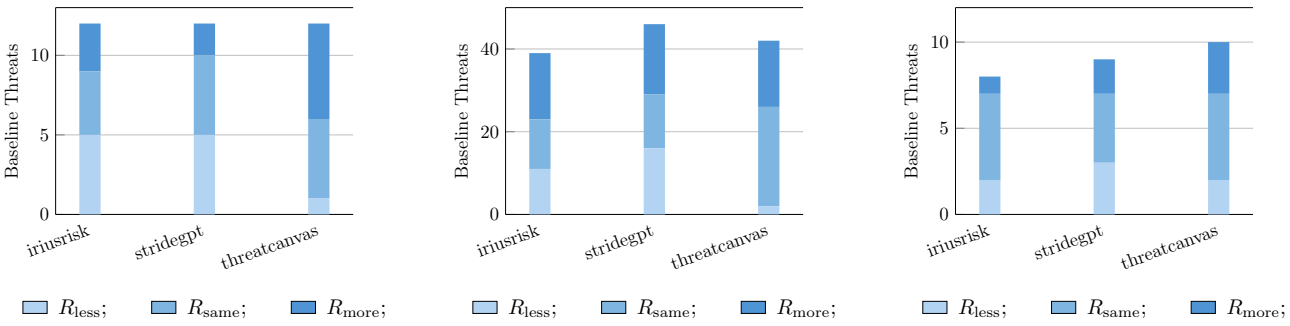
(a) Message Queue Application (b) Web Application (c) IoT Edge Devices

Figure 6: Overview of results for same mitigations metric for the tools

The mitigations are showing a somewhat similar pattern as the threats, however the coverage is much lower ranging between 20% and 55% across the generated models. Again, ThreatCanvas is staying quite consistent across the applications, but it seems that the mitigations have less of an overlap than the threats. This suggests that, although the elicited threats align with those a human would identify, the stringent mitigation suggestions fail to fully reflect those made by human analysts. This could be caused by the mitigations not being tailored to the asset, and thereby missing small subtleties in what can actually be implemented for the asset.

IriusRisk does not capture the mitigations very well, scoring as low as only 2 out of 9 threats for the iot edge devices. Considering that the generated threat model for this system has 87 threats, the score is impeccably low, raising concerns about IriusRisk's ability to appropriately mitigate threats.

8.2.3 M16: Same Risk Score



(a) Message Queue Application (b) Web Application (c) IoT Edge Devices

Figure 7: Overview of results for same risk scores metric for the tools

Before discussing the results, we reiterate the meaning of the three sets: R_{less} when the baseline is lower, R_{same} when they are the same, and R_{more} when the generated is lower. None of the tools manage to correctly prioritise the threats, spanning from higher to lower risk scores compared to the baseline. IriusRisk and STRIDEgpt have a higher ratio of threats in R_{less}, meaning that the tools score the threats higher than the baseline model. On the other hand, ThreatCanvas mostly scores the risks lower than the baseline model, which can again be attributed to its stringent threat modelling

approach. As an example, a lot of the threats in the baseline model are "Information Disclosure" threats, and in the generated model they have a "Low" risk score, but in the baseline models, these are typically at least medium or high depending on the asset. Generally, ThreatCanvas has decided that "Spoofing", "Tampering" and "Elevation of Privilege" are the most sever threats to any system, whilst "Information Disclosure" and "Repudiation" are the lowest.

STRIDEgpt uses DREAD to calculate a numerical risk score, and it seems that this aligns with a human's judgement 1/3 of the time. The rest of the time, it tends to assign higher risk scores, independent of the system's complexity.

Q4 findings

IriusRisk can generate a large number of threats, but somehow it manages to not cover all the threats in the baseline. The mitigations do not align with the baseline, requiring additional effort from the threat modelling team to ensure that the mitigations adhere with their expectations.

STRIDEgpt always creates 17 threats regardless of size and complexity of the system, meaning that its threat model can be rather limited, however the small number of threats seem to cover the baseline well. The risk scores provided by the tool air on the side of caution, and a team would do best in verifying the scores after they have been generated.

ThreatCanvas provides a very stringent one size fits all way for threat modelling, and it does not tailor any of the descriptions, mitigations, or risk scores to the exact system. The generalisation may result in misunderstandings or require extra care to comprehend where and how to implement the mitigations. However, the stringent structure does make the tool capable of achieving a moderate to high match across threats and risk score.

9 Discussion

In this section, we discuss the results of each goal: *G1: Threat Modelling Tools* and *G2: Generated Threat Model*. Thereafter, we discuss the threats to validity, which could impact the results.

9.1 G1: Threat Modelling Tools

The tools do not explain how they circumvent AI limitations; most creators appear to avoid these issue by not using AI for elicitation and mitigation suggestions. Instead, they use a rule engine to propagate the threat model and rely solely on AI to generate a visual representation of the system. The visual representation is directly to user input, and only half of the tools allow users to verify whether the system is accurately understood. Aside from this, the tools generally allow users verify progress, enabling verification at each phase of the threat modelling process to ensure the system is modelled according to the user.

None of the tools using AI for elicitation and mitigation have developed or fine-tuned a pre-trained model. This suggests that users could obtain similar results by using the AI directly from the provider, bypassing the additional layer introduced by the tool. While this would forgo some structural benefits, such as pre-prompting, it would alloe users to tailor outputs to their specific setups. Correlating this with the results from STRIDEgpt in goal *G2* and the study by Sędkowski, it appears feasible to use an AI directly, yielding a reasonably sufficient threat model.

As for the commercial tools, the combination of the closed source and their lack of data processing disclosure means the user has to really trust that the owners have implemented their threat modelling tools properly. Compared to the CrowdStrike incident mentioned in the introduction, vulnerabilities and mismanaged threat modelling tools pose less of a risk. Since a compromised threat modelling tool affects only itself and its stored data, not the user's actual system. The tool's owner should ensure backups are in place, and users should also maintain backups of their threat models. Consequently, users would be less affected by a compromised threat modelling tool than by a system like CrowdStrike's EDR, with the caveat that while both are security-related systems, they serve fundamentally different purposes.

9.2 G2: Generated Threat Model

The review seems to primarily focus on rule-based tools, which appear to partially capture the types of threats a human would identify. IriusRisk has a large rules database and would be expected to correlate closely with human analysis; however, it scores the lowest among the tools By contrast, the stringent approach employed by ThreatCanvas yields the most similar threats. The tool appears to iterate over each system component and assess its vulnerability to each specific threat category, though the decision-making process remains unclear. While the stringent approach suggests the use of a rule-based engine, it is possible that their AI assistant is making the decisions.Based on the STRIDEgpt results, it is plausible that ThreatCanvas uses its AI assistant, thereby supporting the validity of applying AI-driven threat modelling tools.

STRIDEgpt is the only tool confirmed to use AI for threat elicitation, and its creator has limited the number of threats to 17. This may aim to reduce the number of false positives produced by the AI; however, it also requires the team to invest more effort in constructing the complete threat model. IriusRisk tends to generate numerous duplicated or superfluous threats, not due to the AI, but rather its rule engine. This suggests that using AI might yield better results than rule engines, as AI tends to be less rigid in its decision-making, which seems to better suit a subjective activity like threat modelling. However, this may also stem from how IriusRisk visualises the system, using highly specific components in its diagrams, such as "Alibaba Message Queue". Such components may be predisposed to certain threats that are not relevant to the actual system components, leading IriusRisk to over-generate threats. This may also explain why the tool generates threat for assets not present in the actual system.

None of the metrics evaluated the actual descriptions within the threat models, but based on the examples reviewed, this appears to be a promising area for further analysis. IriusRisk provides detailed mitigation guidance, offering a step-by-step implementation process for each suggestion. In contrast, the other two tools present shorter and more concise descriptions, leaving it to the threat modelling team to determine how to implement the mitigations themselves. These are preliminary insights, and more could be made by the addition of more semantically inclined metrics.

9.3 Threats to Validity

The GQM framework is guided by the metrics we defined, and as such, the outcome may be skewed or biased as the metrics can be posed in a way ensuring a specific result. In our implementation, we use generalised metrics supported by the literature, thereby abstracting our own opinions on what tools should achieve and what threat models should contain. For the measurements in *G1*, potential bias may arise from using a single judge, who also created the metrics, and they may therefore recognise subtleties not fully defined in the metric descriptions. In contrast, *G2* is evaluated by separate entities, LLM judges, who lack prior exposure or predispositions regarding the metrics. However, these judges

may introduce their own biases, which we mitigate by employing multiple LLMs to ensure higher result reliability.

Regarding reliability, the Fleiss' Kappa values in Appendix G indicate moderate to perfect agreement among the LLM judges in their assessments. However, these values may be inaccurate, as we only stored the baseline threats and not the full threat pairs. In reality, the baseline threats could be matched with different generated threats, implying that the judges might not fully agree. For the purposes of our review, we are primarily concerned with baseline coverage, making this method valid for our results. However, full threat pairs would provide more accurate input for Kappa calculations.

Some high Kappa values may result from judges matching nearly all threats in the baseline. Not all judges show equal willingness to match threats. However, data trends suggest that the similarity criteria between threats may be too vaguely defined, resulting in excessive matches. Examining specific responses reveals that some judges contradict themselves and forget relevant categories and assets, leading to overly permissive threat matches. This issue may stem from model size or limited context window length, the latter potentially causing loss of earlier prompt information, such as criteria, categories, and assets. Using larger models or shorter prompts may mitigate the issue; alternatively, applying chain-of-thought prompting could improve the LLM's response awareness. Other enhancements would entail fine-tuning of the judges on a labelled threat model, and further simplification of the prompts. We have not explored how the LLMs measure against a human judge, but in future this would allow us to better estimate the correctness of the LLMs' assessments.

The tools in *G2* are tested against several baseline models. To support the viability of the results, we applied them to systems of varying size and complexity. This reduces the risk of a tool performing well on only one type of system. However, the small number of test cases suggests that additional baseline models should be used for stronger generalisation. The chosen baselines are not necessarily high-quality threat models, which may influence the results, and using different baselines could yield different outcomes. Regarding soundness, the correctness of the baseline models is not accounted for, as defining a definitive threat model is inherently difficult. To reduce the impact of poor baselines, a maximal threat set could be employed. This maximal threat set could be derived from knowledge bases such as CVE, ATT&CK, and CAPEC, ensuring that uncovered threats in the baseline are still relevant to the system.

10 Conclusion

This project introduced two sets of metrics: one for analysing AI-driven threat modelling tools, and another for evaluating the soundness and validity of generated threat models. These metrics were applied to a subset of AI-driven threat modelling tools to highlight their features and assess the value they could offer to a threat modelling team. Value was measured through a case study comparing example threat models with those generated by the tools, which were subsequently evaluated by LLMs.

Six AI-driven threat modelling tools were reviewed, and all the tools can operate autonomously and are easy to adopt for novice threat modellers. Half of the tools only use AI for creating system diagrams, and they still rely on rule-based engines to elicit threats. Tool owners struggle to enforce private data handling, so users must remain aware of what information they share with the tools.

Only three tools were used in the case study, and they were able to generate reasonably sound threat models that matched threats in the example models. However, the tools were less successful at suggesting similar mitigations, and their risk scores often deviated, being either consistently lower or higher than those in the example models. The findings also suggest that applying AI for threat elicitation instead of rule-based engines, yield threat models that are more similar to those of a threat modelling team.

The developed metrics highlighted distinct aspects of each tool and may prove useful for other researchers investigating AI-driven threat modelling tools. Using LLMs as judges enabled fast, automated assessments; however, accuracy could have been improved by using chain-of-thought, fine-tuning or more simplistic prompts.

Future work should include repeating the case study with a larger set of example threat models and AI-driven threat modelling tools to strengthen our preliminary findings. Whilst for the approach, future extensions of the metrics could involve semantic evaluation of both the technical depth and the elaborative nature of descriptions in generated threat models.

References

[1] Josh Achiam, Steven Adler, and et al. Gpt-4 technical report, 2024.

[2] Matthew Adams. Ai-driven threat modeling with stride gpt, 12 2024.

[3] Matthew Adams. Pillar - github repository, 01 2025. Commit: c10f85b890756f2a8b31662c103b5500d8c8edc7.

[4] Matthew Adams. stride-gpt - github repository, 03 2025. Commit: c7dc932f918d3b3d4bfa3f13b6bce8d3d9565d1d.

[5] Zarif Bin Akhtar and Ahmed Tajbiul Rawol. Enhancing cybersecurity through ai-powered security mechanisms. *IT Journal Research and Development*, 9(1):50–67, 2024.

[6] Aristiun. Automated threat modeling using ai: The latest addition to our security suite, 2024. Accessed on: 30-01-2025.

[7] Aristiun. White paper - unleashing the power of automated threat modelling with aribot, 2024. Accessed on: 30-01-2025.

[8] Artificial Analysis. Ai model comparisons - artificial analysis, 2025. Accessed: 2025-04-28.

[9] Priyant Banerjee. Crowdstrike cyber incident vs. past major cyber incidents: Analysis and solutions. *International Journal For Multidisciplinary Research*, 6(4), 2024.

[10] Harold Booth, Doug Rike, and Gregory Witte. The national vulnerability database (nvd): Overview, 12 2013. Accessed on 17-03-2025.

[11] DATAtab. Cohen's kappa: Measuring inter-rater agreement, 2025. Accessed: 2025-05-19.

[12] Isra Elsharef, Zhen Zeng, and Zhongshu Gu. Facilitating threat modeling by leveraging large language models. In *Workshop on AI Systems with Confidential Computing*, 2024.

[13] Jack Freund and Jack Jones. *Measuring and managing information risk: a FAIR approach*, chapter 2. Butterworth-Heinemann, 2014.

[14] Daniele Granata and Massimiliano Rak. Systematic analysis of automated threat modelling techniques: Comparison of open-source tools. *Software Quality Journal*, 32:1–37, 05 2023.

[15] Aaron Grattafiori, Abhimanyu Dubey, and et al. The llama 3 herd of models, 2024.

[16] Amina Hajrić, Tarik Smaka, Sabina Baraković, and Jasmina Baraković Husić. Methods, methodologies, and tools for threat modeling with case study. *Telfor Journal*, 12(1):56–61, 2020.

[17] Weiche Hsieh, Ziqian Bi, and et al. A comprehensive guide to explainable ai: From classical models to llms, 2024.

[18] IriusRisk. *IriusRisk Documentation*, 01 2025. Accessed on: 30-01-2025.

[19] Mathias Jesussek. Fleiss' kappa: Measuring agreement among multiple raters, 2025. Accessed: 2025-05-19.

[20] Mehrdad Kaheh, Danial Khosh Kholgh, and Panos Kostakos. Cyber sentinel: Exploring conversational agents in streamlining security tasks with gpt-4. *arXiv preprint arXiv:2309.16422*, 2023.

[21] Aishwarya Kamath, Johan Ferret, and et al. Gemma 3 technical report, 2025.

[22] Uday Kamath, Kevin Keenan, Garrett Somers, and Sarah Sorenson. Llm challenges and solutions. In *Large Language Models: A Deep Dive: Bridging Theory and Practice*, pages 219–274. Springer, 2024.

[23] Meryem Kassou and Laila Kjiri. A goal question metric approach for evaluating security in a service oriented architecture context. *arXiv preprint arXiv:1304.0589*, 2013.

[24] Jan H. Klemmer, Stefan Albert Horstmann, Nikhil Patnaik, Cordelia Ludden, Cordell Burton Jr, et al. Using ai assistants in software development: A qualitative study on security practices and concerns. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2726–2740, 2024.

[25] Heiko Koziolek. Goal, question, metric. *Dependability Metrics: Advanced Lectures*, pages 39–42, 2008.

[26] Vahid Majdinasab, Michael Joshua Bishop, Shawn Rasheed, Arghavan Moradidakhel, Amjed Tahir, and et al. Assessing the security of github copilot's generated code-a targeted replication study. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 435–444. IEEE, 2024.

[27] Majid Mollaeefar, Andrea Bissoli, and Silvio Ranise. Pillar: an ai-powered privacy threat modeling tool, 2024.

[28] Nitin Naik, Paul Jenkins, Paul Grace, Dishita Naik, Shaligram Prajapat, and et al. An introduction to threat modelling: modelling steps, model types, benefits and challenges. In *The International Conference on Computing, Communication, Cybersecurity & AI*, pages 260–270. Springer, 2024.

[29] Livinus Obiora Nweke and Stephen Wolthusen. A review of asset-centric threat modelling approaches. *International Journal of Advanced Computer Science and Applications*, 11(2), 2020.

[30] Onyinye Agatha Obioha-Val, Temitope Ibrahim Lawal, Oluwaseun Oladeji Olaniyi, Michael Olayinka Gbadebo, and Anthony Obulor Olisa. Investigating the feasibility and risks of leveraging artificial intelligence and open source intelligence to manage predictive cyber threat models. *Journal of Engineering Research and Reports*, 27(2):10–28, 2025.

[31] Stack Overflow. 2024 developer survey - stack overflow. https://survey.stackoverflow.co/2024, 05 2024. Accessed on 08-03-2025.

[32] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. *Communications of the ACM*, 68(2):96–105, 2025.

[33] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. Do users write more insecure code with ai assistants? *arXiv preprint arXiv:2211.03622*, 2022.

[34] Nitin Rane, Saurabh Choudhary, and Jayesh Rane. Gemini versus chatgpt: applications, performance, architecture, capabilities, and implementation. *Journal of Applied Artificial Intelligence*, 5(1):69–93, 03 2024.

[35] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

[36] Elena Samuylova. Llm-as-a-judge: A complete guide to using llms for evaluations, February 2025. Accessed on 21-04-2025.

[37] SecureFlag. Automated threat modeling with threatcanvas, 03 2024. Accessed on: 28-01-2025.

[38] SecureFlag. Introducing threatcanvas 2.0: Revolutionizing threat modeling, 11 2024. Accessed on: 28-01-2025.

[39] SecureFlag. Secureflag introduces threatcanvas: an ai-powered tool to automate threat modeling, 11 2024. Accessed on: 28-01-2025.

[40] Nataliya Shevchenko, Timothy A. Chick, Paige O'Riordan, Thomas P. Scanlon, and Carol Woody. Threat modeling: a summary of available methods. *Software Engineering Institute/ Carnegie Mellon University*, pages 1–24, 2018.

[41] Zhenpeng Shi, Kalman Graffi, David Starobinski, and Nikolay Matyunin. Threat modeling tools: A taxonomy. *IEEE Security and Privacy*, 20(4):29–39, 2022.

[42] Aviral Srivastava and Priyansh Sanghavi. Zero-shot learning in cybersecurity: A paradigm shift in attack and defense. In *Advances in Computing and Data Sciences: 8th International Conference, ICACDS 2024, Vélizy, France, May 9–10, 2024, Revised Selected Papers*, volume 2194, page 138. Springer Nature, 2025.

[43] Wiktor Sędkowski. Threat identification using stride and gpt based chatbots. *Studia Społeczne*, 46(3):75–86, 2024.

[44] Kristen Tan and Vaibhav Garg. An analysis of open-source automated threat modeling tools and their extensibility from security into privacy. *usenix - The advanced computing systems association*, 2022.

[45] Tejvir. *A SYSTEM AND METHOD FOR ARTIFICIAL INTELLIGENCE BASED THREAT MODELING*, 10 2024. Pub No.: EP 4 451 152 A1.

[46] The Cucumber Open Source Project. *Cucumber Documentation - Introduction*, 12 2024. Accessed On: 21-04-2025.

[47] ThreatModeler. *What's New in the ThreatModeler 7.0 Platform*, 11 2023. Accessed On: 30-01-2025.

[48] ThreatModeler. Stay up-to-date with threatmodeler, 2025. Accessed On: 30-01-2025.

[49] Dimitri Van Landuyt, Laurens Sion, Walewein Philips, and Wouter Joosen. From automation to ci/cd: a comparative evaluation of threat modeling tools. In *2024 IEEE Secure Development Conference (SecDev)*, pages 35–45, 2024.

[50] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H. Dieter Rombach. Goal question metric (gqm) approach. *Encyclopedia of software engineering*, 2002.

[51] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, Wenhu Chen, and et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024.

[52] Kim Wuyts, Laurens Sion, Dimitri Van Landuyt, and Wouter Joosen. Knowledge is power: Systematic reuse of privacy knowledge for threat elicitation. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 80–83, 2019.

[53] Wenjun Xiong and Robert Lagerström. Threat modeling – a systematic literature review. *Computers and Security*, 84:53–69, 2019.

[54] Shuiqiao Yang, Tingmin Wu, Shigang Liu, David Nguyen, Seung Jang, and et al. Threatmodeling-llm: Automating threat modeling using large language models for banking system. *arXiv preprint arXiv:2411.17058*, 2024.

[55] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and et al. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211, 2024.

[56] Asad Yaseen. Ai-driven threat detection and response: A paradigm shift in cybersecurity. *International Journal of Information and Cybersecurity*, 7(12):25–43, 2023.

[57] Wojciech Zaremba, Greg Brockman, and et al. Openai codex. https://openai.com/index/openai-codex/, 08 2021. Accessed on 24-03-2025.

[58] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, and et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

A Data Flow Diagrams for threat modelled systems

A.1 Message Queue Application

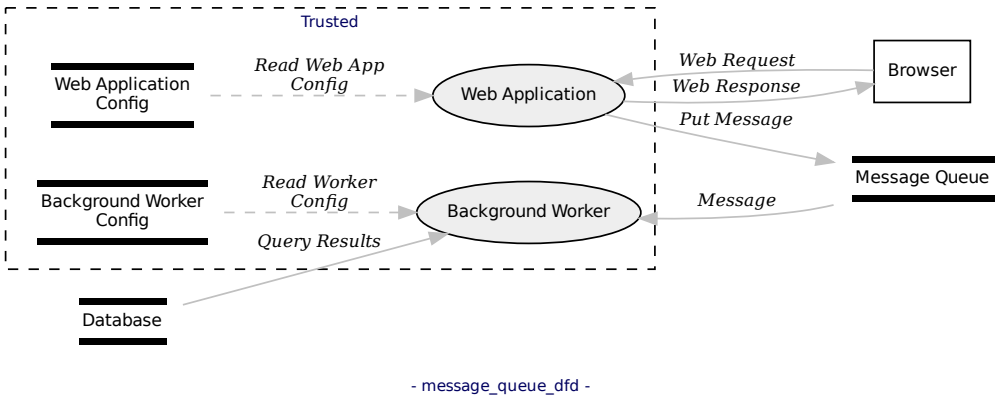


Figure 8: DFD for Message Queue Application system

A.2 Web Application

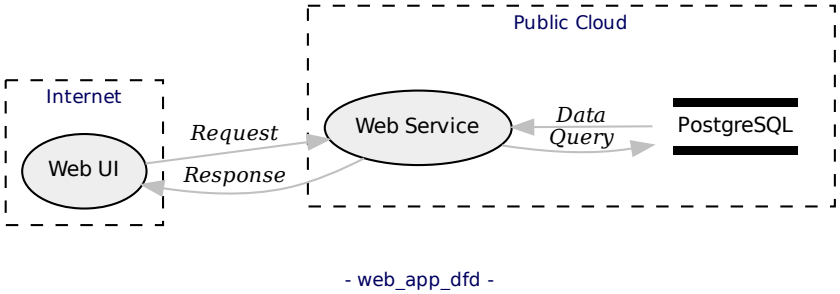


Figure 9: DFD for Web Application system

A.3 IoT Edge Devices

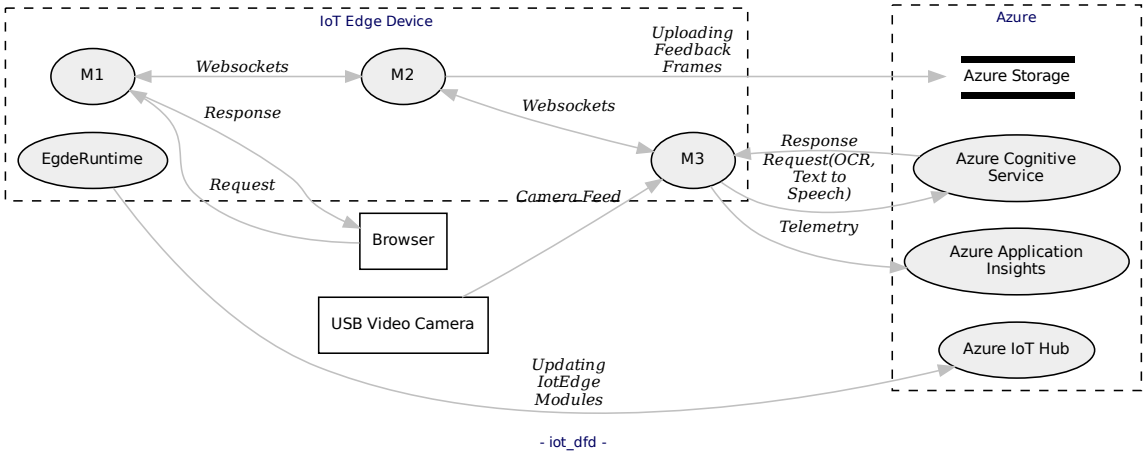


Figure 10: DFD for IoT Edge Device system

B Reformatted Input for Baseline Models

B.1 Message Queue Application

```
An application with a queue-decoupled background process.
The application has one external entity the Browser.
The application has two trusted processes the Web Application and
Background Worker.
The application has two trusted data stores the Web Application
Config and the Background Worker Config, and two untrusted data
stores the Message Queue and the Database.

The Browser requests and sends messages to the Web Application.
The Web Application puts the message in the Message Queue.
The Web Application Config stores credentials used by the Web
Application to access the Message Queue.

The messages in the Message Queue is send to the Background Worker.
The Background Worker can query the Database.
The Background Worker Config stores the credentials used by
the Background Worker to access the Database.
```

B.2 Web Application

```
The web application is build in three tiers: Web UI, Web Service and PostgreSQL.

PostgreSQL is in the trust zone Public Cloud and it is a data store that holds
```

customer data. It is PostgreSQL database deployed on public cloud with Amazon Web Services, specifically Relational Database Service (RDS). The data is encrypted during transfer with HTTPS and SSL/TLS, and the encryption is applied at transport level so covering the whole connection.

Web Service is in the trust zone Public Cloud and it processes customer data, which can contain personally identifiable information. This information is both send and received for the process. The Web Service uses a Java Web Container, running on Amazon Web Services specifically Elastic Compute Cloud (EC2). The Web Service implements an authentication function with a username and password login structure. The Web Service uses session management and it gives a unique session ID, which will be transmitted between the client and server. The data between the Web Service and the client is encrypted with HTTPS and SSL/TLS at the transport level. The Web Service can take XML as input from the client.

Web UI is in the internet trust zone and it sends and receives customer data, which can contain personally identifiable information. The Web UI will request the user for a password based login before they can use the application. The data between the Web UI and the client is encrypted with HTTPS and SSL/TLS at the transport level. The Web UI is written with JQuery.

The system has two processes: Web UI and Web Service, and one data store: PostgreSQL. Web UI is in the Internet trust zone, and the Web Service and PostgreSQL is in the Public Cloud trust zone.
The Web UI exchanges requests and responses with the Web Service.
The Web Service exchanges queries and data with the PostgreSQL.

We have allowed that the PostgreSQL stores sensitive data, and this should not be factored into the threat model.

B.3 IoT Edge Devices

A system which takes video frames from video camera and process these frames on IoTEdge devices and send them to Azure Cognitive Service to get the audio output.
The system consists of the following assets:
- Azure Blob Storage which has a HTTP entry Point and uses a connection string
- Azure Monitor which has a HTTP end point and uses a connection string
- Azure Cognitive Service which has a HTTP end point and uses a connection string
- M1 an IoTEdge Module with an HTTP end point and uses public access LAN
- M2 an IoTEdge Module with an HTTP end point and uses public access LAN
- M3 an IoTEdge Module with an HTTP end point and uses public access LAN
- IoTEdgeMetricsCollector an IoTEdge Module with a HTTP end point and uses public access LAN
- Application Insights which has an HTPP end point and uses a connection string

The client's Browser makes requests to M1. The Browser and M1 device are on the same network, so Browser directly hits the webapp URL.
M1 interacts with other two IoTEdge modules M2 and M3 to render live stream from video device and display order scanning results via WebSockets.

The IoTEdge Modules interact with Azure Cognitive service to get the translated text via OCR and audio stream via Text to Speech Service.
IoTEdge modules send telemetry information to Application Insights.
IoTEdge device is deployed with IoTEdge runtime which interacts with IoTEdge hub for deployments.
IoTEdge module also sends some data to Azure storage which is required for debugging purpose.
Cognitive service, Application Insights and Azure Storage are authenticated using connection strings which arestored in GutHub secrets and deployed using CI/CD pipelines.

The system has one IoT Edge Device that contains four processes: M1, M2, M3 and EdgeRuntime.
The system has one Azure instance with three processes: Azure IoT Hub, Azure Cognitive Service and Azure Application Insights, and one data store: Azure Storage.
The IoT Edge Device and Azure each represent a separate trust boundary.
The system has two external entities: Browser and USB Video Camera

M1 to M2 and M2 to M3 are connected with WebSockets.
The IoT Edge Device sends telemetry information to the Azure Application Insights process
M1 exchanges requests and responses with the Browser.
M2 uploads feedback frames to the Azure Storage.
M3 receives the camera feed from the USB Video Camera.
M3 exchanges requests and responses with the Azure Cognitive Service, and the request contains OCR and Text-to-Speech information.
EdgeRuntime sends IoTEdge Module updates to Azure IoT Hub.

We assume that secrets like ACR credentials are stored in GitHub secrets store which are deployed to IoTEdge Device by Ci/CD pipelines. However CI/CD pipelines are considered

C Prompts for LLM judges

C.1 Prompt for metric M12: Valid Category

CATEGORY PROMPT

[Task Description]
You are an impartial judge evaluating whether this threat has a valid category.
The valid categories are:
- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege
You should disregard big and small letters.

[Threat]
{{tm}}

[Answer Options]
Valid: { "answer": 0 }
Invalid: { "answer": 1 }

C.2 Prompt for metric M13: Valid Asset

ASSET PROMPT

[Task Description]
You are an impartial judge evaluating whether this threat only addresses allowed assets.
The valid assets are: {{assets}}
A threat only addresses allowed assets if:
- The ‘Asset‘ field if present should be a valid asset
- If the ‘Threat‘ defines the target or entry point as a valid asset, typically these are capitalized
- If the ‘Mitigation‘ suggest to apply changes to a valid asset, typically these are capitalized

[Threat]
tm

[Answer Options]
Valid: { "answer": 0 }
Invalid: { "answer": 1 }

C.3 Prompt for metric M14: Same Threats

THREAT PROMPT

```
[Task Description]
You are an impartial judge evaluating whether the following two threats
describe the same issue.

A Threat A and B are similar if:
- The ‘Category‘ is the same.
- If both threats specify an ‘Asset‘, they must match.
  If not, infer the asset from the ‘Threat‘ field, which is the target
  or entry point, and capitalised.
- The ‘Threat‘ descriptions must have the same malicious objective and impact,
  either stated or implied.
- Ignore differences in style, grammar, or formatting.

[Examples Begin]
...\\
[Examples End]

[Threat A]
{{tm1}}

[Threat B]
{{tm2}}

[Answer Options]
Similar: { "answer": 1 }
Not similar: { "answer": 0 }
```

C.4 Prompt for metric M15: Same Mitigations

MITIGATION PROMPT

[Task Description]
You are an impartial judge evaluating whether the following two mitigations suggest the same underlying method.
A Mitigation A and B are the similar if:
- The ‘Mitigation‘ field describe the same or overlapping method(s).
- A partial overlap is acceptable — not all techniques need to be listed in both.
- Ignore differences in wording, formatting, or technical phrasing.

[Examples Begin]
...
[Examples End]

[Mitigation A]
{{tm1}}

[Mitigation B]
{{tm2}}

[Answer Options]
Similar: { "answer": 1 }
Not similar: { "answer": 0 }

C.5 Prompt for metric M16: Same Risk Score

RISK PROMPT

[Task Description]
You are an impartial judge evaluating the severity of risks in two elements in a tuple.
Risk A and B are the same if:
- The risk presents the same level of severity, regardless of the format.
- Ignore any difference in style, grammar, or punctuation.

[Examples Begin]
...
[Examples End]

[Risk A]
{{tm1}}

[Risk B]
{{tm2}}

[Answer Options]
Similar: { "answer": 0 }
A bigger than B: { "answer": -1 }
B bigger than A: { "answer": 1 }

D Cohen's Kappa Formula

In Cohen's Kappa, the observed agreement p_o is the sum of all subjects, where two judges agreed, and the expected p_e is the sum of judges assigning each outcome, both are then normalised against the total number of assessments. The formula for calculating the expected and observed agreement for Cohen's Kappa is [11]:

Let i be the evaluation outcomes, and n the number of total assessments
Then x_{ij} is the number of times judge j assigned outcome i
and x_i is the number of times both judges assigned outcome i

$$p_o = \frac{1}{n} \sum_i x_i \qquad\qquad p_e = \frac{1}{n^2} \sum_i x_{i1} * x_{i2}$$

The Kappa value is then calculated with the formula:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

E Fake Threat Model Experiment

E.1 Fake aseline threat model

[

```
    {
        "ID": 0,
        "Category": "Elevation of Privilege",
        "Asset": "Frontend",
        "Threat": "A malicious user performs an SQL injection
                  and they gain unauthorized access to the system.",
        "Mitigation": "Sanitize user input before applying it anywhere
                       in the system, and use an allowlist if possible.",
        "Risk": "Low Severity"
    },
    {
        "ID": 1,
        "Category": "Elevation of Privilege",
        "Asset": "Frontend",
        "Threat": "A malicious user performs an SQL injection and
                  they gain unauthorized access to the system.",
        "Mitigation": "Sanitize user input before applying it anywhere
                      in the system, and use an allowlist if possible.",
        "Risk": "Medium Severity"
    },
    {
        "ID": 2,
        "Category": "Elevation of Privilege",
        "Asset": "Frontend",
        "Threat": "A malicious user performs an SQL injection and
                  they gain unauthorized access to the system.",
        "Mitigation": "Sanitize user input before applying it anywhere
                      in the system, and use an allowlist if possible.",
        "Risk": "High Severity"
    }
]
```


E.2 Fake generated threat model


```
[
    {
        "ID": 0,
        "Category": "Elevation of Privilege",
        "Asset": "Frontend",
        "Threat": "Attackers may exploit improperly implemented input validation
                  mechanisms on the Frontend, and thereby gain unauthorized access
                  to the Web Server.",
        "Mitigation": "Implement strict input validation on all user-supplied data
                      to prevent injection attacks like SQL and XSS.",
        "Risk": "2 out of 3"
    },
    {
        "ID": 1,
        "Category": "Elevation of Privilege",
```

```
        "Asset": "Frontend",
        "Threat": "Attackers may exploit improperly implemented input validation
                  mechanisms on the Frontend, and thereby gain unauthorized access
                  to the Web Server.",
        "Mitigation": "Limit the amount of requests that the user can make at once.",
        "Risk": "50 out of 100"
    },
    {
        "ID": 2,
        "Category": "Elevation of Privilege",
        "Threat": "Attackers may exploit improperly implemented input validation
                  mechanisms on the Frontend, and thereby gain unauthorized access
                  to the Web Server.",
        "Mitigation": "Implement strict input validation on all user-supplied data
                      to prevent injection attacks like SQL and XSS.",
        "Risk": "50 out of 100"
    },
    {
        "ID": 3,
        "Category": "Non-Repudiation",
        "Asset": "Frontend",
        "Threat": "Attackers may exploit improperly implemented input validation
                  mechanisms on the Frontend, and thereby gain unauthorized access
                  to the Web Server.",
        "Mitigation": "Implement strict input validation on all user-supplied data
                      to prevent injection attacks like SQL and XSS.",
        "Risk": "50 out of 100"
    },
    {
        "ID": 4,
        "Category": "Elevation of Privilege",
        "Asset": "Database",
        "Threat": "Attackers may exploit improperly implemented input validation
                  mechanisms on the Frontend, and thereby gain unauthorized access
                  to the Web Server.",
        "Mitigation": "Implement strict input validation on all user-supplied data
                      to prevent injection attacks like SQL and XSS.",
        "Risk": "50 out of 100"
    }
]
```


E.3 Human evaluation of fake threat model

Result M12: Valid Category
$C = \{3\}$

Result M13: Valid Asset
$A = \{4\}$

Result M14: Same Threats
$T = [0, 0, 0, 1, 1, 1, 2, 2, 2]$

Result M15: Same Mitigations
$M = [0, 0, 1, 1, 2, 2]$

Result M16: Same Risk Score
$R_{same} = [1, 1, 1]$
$R_{more} = [0, 0, 0]$
$R_{less} = [2, 2, 2]$

E.4 Experiment Results

Positional Bias

	Average Cohen's Kappa				
LLM Models	Threats	Mitigations	Same Risk Level	Higher Risk Level	Less Risk Level
command-r7b:7b-12-2024	1.00	1.00	0.00	0.00	0.00
deepseek-r1:7b	0.78	0.86	0.00	0.92	0.75
deepseek-r1:14b-qwen-distill	0.94	0.97	0.00	0.83	0.83
gemma3:4b	0.55	0.62	0.00	0.00	0.00
llama3.1:8b-instruct	0.89	0.92	0.00	0.87	0.57
phi3:3.8b-mini	0.58	0.00	0.00	-0.02	0.00
phi3:14b-medium	0.00	0.00	0.00	0.00	0.00
qwen2.5:7b-instruct	0.64	0.58	0.00	-0.02	0.56
yi:9b-v1.5	0.00	0.00	1.00	1.00	1.00
yi:9b-chat-v1.5	1.00	1.00	0.00	0.23	0.13

Table 15: Position Bias observed for possible LLMs when the order of threats is switched

Repetition Stability

LLM Models	Fleiss' Kappa				
	Threats	Mitigations	Same Risk Level	Higher Risk Level	Less Risk Level
command-r7b:7b-12-2024	1.00	1.00	0.74	1.00	1.00
deepseek-r1:7b	1.00	1.00	0.68	1.00	0.79
deepseek-r1:14b-qwen-distill	1.00	1.00	0.82	0.87	0.76
gemma3:4b	1.00	1.00	1.00	1.00	0.68
llama3.1:8b-instruct	1.00	1.00	0.13	0.54	0.75
phi3:3.8b-mini	0.45	0.13	-0.01	-0.03	-0.23
phi3:14b-medium	1.00	1.00	1.00	0.79	1.00
qwen2.5:7b-instruct	1.00	1.00	0.89	0.48	0.85
yi:9b-v1.5	-0.01	-0.01	1.00	1.00	1.00
yi:9b-chat-v1.5	1.00	1.00	1.00	0.79	0.85

Table 16: Repetition Stability observed for possible LLMs for five iterations

Timing

LLM Models	Time Spend			
	Total Time	Hallucinations Time	Threats Time	Request Time
command-r7b:7b-12-2024	6 min 40 sek	6 sek	36 sek	2 sek
deepseek-r1:7b	1 h 18 min 59 sek	54 sek	7 min 24 sek	30 sek
deepseek-r1:14b-qwen-distill	2 h 51 min 28 sek	2 min 19 sek	15 min 59 sek	1 min 4 sek
gemma3:4b	4 min 16 sek	4 sek	23 sek	1 sek
llama3.1:8b-instruct	18 min 24 sek	42 sek	1 min 30 sek	6 sek
phi3:3.8b-mini	14 min 30 sek	17 sek	78 sek	5 sek
phi3:14b-medium	11 min 34 sek	26 sek	56 sek	4 sek
qwen2.5:7b-instruct	7 min 34 sek	20 sek	35 sek	2 sek
yi:9b-v1.5	5 h 21 min 36 sek	5 min 56 sek	29 min 11 sek	1 min 57 sek
yi:9b-chat-v1.5	2 h 16 min 20 sek	4 min 8 sek	11 min 34 sek	46 sek

Table 17: Total & average time spend by the judges

F Application of Tools

We describe the procedure for generating the threat models, and how the results are extracted into a list of JSON objects:

```
{
    "Category": "",
    "Asset": "",
    "Threat": "",
    "Mitigation": "",
    "Risk": ""
}
```

F.1 IriusRisk

In IriusRisk, before starting a project Jeff (their AI) is selected, as otherwise the tool will run without the AI. This will open a new window with a input field, in which we paste the textual description of the system. Hereafter, Jeff will ask questions to ensure that it has understood the system correctly, and it will show the user a diagram of the system. Throughout, this prompting we only answered "yes", to not affect the AI's process.

Then, the threat model project is created, and IriusRisk's rule-engine runs in the background to elicit the threats and mitigations. Once, it has completed, we navigated to the "Countermeasure and Threats" tab, to view the threat model. The threats and mitigations were extracted into a csv format using the report function, the threats are from the "Technical Report" and the mitigations from the "Countermeasure Report". The extracted csv files are then processed into the JSON format, which aligns with the other generated threat models.

During the processing we extract the Use Case, Component, and Current Risk from the threats file, and the Description from the mitigations. If more than one mitigation applies to the threat and asset, we combine them into one long string. There is no other reformatting or processing performed on the extracted information.

F.2 STRIDEgpt

In STRIDEgpt, the defined system description is inserted in the input box, and the application is set to match that of the system. If the baseline model contains information about the login type, we also supply this information using the drop-down for authentication. Then, the threats are elicited using the button at the bottom of the page, and downloaded as a markup file. For the mitigations and risk score, we navigate to the respective tab, and use the button to generate them, finishing by downloading them as markdown files.

The markdown files each contain a table listing the threats, mitigations and risk scores, and they are loaded into memory and combined based on the threat description present in all of the markdown table. After they have been combined, they are saved in the JSON format, however they do not contain the asset field as this is not listed for the threats provided by STRIDEgpt.

F.3 ThreatCanvas

In ThreatCanvas, the defined input is given to the AI, which is running along side the regular threat modelling tool. The AI did not request any further information to model the system, and it creates the diagram and threat model simultaneously. The threat model can only be downloaded as a PDF, and therefore we manually extract the threat categories for each asset into a JSON file, mapping the asset to a list of threat categories.

The threat categories, descriptions, mitigations and risk score are all tied together, and this information is saved in to static JSON files. The threat and risk score information is saved in one file, together with the names of the mitigations that apply to each category. The mitigations and their name is saved in a separate file.

After this is setup, the assets are combined with the threat and mitigation files to create the actual threat model, during the combination, the name of the mitigation is concatenated with the description. If more than one mitigation applies, then all mitigations are merged into one long string. After the information has been merged, it is saved in the previously mentioned JSON format for the threat models.

G Measurements for G2

G.1 IriusRisk

Metric	Kappa
M12: C	0.03
M13: A	0.03
M14: T	0.98
M15: M	0.87
M16: R_{same}	0.61
M16: R_{more}	0.84
M16: R_{less}	0.64

Threat Model	No. of Threats
Baseline	13
Generated	57
Judge	**Identical Threats**
human	7
deepseek-r1:14b	12
gemma3:4b	13
phi3:14b	12
qwen2.5:7b	11
yi:9b	13

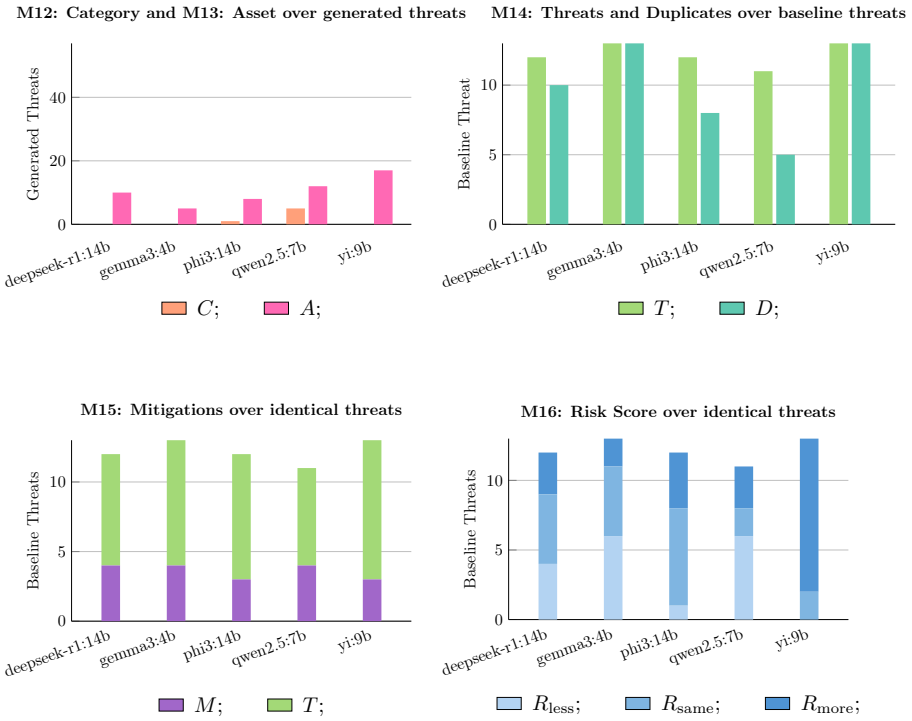
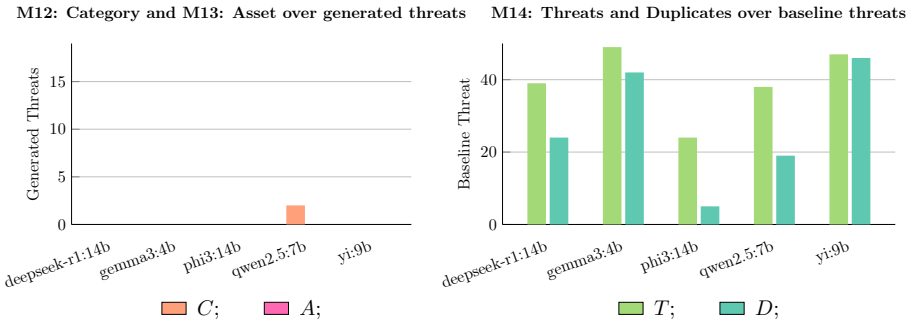


Figure 11: Comparison of baseline and generated Message Queue threat model for IriusRisk for M12, M13, M14, M15 and M16

Metric	Kappa
M12: C	0.00
M13: A	0.00
M14: T	0.85
M15: M	0.62
M16: R_{same}	0.38
M16: R_{more}	0.40
M16: R_{less}	0.57



Threat Model	No. of Threats
Baseline	50
Generated	19

Judge	Identical Threats
deepseek-r1:14b	39
gemma3:4b	49
phi3:14b	24
qwen2.5:7b	38
yi:9b	47

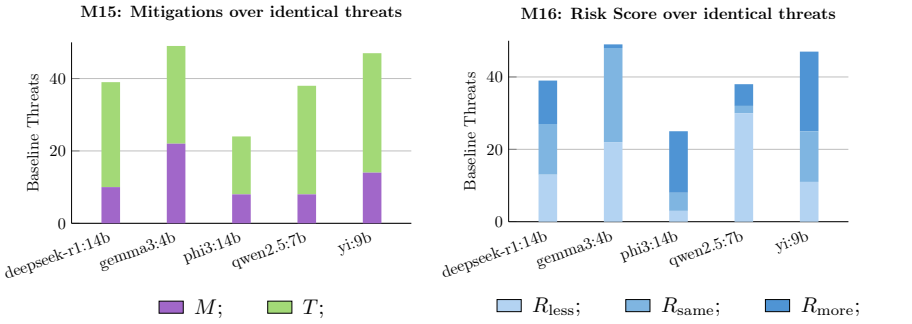
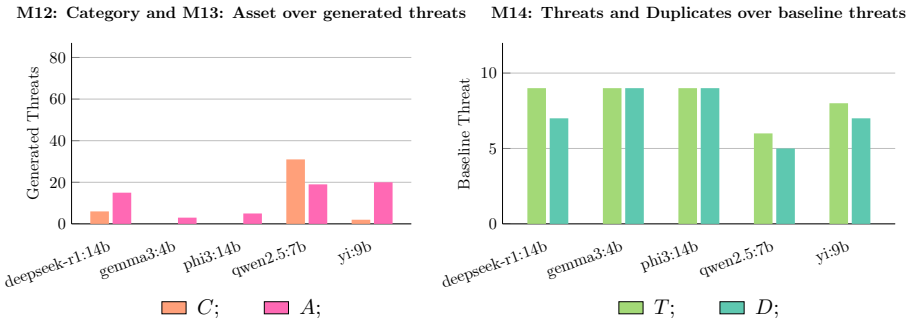


Figure 12: Comparison of baseline and generated Web threat model for IriusRisk for M12, M13, M14, M15 and M16

Metric	Kappa
M12: C	0.30
M13: A	0.30
M14: T	0.83
M15: M	0.46
M16: R_{same}	0.48
M16: R_{more}	0.51
M16: R_{less}	0.21



Threat Model	No. of Threats
Baseline	12
Generated	87

Judge	Identical Threats
deepseek-r1:14b	9
gemma3:4b	9
phi3:14b	9
qwen2.5:7b	6
yi:9b	8

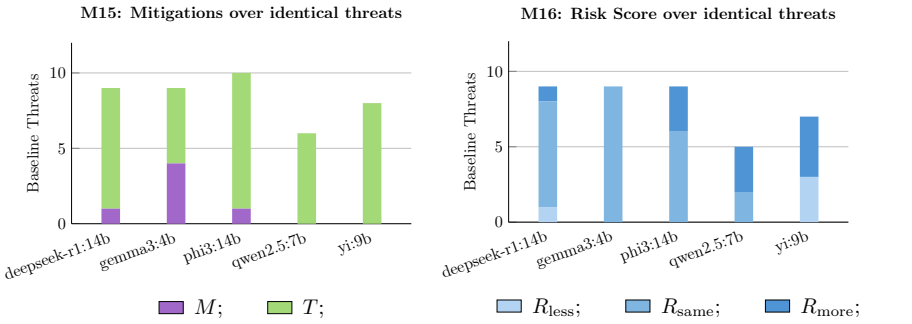


Figure 13: Comparison of baseline and generated IoT threat model for IriusRisk for M12, M13, M14, M15 and M16

G.2 STRIDEgpt

Metric	Kappa
M12: C	1.00
M13: A	0.01
M14: T	0.93
M15: M	0.86
M16: R_{same}	0.56
M16: R_{more}	0.47
M16: R_{less}	0.71

Threat Model	No. of Threats
Baseline	13
Generated	17

Judge	Identical Threats
human	7
deepseek-r1:14b	12
gemma3:4b	13
phi3:14b	10
qwen2.5:7b	10
yi:9b	13

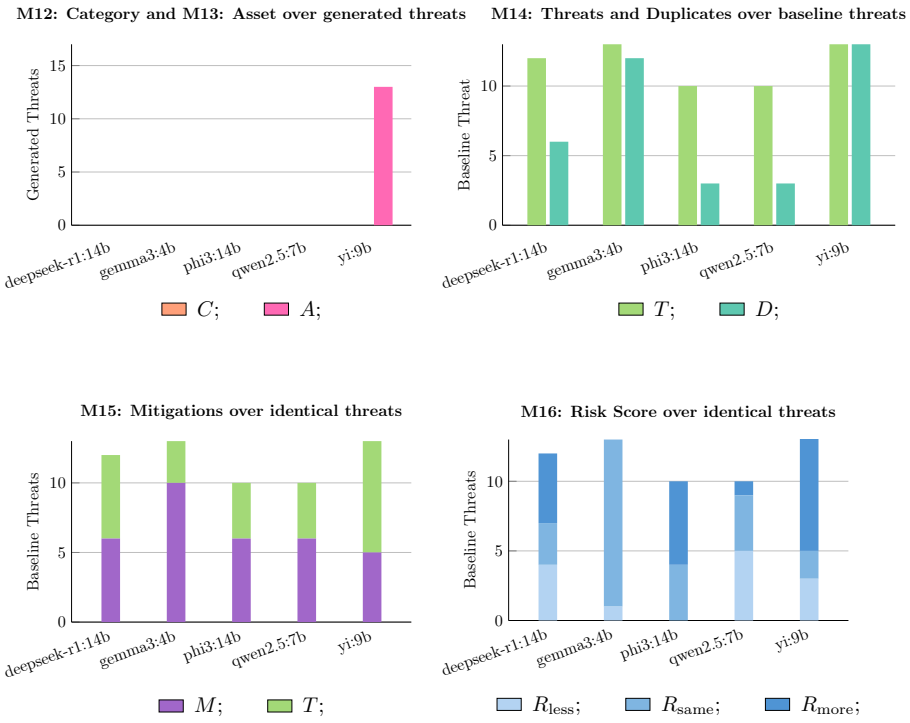


Figure 14: Comparison of baseline and generated Message Queue threat model for STRIDEgpt for M12, M13, M14, M15 and M16

Metric	Kappa
M12: C	1.00
M13: A	1.00
M14: T	0.96
M15: M	0.68
M16: R_{same}	0.70
M16: R_{more}	0.49
M16: R_{less}	0.70

Threat Model	No. of Threats
Baseline	50
Generated	17

Judge	Identical Threats
deepseek-r1:14b	46
gemma3:4b	49
phi3:14b	41
qwen2.5:7b	44
yi:9b	50

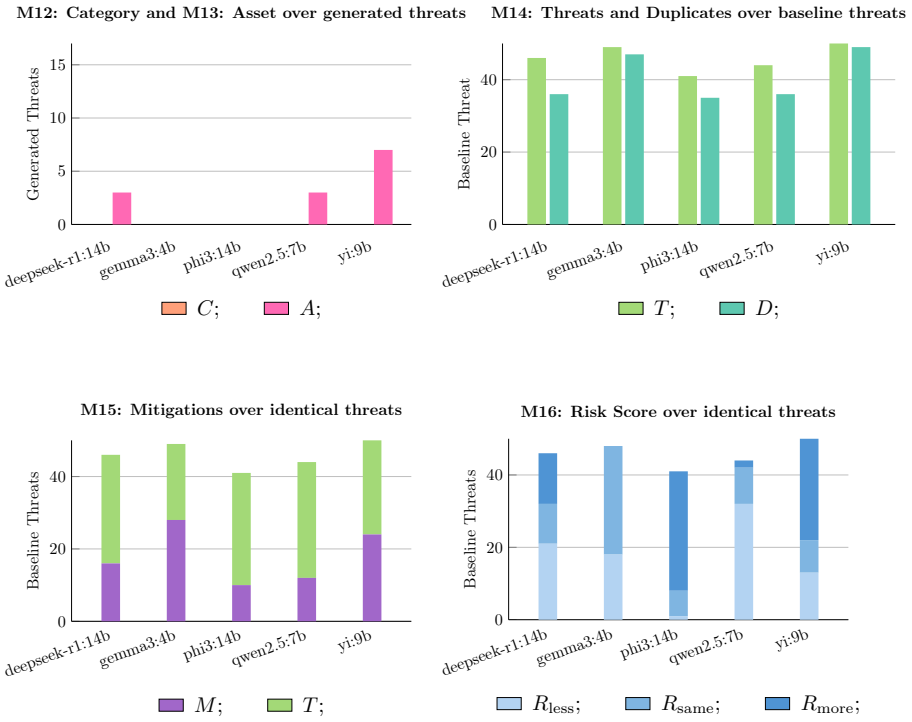


Figure 15: Comparison of baseline and generated Web threat model for STRIDEgpt for M12, M13, M14, M15 and M16

Metric	Kappa
M12: C	1.00
M13: A	1.00
M14: T	0.86
M15: M	0.54
M16: R_{same}	0.36
M16: R_{more}	0.55
M16: R_{less}	0.15

Threat Model	No. of Threats
Baseline	12
Generated	17

Judge	Identical Threats
deepseek-r1:14b	9
gemma3:4b	12
phi3:14b	8
qwen2.5:7b	7
yi:9b	11

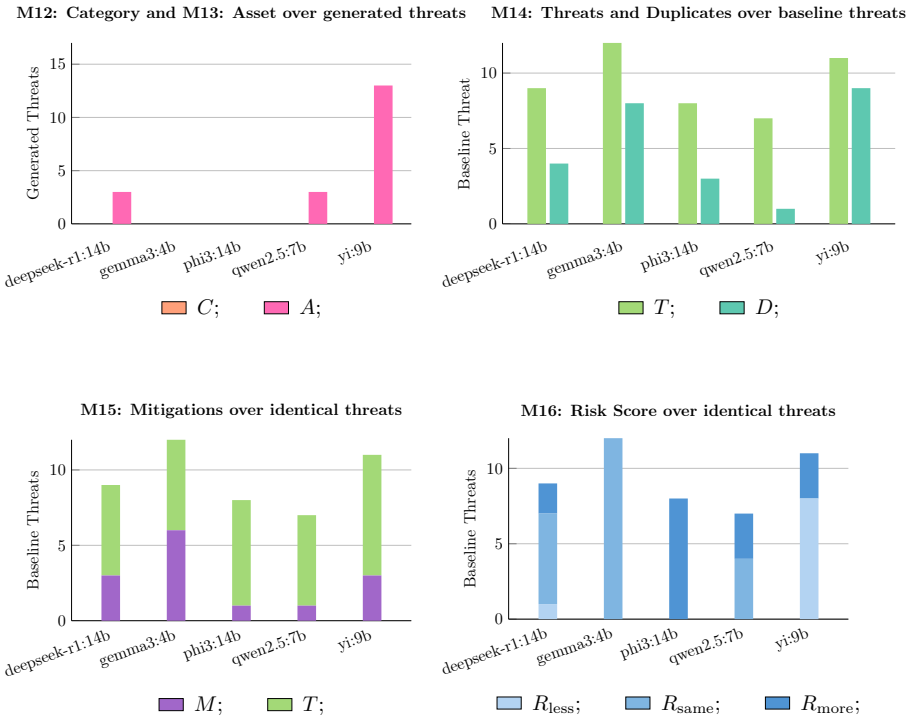
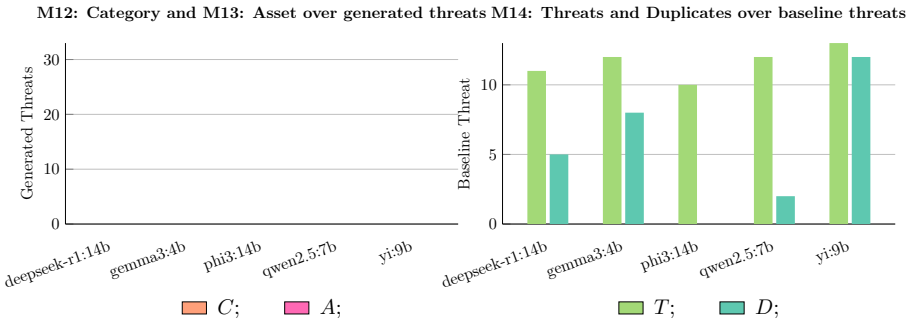


Figure 16: Comparison of baseline and generated IoT threat model for STRIDEgpt for M12, M13, M14, M15 and M16

G.3 ThreatCanvas

Metric	Kappa
M12: C	1.00
M13: A	1.00
M14: T	0.95
M15: M	0.40
M16: R_{same}	0.47
M16: R_{more}	0.31
M16: R_{less}	0.76



Threat Model	No. of Threats
Baseline	13
Generated	33

Judge	Identical Threats
human	9
deepseek-r1:14b	11
gemma3:4b	12
phi3:14b	10
qwen2.5:7b	11
yi:9b	13

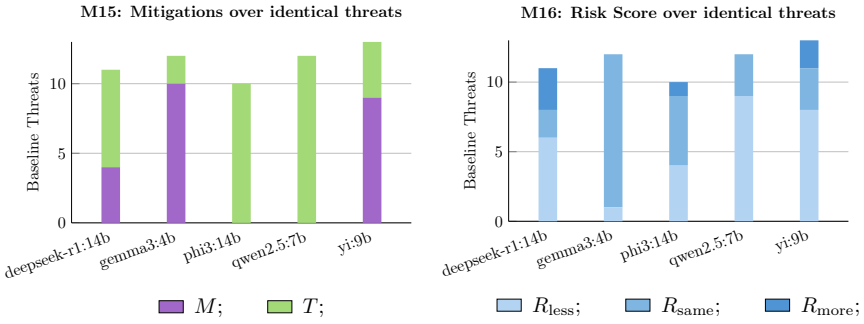


Figure 17: Comparison of baseline and generated Message Queue threat model for ThreatCanvas for M12, M13, M14, M15 and M16

Metric	Kappa
M12: C	1.00
M13: A	1.00
M14: T	0.90
M15: M	0.54
M16: R_{same}	0.64
M16: R_{more}	0.06
M16: R_{less}	0.57

Threat Model	No. of Threats
Baseline	50
Generated	14

Judge	Identical Threats
deepseek-r1:14b	38
gemma3:4b	49
phi3:14b	34
qwen2.5:7b	39
yi:9b	50

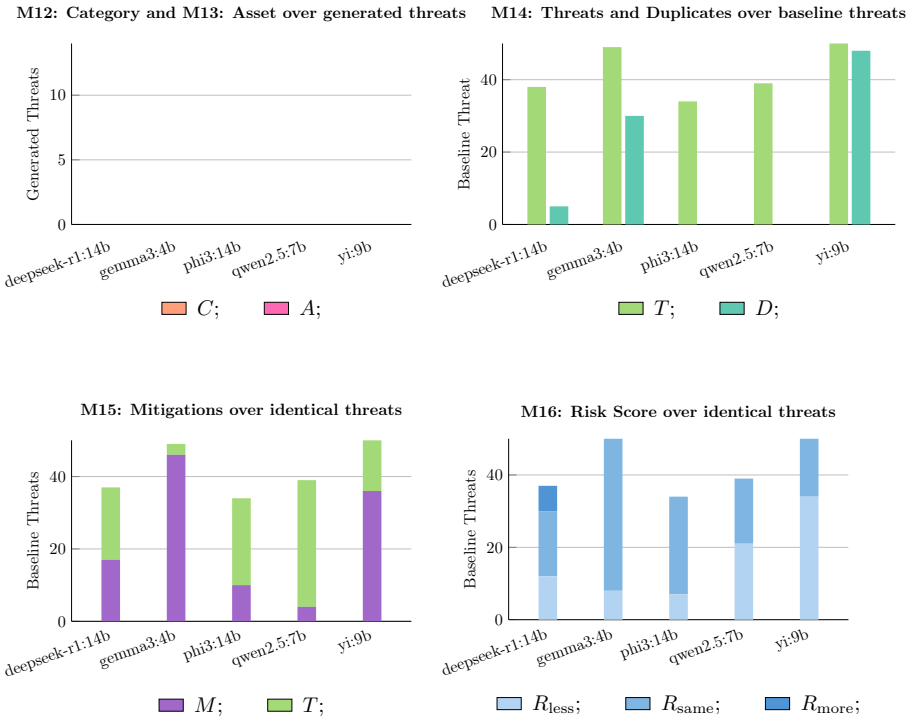


Figure 18: Comparison of baseline and generated Web threat model for ThreatCanvas for M12, M13, M14, M15 and M16

Metric	Kappa
M12: C	1.00
M13: A	1.00
M14: T	0.91
M15: M	0.71
M16: R_{same}	0.56
M16: R_{more}	0.34
M16: R_{less}	0.23

Threat Model	No. of Threats
Baseline	12
Generated	54

Judge	Identical Threats
deepseek-r1:14b	10
gemma3:4b	12
phi3:14b	9
qwen2.5:7b	9
yi:9b	12

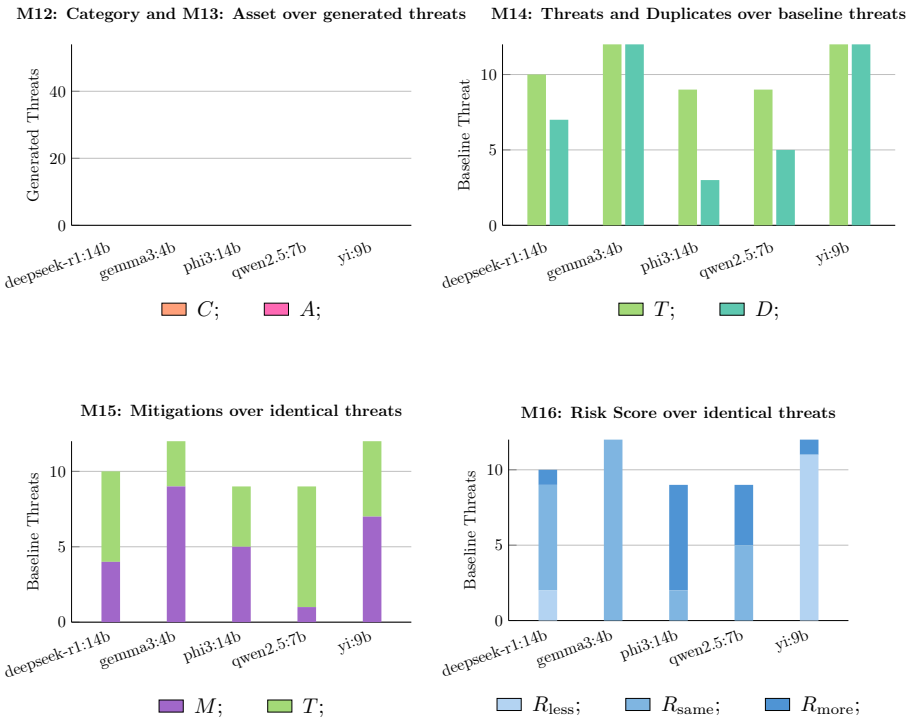


Figure 19: Comparison of baseline and generated IoT threat model for ThreatCanvas for M12, M13, M14, M15 and M16