Software Quality Journal (2024) 32:125–161
https://doi.org/10.1007/s11219-023-09634-4

RESEARCH

Systematic analysis of automated threat modelling techniques: Comparison of open-source tools

Daniele Granata[1] · **Massimiliano Rak**[1]

Accepted: 19 April 2023 / Published online: 25 May 2023

Abstract
Companies face increasing pressure to protect themselves and their customers from security threats. Security by design is a proactive approach that builds security into all aspects of a system from the ground up, rather than adding it on as an afterthought. By taking security into account at every stage of development, organizations can create systems that are more resistant to attacks and better able to recover from them if they do occur. One of the most relevant practices is threat modelling, i.e. the process of identifying and analysing the security threat to an information system, application, or network. These processes require security experts with high skills to anticipate possible issues: therefore, it is a costly task and requires a lot of time. To face these problems, many different automated threat modelling methodologies are emerging. This paper first carries out a systematic literature review (SLR) aimed at both having an overview of the automated threat modelling techniques used in literature and enumerating all the tools that implement these techniques. Then, an analysis was carried out considering four open-source tools and a comparison with our threat modelling approach using a simple, but significant case study: an e-commerce site developed on top of WordPress.

Keywords Security · Automated threat modelling · Security assessment · Threat modelling tools

1 Introduction

In the digital age, security has become increasingly important, as more and more information is stored electronically and more systems are connected to the internet. The mass adoption of these systems has led to the need of ensuring security by regulations that impose security requirements. To address security, best practices suggest the adoption of

Massimiliano Rak contributed equally to this work.

✉ Daniele Granata
daniele.granata@unicampania.it

Massimiliano Rak
massimiliano.rak@unicampania.it

[1] Department of Engineering, University of Campania Luigi Vanvitelli, via Roma, 29, Aversa I-81031, CE, Italy

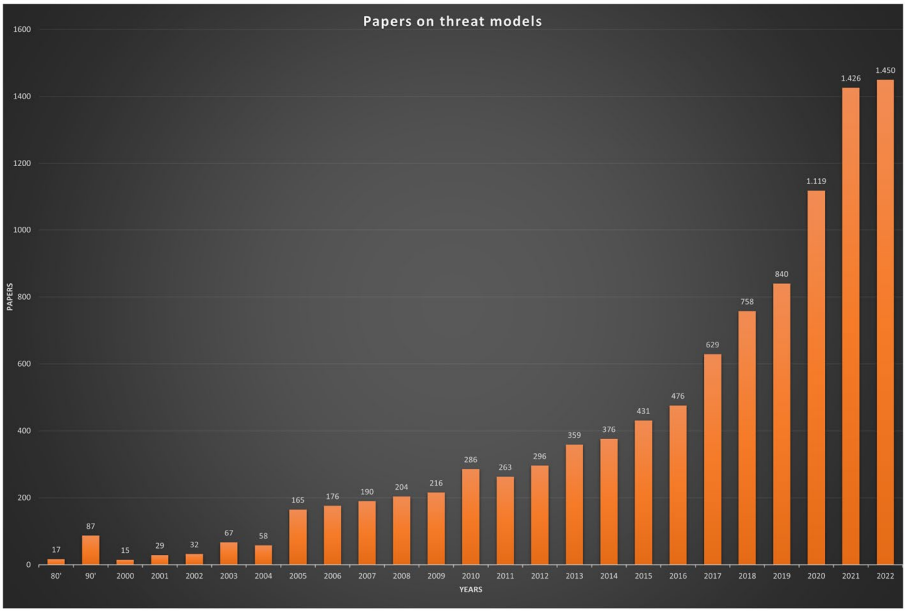


Fig. 1 Paper on threat model

threat modelling and risk analysis methodologies that allow the security administrator to obtain (in a preliminary way) information on the security problems from the early stages of the system life cycle. The choice of using threat modelling processes is also widely shared by the scientific community and is becoming increasingly widespread. To demonstrate the increasing proposals of threat modelling techniques, we produced a bar graph, shown in Fig. 1, that plots the number of papers related to *threat model** from 1980 to 2022. The digital libraries used to produce it are IEEE, Springer, Scopus, and ACM. It is worth noting that results from Springer are filtered to the articles in *Computer Science*, written in English. It shows clearly how much the scientific literature has focused on threat modelling in recent decades.

The overall results, described in Table 1, claim that there are many different proposals in the literature.

In particular, this paper extended the conference work (Granata et al., 2022) and aims at providing a systematic analysis of automated threat modelling techniques and a comparison of open-source tools.

The main contributions of this work are (i) a literature review on automated threat modelling approaches and their supporting tools; (ii) a detailed description of the supporting tools; and (iii) a comparison carried out on a simple, but significant case study, involving a well-known content management system (CMS) platform: WordPress

Table 1 Results threat model*

Search string	IEEE	Scopus	Springer	ACM	Total
Threat model	574	2204	7159	175	10,112
Threat modelling	44	1421	7144	65	8674

It is worth noticing that the new contributions compared to the previous work are (i) the systematic analysis of the threat modelling methodologies; (ii) the selection of a lot of tools based on the criteria found in the previous analysis; and (iii) considering this selection, a more precise analysis was made on one tool supported by OWASP: PyTM.

The remainder of the paper is organized as follows: Sect. 3 summarizes the automated threat modelling techniques in a systematic way and also enumerates all the supporting tools; Section 4 describes the tools that we addressed in our analysis. Section 5 compares the tools, using the WordPress application as a basis for the comparison. Finally, Section 6 summarizes our conclusions and future work.

2 Related work

As already mentioned in the introduction section, over the years threat modelling has undergone many changes, and numerous approaches and tools have been developed in some surveys and research. Despite the wealth of literature, there is a need for a systematic and comprehensive review of the state of the art in threat modelling. A thorough evaluation of the literature would concentrate on the benefits and weaknesses of some methodologies as well as the associated tools. This review would also provide a clear overview of the existing gaps in the literature, allowing for a more focused and targeted approach to addressing these disparities. By taking a systematic and comprehensive approach to the existing literature, it will be possible to gain a better understanding of the current state of the art in threat modelling, and to identify areas for future research and development.

Existing surveys focus on methodologies and techniques aimed at modelling a software system and automatically deriving the threats and attacks. Some authors focused on analysing the most common techniques and the related and open-source tools. Hussain et al. (2014) focus on several threat modelling techniques, such as STRIDE, attack-tree, and CORAS, and related tools like Microsoft threat modelling and the CORAS tool. This work highlights the need for using formal modelling techniques and formal methods to verify the requirements. Even if they analysed both threat modelling techniques and the implementations, the analysis is not systematic since they focused on the most common models and tools. A similar study is conducted by Bernsmed et al. (2021) that conducted research on the usability of threat modelling techniques in industrial and academic contexts by focusing only on the most used modelling technique (i.e. data flow diagram) and using STRIDE classification and threat modelling automation. All the application scenarios have some difficulties in implementing threat modelling concepts in practice and suggest focusing on attacks using some automation techniques.

Some other authors instead took threat modelling techniques into account by applying them only in a specific context. As an example, Nweke and Wolthusen (2020) focus on asset-centric threat modelling techniques that aim to protect system components by producing threat models. They analyse techniques such as DREAD, Trike, and OCTAVE and their strengths and limitations. The authors also suggest that formal methods can improve the automation of threat modelling and security assessment processes. A specific study that takes into account the IoT domain was conducted by Mahak and Singh (2021) that analysed the existing threat modelling methodologies aimed at assessing all the threats an IoT infrastructure is affected by and evaluating the related risks through some risk analysis procedures. Omotunde and Ibrahim (2015) explain how threats are modelled in the design phase and present research activities that use techniques such as attack-specific modelling

techniques and misuse case analysis. They suggest integrating threat modelling practices with these techniques. All the relevant surveys were applied to specific contexts, but the most complete study was conducted by Tatam et al. (2021). Their study discusses the limitations, strengths, and gaps in threat modelling processes for advanced persistent threats (APTs). The authors present a taxonomy of threat modelling approaches, including asset-centric, threat-centric, data-centric, and system-centric. They analyse each technique and suggest a hybrid approach that considers both threat modelling activities and operational models based on attack patterns. Despite the completeness of the taxonomy they propose, the authors have not focused on the tools that implement the methodologies described.

A few different works were carried out on the most common tools related to the existing methodologies.

Shi et al. (2021) provide an overview of threat modelling tools, dividing them into those based on diagrams, text-based models, and others, but they focus only on the most commonly used.

Unlike the latter, Tan and Garg's (Tan and Garg) research study focuses on open-source tools that are recently committed, applicable to different domains, and have a machine-readable model. They label each tool with features such as complexity of logic, amenability of custom threats, operational usability, security functionality, and extensible for privacy, and assign a score for each feature to determine the most useful tool. To the best of the authors' knowledge, there has been no complete threat modelling analysis carried out that provides a precise description of how threat modelling works are selected.

Despite the surveys in the literature regarding threat modelling techniques, as far as we know, the literature lacks a systematic study that takes into account both the threat modelling techniques and the tools that implement these techniques. For this reason, the proposed work aims to systematically produce an updated review of these techniques in different domains and to show the applicability across the use of some tools.

3 Systematic literature review

Automated threat modelling approaches have become widespread. In order to have a complete literature overview of the automated approaches used to model the systems and obtain all the possible threats it is affected by, we applied for a systematic literature review (SLR). The technique used, explained in Subsection 3.1, aims to answer some research questions synthetically, as many works have been carried out on this line of research in very different ways.

3.1 Methodology

The systematic literature review has been conducted using the method proposed by Kitchenham et al. (2009). According to the authors, the review process is divided into three parts: Planning, Conducting, and Reporting (as shown in Fig. 2).

The Planning phase consists of developing a protocol used for searching articles from the sources, including and excluding papers from the overall results to answer some research questions. In the Conducting phase, we apply the rules developed in the protocol to obtain the list of the accepted papers suitable for answering research questions previously developed. The Reporting phase involves writing up the results of the review and circulating the results to potentially interested parties. The aim of conducting an SLR is both

Fig. 2 Three-step SLR methodology

having a complete overview of the automated threat modelling approaches and enumerating the tools used to support the processes. We used the tool StArt, developed by LaPES (Fabbri et al. 2016), in order to support the systematic literature review. The tool was used only for supporting us to obtain the data of the paper, avoid duplicated papers that may be included in different digital libraries and collect data in order to produce charts. In the next subsections, we will show in detail the protocol used to perform the SLR and the results from the execution phase.

3.2 Planning

The first step of each SLR is the individuation of the need for the review. The aim of the article is to understand how widespread are automation techniques and which methodologies and tools are used. In order to have an exhaustive overview of these techniques, we formulate two different research questions:

- **RQ1**: Which methodologies are used to produce a threat model automatically?
- **RQ2**: Which open-source tools are used to support the automated threat modelling methodologies?

To answer these questions, we used a keyword-based approach to select the papers from which to obtain the data. The review keywords we choose are **Systematic threat model***, **Automated threat model***, **Tool for threat model***, and **tool supporting threat model***. The sources we choose as search engines are the most common ones: IEEE, Scopus, Springer, and ACM. Before searching papers from the sources, we applied a preliminary analysis in order to choose the right query and avoid not-relevant results.

We firstly present three different search approaches, shown in Table 2. Each search query has been performed on each source considering the abstract, the title, and the related keywords. It is important to take into account that all the results are from 2012 until now and, as for Table 1, Springer results are only *articles in Computer Science written in English*.

As shown by the high numbers, the first search approach is too generic since it can contain some non-relevant articles. The second approach instead contains some strict constraints on the query string (e.g. if it contains an "Automated Threat Model"), and we obtained few results; therefore, we discarded it. The third approach gives a reasonable number of papers that could be relevant to answer our research questions. For this reason, we used the third query as input to conduct the SLR. Chosen the search query, we also needed eligibility criteria to select the relevant works. The inclusion and exclusion criteria choice depends on the aim of the systematic literature review: having an overview of the automated threat modelling approaches used in literature and the existing supporting tools. For this reason, we accepted all the papers that:

- Describes a fully or partially automated threat modelling methodology;
- Presents a supporting tool aimed at automating threat modelling approaches.

Table 2 Different search approaches

Num.	Search approach	IEEE	Scopus	Springer	ACM	Total
1	(Automated AND Threat AND Model*) OR (Systematic AND Threat AND Model*) OR (Tool AND Threat AND Model*)	1089	2841	6083	64	10,077
2	(Automated Threat Model) OR (Systematic Threat Model*) OR (Tool supporting Threat Model*) OR (Tool for Threat Model*)	4	18	3	2	27
3	(Automated AND Threat Model*) OR (Systematic AND Threat Model*) OR (Tool AND (supporting) Threat AND Model*)	74	269	109	14	466

The inclusion criteria are built considering both RQ1 and RQ2. We consider as *automated threat modelling a process that does not require human effort to evaluate the system security (previously modelled)*. A methodology can also be partially automated when a few tasks are performed manually, but the effort is still minimum compared to the whole process. The exclusion criteria instead are the following:

- The paper is not about a new threat modelling methodology;
- Threat modelling is performed to a specific field;
- The process is carried out manually;
- The paper is not downloadable;
- The language is not English.

Once the papers have been selected using the inclusion/exclusion criteria, some data can be extracted to answer the research questions. According to our protocol, we collected different data from each paper: (i) the technique used to model the architecture, (ii) the threat classification method used, (iii) the way their methodology and supporting tool select the threats from the model, and (iv) the supporting tool they provide. Using all these extraction fields defined in the protocol, we expect to have a complete overview of automated threat modelling and the related tool.

3.3 Conducting

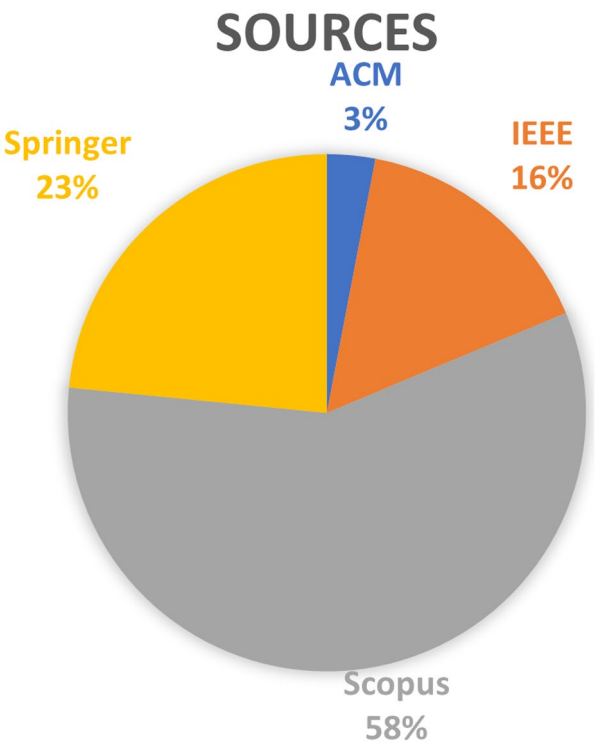
The second phase of a systematic literature review is the identification and selection of relevant papers (i.e. the ones that meet the criteria described in the protocol) and the extraction of the relevant data (i.e. according to the data extraction fields provided in the protocol). We performed the third query described in Table 2 in all the selected sources. From the search, we obtained 465 papers as a result. As shown from the pie chart in Fig. 3, the most selected papers are from Scopus library (58%, 269 papers), followed by Springer (23%, 109 papers), IEEE (16%, 73 papers), and ACM with only 3% (14 papers).

The first step to being carried out when conducting an SLR is to apply the eligibility criteria by reading only the abstracts. Our approach relies on the easy to use of StArt tool that automatically takes the Bibtex files (automatically downloaded from each source engine) as input and then lists all the papers in an interactive view to obtain easily all the information needed (abstract, keywords, journal, etc.). Applying the inclusion and exclusion criteria to the abstracts, we reduced the number of papers from 465 to 115. StArt tool was useful to automatically remove the 103 duplicated papers coming from different sources, as shown in Fig. 4.

Once the number of papers has been reduced by the reading of the abstracts, a deeper analysis has to be performed by carefully reading the accepted papers. Considering the same criteria used in the previous phase, but applied to the overall paper, only 55 papers met the inclusion criteria and have been used to extract the data. On the other hand, 9 papers have been classified as Duplicated (StArt was not able to recognize them) and 51 have been rejected as out of the SLR scope. Analysing the inclusion criteria from Fig. 5a, it is worth noting that most of them presented a new threat modelling methodology, but not a supporting tool.

Similarly, a bar chart has been produced by the StArt tool for all the rejected papers in Fig. 5b. Most papers did not present threat modelling techniques but were still about security

Fig. 3 Sources chart



assessment procedures. Other works were not generically applicable but concerned a specific area. Few works instead were not available to download[1].

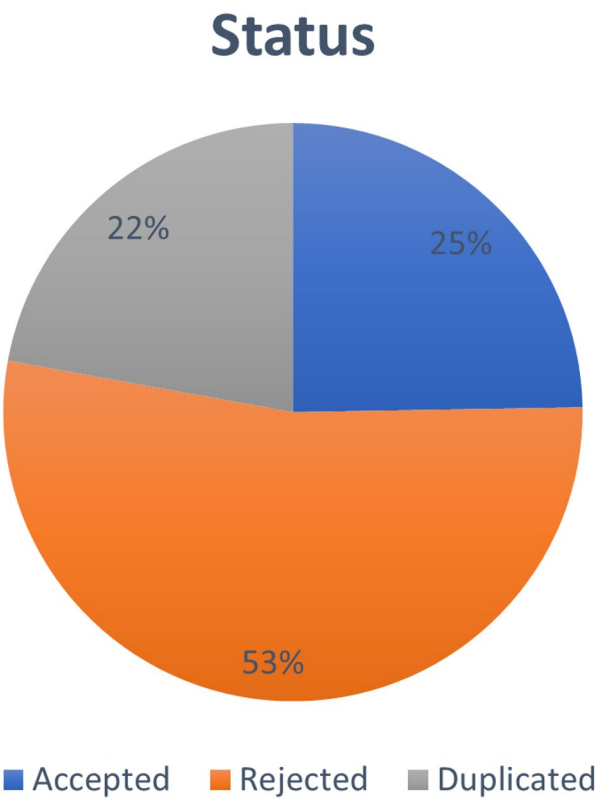
3.4 Results

Data have been extracted from the paper performing the SLR. We divided all the data into 4 groups taking into account the data field described in the SLR protocol: (I) modelling data, (ii) threat classification data, (iii) threat selection approaches, and (iv) tools. As highlighted from Table 3, most of the approaches used to automate threat modelling rely on data flow diagram (DFD) modelling technique.

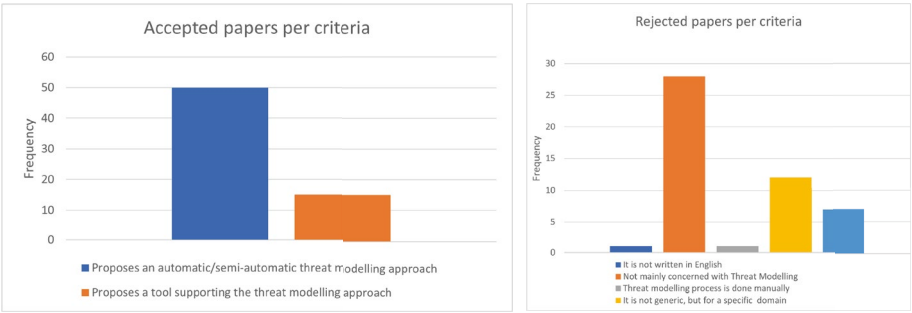
A data flow diagram is a model, introduced by DeMarco (1979), that describes a system as a set of specific components, and the data flows from/to its components. Each component of the system can be modelled as a generic function, a database and external agents, while a data flow is represented as an arch. Because of its simplicity, some authors (Haitao et al., 2022; Frydman et al., 2014; Mani & Venkat, 2017; Von Der Assen et al., 2022; Sion et al., 2021; Martins et al., 2015) have extended the model with further component types useful to obtain security features. As a result of the SLR, the most widespread models are based on graphs (e.g. DFD). Several works used their own graph-based model to describe

[1] All the papers have been downloaded using *Università della Campania Luigi Vanvitelli* Institutional Sign In.

Fig. 4 Selection result chart



the components (i.e. nodes) and the relationships between them (i.e. arcs). A recurring model resulting from the SLR is the MACM (multi-purpose application composition model), described in detail by the authors (Salzillo et al., 2020; Rak et al., 2019; Casola et al., 2019; Rak et al., 2020; Granata et al., 2021; Ficco et al., 2021; Rak et al., 2022). While graph-based models are highly expressive and very intuitive, some authors prefer to use the unified modelling language (UML) standard (Rumbaugh et al., 2004) to describe the system. The authors extended some UML diagrams to describe security requirements and automatically derive security issues. Anyway, the modelling phase is the first step of a security assessment process and depends on its scope. For this reason, some authors rely on specific own models to describe the software system (Messe et al., 2020; Chen, 2018; Schaad & Borozdin, 2012; Dominic et al., 2016; Ding et al., 2017; Monteuuis et al., 2018), while some of them are only interested in their assets (i.e. valuable items owned by the company) (Althar et al., 2022; Valenza et al., 2022; Schlegel et al., 2015; Hasan & Hasan, 2021; Haji et al., 2019; Ansari et al., 2019; Ivanova & Ivanenko, 2022). Once the model of a system is provided, an automated (or semi-automated) threat modelling technique can derive threats from it without much additional effort. Most of the threat modelling techniques in the literature are based on a specific threat classification. As shown in Table 4, the most common threat classification technique is STRIDE (Ansari et al., 2019). Since the threats are very heterogeneous, a complete threat model is difficult to generate.



(a) Accepted Papers **(b)** Rejected Papers

Fig. 5 Inclusion and exclusion criteria

To face this problem, all the threats can be described in classes: Spoofing, Tampering, Reputation, Information Disclosure, Denial of Services, and Elevation of Privileges.

In literature, there are several extensions (Chen, 2018; Monteuuis et al., 2018; Chen, 2019; Vallant et al., 2021) of STRIDE classification (e.g. considering Privacy and a different threat class). Similar threat classification is provided from the LINDDUN methodology (LINDDUN, LINDDUN). According to the LINDDUN knowledge base, there are 7 threat categories encapsulated in the LINDDUN acronym: Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance. Some authors (Wuyts, 2020; Sion et al., 2018; Wuyts et al., 2018) associated their own threats to LINDDUN classes and used the LINDDUN knowledge base as input to the threat modelling process. STRIDE and LINDDUN are related to the malicious behaviours the threat can implement, another approach is to consider the security requirements the threat can compromise in terms of CIA: Confidentiality, Integrity, and Availability. Similarly, considering Privacy as a compromise-able requirement, some extensions considering CIAP have been considered. STRIDE and LUNDDUN classify each threat expressed by a high-level malicious behaviours. Few works instead use *Attack patterns* classification to perform threat modelling. According to CAPEC, *Attack Patterns are descriptions of the common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities.*. These approaches (Messe et al., 2020; Alwaheidi & Islam, 2022) rely on CAPEC knowledge base and their attack-pattern classification, therefore they are more attack-specific compared with STRIDE and LINDDUN. As a result of classifications in the literature, some more elaborate techniques make use of first-order logic (Althar et al., 2022) (i.e. Prolog rules), or data structures such as threat-trees (Moreira et al., 2016; Radoglou-Grammatikis et al., 2022), threat paths (Ramazanzadeh et al., 2022), and threat matrix (Ivanova & Ivanenko, 2022).

Based on well-defined models and threat classification, there are much different threat modelling techniques that enable enumerating all the threats the system is affected by. A full report of all threat selection approaches is shown in Table 5. It is worth noting that some articles can use a combination of the selection approaches taking into account different points of view.

As in evidence, most approaches (especially the ones based on graph models) select all the threats from their threat catalogue using some labels. Labels are some words that describe the

Table 3 Modelling technique used in literature

Model	Papers	Num.
Data flow diagram (DFD)	AbuEmera et al. (2022), Khan et al. (2017), Wuyts (2020), Verreydt et al. (2022), Sion et al. (2018), Danielis et al. (2020), Casola et al. (2021), Asif et al, (2021), Wuyts et al. (2018), Chen (2019), Vallant et al. (2021), Mahmood et al. (2022), Alwaheidi and Islam (2022), Brown et al. (2022)	14
DFD extended	Haitao et al. (2022), Frydman et al. (2014), Mani and Venkat (2017), Von Der Assen et al. (2022), Sion et al. (2021), Martins et al. (2015)	6
MACM	Salzillo et al. (2020), Rak et al. (2019), Casola et al. (2019), Rak et al. (2020), Granata et al. (2021), Ficco et al. (2021), Rak et al. (2022), Granata et al. (2022)	8
Own graph-based	Almubairik and Wills (2016), Meland et al. (2014), Shelupanov and Konev (2019), Kosachenko et al. (2021)	4
Own architecture model	Messe et al. (2020), Chen (2018), Schaad and Borozdin (2012), Dominic et al. (2016), Ding et al. (2017), Monteuuis et al. (2018)	6
List of components/assets	Althar et al. (2022), Valenza et al. (2022), Schlegel et al. (2015), Hasan and Hasan (2021), Haji et al. (2019), Ansari et al. (2019), Ivanova and Ivanenko (2022)	7
UML-based	Moreira et al. (2016), Zeinali and Hadavi (2018), Naagas (2018)	3
Developers brainstorming	Hoque and Hasan (2019), Jamil et al. (2021)	2
Not formally expressed/other minor techniques	Radoglou-Grammatikis et al. (2022), Leander et al. (2019), Saatkamp et al. (2019), Wirtz and Heisel (2020), Ramazanzadeh et al. (2022)	5

Table 4 Threat classification used in literature

Threat classification	Papers	Num.
STRIDE	Salzillo et al. (2020), Rak et al. (2019), Casola et al. (2019), Rak et al. (2020), Granata et al. (2021), Ficco et al. (2021), Rak et al. (2022), Hoque and Hasan (2019), Haitao et al. (2022), Jamil et al. (2021), AbuEmera et al. (2022), Khan et al. (2017), Von Der Assen et al. (2022), Verreydt et al. (2022), Martins et al. (2015), Saatkamp et al. (2019), Danielis et al. (2020), Casola et al. (2021), Asif et al, (2021), Leander et al. (2019), Hasan and Hasan (2021), Schaad and Borozdin (2012), Kornecki and Janusz (2015), Dominic et al. (2016), Mani and Venkat (2017), Naagas (2018), Haji et al. (2019), Ansari et al. (2019), Mahmood et al. (2022), Brown et al. (2022)	30
Extended STRIDE	Chen (2018), Monteuuis et al. (2018), Chen (2019), Vallant et al. (2021)	4
CIA and extension	Salzillo et al. (2020), Rak et al. (2019), Casola et al. (2019), Rak et al. (2020), Granata et al. (2021), Ficco et al. (2021), Rak et al. (2022), Althar et al. (2022), Casola et al. (2021), Danielis et al. (2020), Wirtz and Heisel (2020), Hoque and Hasan (2019), Shelupanov and Konev (2019)	13
LINDDUN	Wuyts (2020), Sion et al. (2018), Wuyts et al. (2018)	3
CAPEC	Messe et al. (2020), Alwaheidi and Islam (2022)	2
Prolog-rules	Althar et al. (2022)	1
Threat tree	Moreira et al. (2016), Radoglou-Grammatikis et al. (2022)	2
Threat paths	Ramazanzadeh et al. (2022)	1
Threat matrix	Ivanova and Ivanenko (2022)	1
Own language	Almubairik and Wills (2016), Sion et al. (2021), Zeinali and Hadavi (2018), Meland et al. (2014)	4
None	Schlegel et al. (2015), Frydman et al. (2014), Kosachenko et al. (2021)	3

Table 5 Threat selection techniques used in literature

Threat selection	Papers	Num.
Label-based	Salzillo et al. (2020), Rak et al. (2019), Casola et al. (2019), Rak et al. (2020), Granata et al. (2021), Ficco et al. (2021), Rak et al. (2022), Jamil et al. (2021), Khan et al. (2017), Althar et al. (2022), Martins et al. (2015), Chen (2018), Casola et al. (2021), Schaad and Borozdin (2012), Frydman et al. (2014), Kornecki and Janusz (2015), Meland et al. (2014), Ding et al. (2017), Granata et al. (2022)	19
STRIDE-based	AbuEmera et al. (2022), Khan et al. (2017), Danielis et al. (2020), Leander et al. (2019), Hasan and Hasan (2021), Mani and Venkat (2017), Naagas (2018)	7
STRIDE/LINDDUN-to-DFD	Sion et al. (2018), Wuyts (2020), Monteuuis et al. (2018), Chen (2019)	3
Relationships-based	Sion et al. (2018), Chen (2018), Danielis et al. (2020), Asif et al, (2021), Zeinali and Hadavi (2018), Ding et al. (2017), Shelupanov and Konev (2019), Vallant et al. (2021), Rak et al. (2022), Granata et al. (2022), Mahmood et al. (2022)	11
Propagation rules	Verreydt et al. (2022), Meland et al. (2014), Ivanova and Ivanenko (2022), Rak et al. (2022)	4
Threat patterns/rule-based	Sion et al. (2021), Zeinali and Hadavi (2018), Frydman et al. (2014), Messe et al. (2020), Valenza et al. (2022)	5
Based on model requirements	Moreira et al. (2016), Wirtz and Heisel (2020)	2
Using questionnaires	Schaad and Borozdin (2012), Wuyts et al. (2018)	2
Threat tree catalogue	Wuyts et al. (2018), Haji et al. (2019)	2
Using generic parameters	Dominic et al. (2016), Ansari et al. (2019)	2
Other minor techniques	Von Der Assen et al. (2022), Haitao et al. (2022), Ramazanzadeh et al. (2022), Alwaheidi and Islam (2022)	4
Manually or not-defined	Almubairik and Wills (2016), Schlegel et al. (2015), Radoglou-Grammatikis et al. (2022), Hoque and Hasan (2019), Saatkamp et al. (2019), Zeinali and Hadavi (2018)	6

behaviours of a component. Each component can be associated with one or more labels. Leveraging this relationship, the literature approaches automatically lists all the threat for each component. A common and simple approach consists of using STRIDE classes. The model is decomposed and the STRIDE threats are analysed component-per-component. The overall threat model is a report for each STRIDE class. Despite the simplicity of this approach, some authors prefer a more detailed approach. Some techniques still rely on STRIDE (or LINDDUN), but they mapped each class to the component typology described in the model (e.g. DFD element type). This is useful to automatically derive STRIDE threat for each component. The used approach is fully automated, but is not detailed (e.g. threats belonging to the same class can be several). Microsoft proposes a methodology implemented by their tool (Microsoft, 2018) that automatically derives threats from the relationships between the components. They take into account the relationship type between two components and also their typology (e.g. a threat can be due to the communication between a web application and a database). Several works (Sion et al., 2018; Chen, 2018; Danielis et al., 2020; Asif et al, 2021; Zeinali & Hadavi, 2018; Ding et al., 2017; Shelupanov & Konev, 2019; Vallant et al., 2021; Mahmood et al., 2022) based their approach on the one proposed by Microsoft (using also their tool). Other authors, instead consider the relationship as an additional threat source depending on the protocol used (Rak et al., 2022; Granata et al., 2022). Similar to their approaches, some authors (Verreydt et al., 2022; Meland et al., 2014; Ivanova & Ivanenko, 2022; Rak et al., 2022) claim that relationships and components are not the only threat source. According to them, some threats can be added to the overall threat model considering some Propagation rules (e.g. a threat can compromise a component, but also the services used by it). Unlike these simple approaches, there are more complex threat selection criteria based on pattern recognition in the graph or specific rules mapped with threats (Sion et al., 2021; Zeinali & Hadavi, 2018; Frydman et al., 2014; Messe et al., 2020; Valenza et al., 2022). Threat modelling approaches aim also at ensuring security requirements in the system. Few authors proposed their methodology based on model requirements (Moreira et al., 2016; Wirtz and Heisel, 2020). As a result of SLR, we also analysed some less widespread techniques using questionnaires (Schaad & Borozdin, 2012; Wuyts et al., 2018), threat tree (Wuyts et al., 2018; Haji et al., 2019), and generic parameters (Dominic et al., 2016; Ansari et al. (2019). Some of them instead used OWASP ASVS (Von Der Assen et al., 2022), Petri-nets (Ramazanzadeh et al., 2022), weaknesses from CWE database (Alwaheidi & Islam, 2022), and attack-paths (Haitao et al., 2022).

3.4.1 Tools

Once we analysed all the methodologies, we provide an enumeration of the tools used to implement them. The most commonly used for automating threat modelling is Microsoft threat modelling (Microsoft, 2018) used in 8 papers from the SLR. A research line carried out by Granata et al. (2022) used instead *sla-generator* to perform threat modelling and even risk analysis. Another supported tool is Threat Dragon provided by OWASP (Bhattacharya, 2020). Some others' tool are SPARTA (Sion et al., 2018) from KU Leuven, CoreTM (Von Der Assen et al., 2022), TAMELESS (Valenza et al., 2022), MetaGME (Martins et al., 2015), TAM tool (Schaad & Borozdin, 2012), AutSEC (Frydman et al., 2014), and STS-tool (Meland et al., 2014). The SLR results reported 9 threat modelling tools, and we enriched this list through a specific research by adding further tools such as CAIRIS (Faily, 2018), Threats Manager Studio (Curzi, 2020), Threatspec (Fraser Scott & Smotrakov, 2019), PyTM (pyTM, 2019), and Threagile (Threat Agile, 2020). It is worth noticing that in this

work, the enumeration of the tools is not complete, but it both derives from the SLR and is carried out manually. This is why our focus regards the threat modelling techniques each tool relies on and the tool is needful only to validate them.

Considering all these tools aimed at performing automatic threat modelling, we decided to analyse the open-source ones that do not require any manual task to select the threats and focus on web application Domain. Accordingly, a full analysis of the tools that meet these requirements is shown in Table 6. For each tool selected from both the SLR and the manual extension, the analysis provides the modelling techniques. The modelling field describes two different parameters: (i) how the model is created by the user (i.e. graphically or by coding) and (ii) the modelling technique (e.g. Threatspec generates a DFD graph using the code provided as an input, instead OWASP Threat Dragon uses directly a DFD model as an input).

The domain field is a list of all the areas in which the tools have been applied according to the SLR or the documentation page. It is worth noting that taking into account the application domains is useful to choose the appropriate tool for the threat modelling scope. In addition to the threat modelling process, some tools were analysed to provide a risk analysis process aimed at offering a rough evaluation of the risk (i.e. the probability that a threat would occur). Among all the selected tools, sla-generator, SPARTA, and Threat-agile support risk analysis and (in some cases) suggest the appropriate countermeasures to reduce the risk. The threat selection methodology column describes how each tool can use one or more approaches (described as a result of the SLR in Table 5) to select the threats compromising each asset. As an example, Microsoft tool selection methodology is based on the DFD relationships and, accordingly, associates each threat to a triple: *PointingNode*, *arch*, *PointedNode*. Sla-generator, instead, has three different ways to select the threats: (i) based on the component labels (i.e. specific parameters provided in the modelling phase), (ii) considering both the protocols used in each relationship and the role in the communication (e.g. if an asset is a client or a server), and (iii) extending the analysis to the threats that affect a component and can compromise also the neighboards.

4 Threat modelling tools

As already outlined, security experts are costly and human-driven threat modelling is costly both in terms of money and time. As a result of SLR, there are now a few tools that aim at offering support to experts in threat modelling, simplifying the work, requiring less experienced experts (most of the threats are catalogued), and offering solutions that produce the models in limited time. According to our analysis made in Section 3, most of the threat modelling tools use graph-based approaches to model the system. Therefore, this section provides a detailed analysis of the tools that rely on a graph-based model or can generate it, created by the user: *Microsoft threat modelling tool*, which is probably the most largely adopted one, the *OWASP Threat Dragon* and PyTM, supported by the OWASP consortium, and the *SLA-generator* tool, developed in the H2020 MUSA European project. SPARTA is also a promising tool, but it has not been selected for the following analysis because, using the catalogue offered by the official documentation, the tool provides threats using a stride-based approach: each STRIDE category is associated with the components of the DFD and to their relationships. This technique is not (yet) compatible with the analysis (i.e. it cannot be compared to the Microsoft and OWASP ones). In the following, we briefly outline the threat modelling approaches they support.

Table 6 Tools features

Tool	Modelling	Domain	Risk analysis	Threat selection methodology
sla-generator	Graphically: MACM	IoT, Edge, Cloud, 5G, Web	✓	Label-based Relationship-based Propagation based
Microsoft	Graphically: own DFD	Cyber-physical systems, Smart manufacturing systems, Edge, Web apps, Smart Energy systems, Automotive	×	Relationship-based
OWASP Threat Dragon	Graphically: DFD	Web apps	×	Label-based Relationship-based
SPARTA	Graphically: DFD	Web apps	✓	STRIDE-to-DFD based Relationship-based
STS-tool	Graphically: own graph-based model	Air traffic management, socio-technical systems	×	Rule-based Propagation-based
Threat-spec	Code: DFD	Not provided	×	Code annotations
PyTM	Code: DFD and sequence diagram	Web app	×	Label-based
Threat-agile	Code: extended DFD	Not provided	✓	Parameters-based

4.1 Microsoft threat modelling tool

The Microsoft threat modelling tool we tested was released in September 2018 (Microsoft, 2018). It aims at reducing threat modelling times, generating the threats to which a system is subjected automatically, relying on a model of the system. The system under analysis (SuA) is modelled by the user through a graph-based model. The user has the possibility to choose various stencils to be included in the application. Each node of the graph represents an application service, while each edge indicates a generic data flow (i.e. request or response). The Microsoft modelling technique requires that each node is characterized by two labels: component type and component value. The first one describes the type of the component while the second provides further functional information. Most of the pairs (*componenttype*, *componentvalue*) are shown in Table 7. For instance, a node can represent a generic database, so it can be modelled as a *generic data store* type and *database* value. The table does not represent all possible values for brevity's sake.

Once the application is modelled, the tool generates a threat report automatically. Threats are associated with each interaction between components. Each threat is selected from a proprietary catalogue taking into account the type of components involved in the interaction and the type of interaction. For example, *requests* made by a *web application* toward a *storage service* can generate the *SQL Injection* threat. In addition to providing threats associated with system assets, the tool suggests possible mitigations selected from a proprietary Microsoft database.

4.2 OWASP threat dragon

Threat Dragon is a free, open-source, cross-platform threat modelling application based on diagram models and rule engines to auto-generate threats and mitigations (Bhattacharya, 2020). It supports STRIDE (Ansari et al., 2019) classification and CIA. The tool was presented during the OWASP Open Security Summit in June 2020 by OWASP Lab Project and it is available as open-source code in Goodwin (Goodwin). The tool requires the application to be modelled through a graph-based model in which the nodes represent the components, while the edges

Table 7 Example values related to each component type

Component type	Component value
Generic data store	Azure Cosmos DB
	Azure Key Vault
	Azure Redis Cache
	Database
	Cache
Generic external interactor	Browser
	Dynamics CRM Mobile Client
	IoT Device
Generic process	Azure AD
	Azure ML
	Host
	Web Application
	IoT Cloud Gateway

Table 8 Parameters related to each component type

Component type	Parameters
Actor	Provide authentication
Process	Handle card payment
	Is a web application
	Handles goods and services
Store	Is a log
	Stores credentials
	Stores inventory
	Is encrypted
	Is signed
Data flow	Protocol
	Is encrypted
	Is over a public network

define the transfer of data between them. Each node can be (i) a generic running process, (ii) an actor, or (iii) a component that stores the data. Each element (node or edge) is characterized by a set of attributes that can be used to identify its security problems. All the parameters related to each element of the graph are described in Table 8.

The pair (*componentType*, *associatedParameters*) is used to obtain the threats associated with the component/flow (i.e. asset) of the diagram. For example, a store that has *is encrypted* as a parameter may be subject to the *vulnerable encryption algorithm* threat that could lead a malicious user to obtain data out of the application. The threats are obtained from the related catalogue (OWASP, OWASP) in a fully automatic way. The user can also define some custom threats and associate them with each element of the application. For each pair (*asset*, *threat*), the tool asks the user for the *threat status* (open or mitigated) field and a priority level (low, medium, high) and then suggest a list of possible mitigations.

4.3 SLA-generator

The SLA-generator threat modelling technique (Granata & Rak, 2021; Rak et al., 2022) relies on MACM, an expressive model that describes *WHAT* to assess and test. The MACM is a graph-based modelling technique in which each graph node represents a component of the system, and each edge characterizes the existing connection between two different components. MACM offers a simple way to synthesize an application architecture, focusing on its main components and relationships, enabling the security evaluation automation of the assessed systems. Nodes have a primary label, which identifies the asset class and may have a secondary label, which further specifies the primary class. Moreover, each node has a set of properties that better describe more specific aspects. A mandatory property is the *asset type*, which specifies the functional behaviour of the asset represented by the node. The allowed *asset types* for a node depend on the labels. *Labels* and supported *asset types* are listed and described in Table 9.

The possible relationships between the nodes are *uses*, *hosts*, *provides*, and *connects* and are described extensively in some works, cited above. In order to manage the MACM model, the tool represents them in a graph database, namely Neo4j. The MACM

Table 9 MACM node labels and assets

Primary label	Secondary label	Asset type(s)	Description
CSC		CSC.Human	A customer that uses services
CSP		CSP	A service provider like Amazon, Google, or a telecom provider
service	*IaaS*	VM, Container	Virtual machine or containers
service	*PaaS*	VM, Container	Virtual machine or containers
service	*SaaS*	Service.Web, Service.DB, Service.IOTGW, Service.MQTTBroker	Software (typically COTS) offered as a service
Network	WAN	Internet	A wide area network, typically the Internet
Network	LAN	Network.WiFi, Network.Wired	Network, the assets differs depending on the involved technologies
Network	PAN	Network.BLE, Network.ZigBee	Peronal area network, the assets differs depending on the involved technologies
HW		HW.server, HW.PC, HW.UE, HW.micro, HW.IOTDevice	A physical hosting hardware

Table 10 Threat catalogue template

Threat catalogue field	Description
Threat	A synthetic high-level label of the behaviour
Asset type	The asset typology to which the threat is subject
Relationship	Relation type
Protocol	Protocol used in the communication
Role in relationship	Role in communication
Behaviour	Detailed description of the threat
STRIDE	Stride classification (Ansari et al., 2019)
Compromised	Which assets the malicious behaviour compromises

is preliminarily produced by the user in Neo4j and then requested by the tool (available at link[2]) for the threat modelling phase. The tool communicates with the graph database, obtaining the correctly modelled applications. The technique selects all the threats applicable to the SuA by evaluating the *asset-type* field of each component (i.e. MACM node). The technique relies on a threat catalogue, which organizes the threats according to their asset type. The catalogue describes the threats with 8 parameters, as shown in Table 10.

A threat can be linked to an asset (asset type) or a communication protocol. For this reason, some fields may be left blank. For example, if a threat affects a specific asset typology, i.e. the *read DB configuration* threat for a *service.DB* asset type, both the relationship and role fields are left unspecified.

The *compromised* field indicates the asset that is compromised by the malicious behaviour and it can assume the following values:

- *Self*, if the threat compromises only the node specified by the asset type;
- *Source (relation)*, when it compromises the node pointing from the arch;
- *Target (relation)*, when it compromises the node pointed by the arch;

It is worth noting that when the *compromised* field is source or target, the argument *relation* can be *uses*, *connects*, or *hosts*. The tool then obtains the threats by considering both the asset-type field of the component and the related communication protocols used by the component. The tool also suggests, for each selected threat, one (or more) NIST SP-800-53 (Joint Task Force Interagency Working Group, Joint Task Force Interagency Working Group) controls.

4.4 PyTM

A Pythonic framework for threat modelling (PyTM) (pyTM, 2019) is a framework developed in Python by the OWASP consortium. The tool is not based on graph-model directly, but it implements a new paradigm: *threat modelling as a code*. Accordingly, the threat modeller should model the system using Python code and this should improve the usability of the threat modelling process by integrating this phase in the development. The model is produced leveraging a set of Python elements (i.e. *pytm Python module*).

[2] https://github.com/DanieleGranata94/SlaGenerator

Table 11 Threat parameters

Python element	Description
Generic element	A Python element whose function depends only on the specified properties.
External entity	It is where certain data comes from or goes to
Datastore	A data store represents the storage of persistent data required and/or produced by the process
Server	An entity processing data
Actor	User agent/browser
Process	Function where the transformation of data takes place
Dataflow	It is used to show the movement of data between the elements
Boundary	The boundary describes the limits of the system/sub-system
Lambda	A lambda function running in a function-as-a-service (FaaS) environment

Table 11 lists the supported elements and their brief description.

The tool also allows the user to associate some properties to each element of the model, but, we will not enumerate them for brevity's sake. Inspired by the infrastructure-as-a-code approach, the core idea of pyTM is that the target system architecture can be described through code, using as assets the elements outlined in Table 11. The model can be enriched through a set of properties associated to the assets as variables of the object.

In order to illustrate the idea, consider the following code: it describes the creation of a datastore, based on the CentOS operative system. The *datastore* is a relational database based on MySQL. Setting the *inScope* parameter to true, allow the tool to consider this database as in-scope of the threat model. Also, some controls can be applied in the modelling phase, as shown in the code example. In this case, the database has been analysed using a hardening process in order to find security vulnerabilities by applying recommended best practices.

```
db = Datastore("SQL Database")
db.OS = "CentOS"
db.type = DatastoreType.SQL
db.inScope = True
db.controls.isHardened = False
```

Once the model is built with all the data, the tool allows to:

- Generate a DFD from the code by parsing the Python model into a graph-based model;
- Generate a sequence diagram by considering the Python execution order.
- Produce a threat model

Since our scope is to analyse the threat modelling process, we focus on how threats are selected. The tool has its own threat database based on data provided by MITRE and CAPEC. A threat is described as a set of 7 parameters, as shown in Table 12.

It is worth noticing that the most crucial parameter is *Target* since it is used to select the threats for each element of the model. Custom threats (i.e. manually provided by the users/developers) are supported by leveraging the *overriding* function.

The tool considers as assets all the elements defined in Python; accordingly, both the DFD nodes and the data flows are the resources to protect and some threats are associated

Table 12 Threat parameters

Threat element field	Description
Id	Identifier of the threat
Description	An high-level description of the threat
Detail	A detailed description of the threat in natural language
Target	List of all the Python elements the threat applies to
Likelihood and impact	Low, medium, high parameters describing the probability and the severity of an attack that implements the threat
Prerequisites	Necessary condition in order to apply the threat
Condition	Boolean rule describing when the threat is applicable

with them. According to the tool technique, a threat compromises an element if the element type is the target list of the threat and the condition is verified. The condition is a logical condition that considers some Boolean parameters. As an example, a *Privilege Abuse* threat can be applied only if the Target value is **Server**, **Process**, or **Datastore** and the condition is *target.controls.hasAccessControl is False or target.controls.authorizesSource is False*. Accordingly, the threat is applicable only if the server does not have an access control system enabled *AND* an authorization control system.

An innovative function is the possibility to add controls to mitigate threats to the model. This allows the user to apply the countermeasures, produce a new model, and update the threat model (no longer considering the threats mitigated by the countermeasure). In this way, the tool supports the continuous threat modelling paradigm since the process is carried out on an ongoing basis rather than just being a one-time assessment.

5 Comparison

In this chapter, we want to compare the different threat modelling tools and the approaches they adopt. In order to show the differences, we will use a very common application, typically executed on a cloud infrastructure: an e-commerce site developed on top of WordPress. Considering this application, we modelled the system with the four different tools and documented the threat modelling results from each tool offered.

5.1 The WordPress case study

WordPress is an open-source content management system, which allows the creation and distribution of an Internet site made up of textual or multimedia content, which can be managed and updated dynamically. The web application WP is hosted on a cloud virtual machine on top of an Apache web server and interfaced with a MySQL database. In order to enable scalability, the WordPress component can be deployed multiple times, reusing always the same Database (that can scale only vertically, i.e. adding memory and/or CPU to the hosting VM). A Load Balancer distributes the Client requests to the connected WP instances. The developer simply customizes the WP instances, through custom plugins and customizing the application behaviour.

Even if the development of such systems is simple and commonly relies on very limited skills from the developer/system administrators, the application manages money and personal data, so it has strict security requirements. It must be considered that an incredible amount of WordPress instances on the web are vulnerable (see Abela, 2020), due to incorrect security planning and management.

5.2 Microsoft tool analysis

The Microsoft tool allowed us to describe the WordPress application in a complete way, as it supports a large number of stencils. As described above, the Microsoft tool considers the interactions between components (arcs of the graph) as assets and obtains security information by evaluating the type of the two components involved in the communication (Fig. 6).

In order to model the application, the client was modelled as a *Browser*, while WordPress and Load Balancer as a *web application*. The MySQL database instead was modelled as a *database* component value. Each service is running on a *host* node. Once the user has modelled the application, the tool automatically generates the threats for each asset (i.e. threat model) by producing a report in HTML format. Part of the threat model is described in Table 13.

It is important to note that the threat model shows, in this case, three values as asset field: *sourcenode*, *typeofrelationship*, *destinationnode*. From the results, it can be noted that, for example, each service exposes some threats in the relation to the *generic process* it hosts. As an example, a malicious user can get sensitive data from the service configuration files. A possible countermeasure that the tool suggests is to encrypt only the configuration files that contain sensitive data. The sending of the access credentials by the user to the service can also be compromised. In fact, a malicious user can steal this data in different ways. In order to reduce the risk that this threat happens, the Microsoft tool suggests some countermeasures. As an example, the user can disable the auto-complete HTML attribute in sensitive forms and inputs. The analysis also shows problems related to the use of weak encryption algorithms in the communication between the Load Balancer and WordPress. In fact, a malicious user can intercept the packets containing the encrypted data and apply an encryption reversing algorithm to recover the plain-text data.

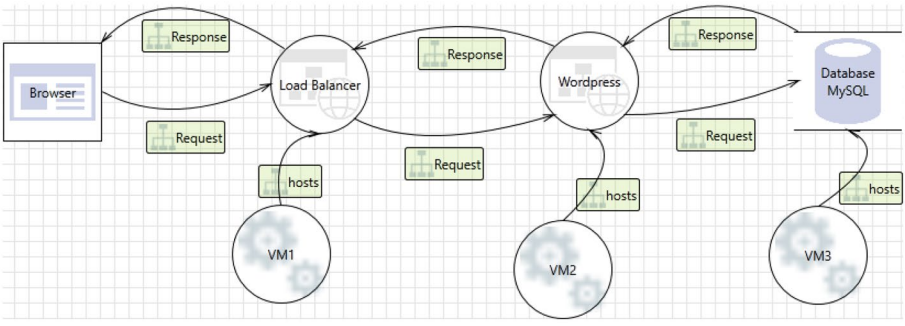


Fig. 6 Microsoft tool model WordPress

Table 13 Part of the threat model WordPress using Microsoft tool

Asset	Threat	STRIDE	Mitigation
VM-hosts-Service	An adversary can gain access to sensitive data stored in Web App's config files	Tampering	Encrypt sections of Web App's configuration files that contain sensitive data
Client-request-LoadBalancer	An adversary can steal sensitive data like user credentials	Spoofing	Explicitly disable the autocomplete HTML attribute in sensitive forms and inputs,...
LoadBalancer-request-WordPress	An adversary can reverse weakly encrypted or hashed content	Information Disclosure	Do not expose security details in error messages, Implement Default error handling page
WordPress-request-MySQL	An adversary can gain access to sensitive data by sniffing traffic to database	Information Disclosure	Ensure SQL server connection encryption and certificate validation

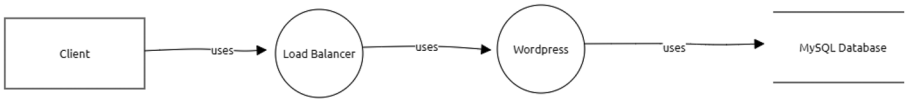


Fig. 7 Threat Dragon model WordPress

5.3 Dragon analysis

We modelled the system using the Threat Dragon diagram tool. The number of stencils available is limited, so, as shown in Fig. 7, The WordPress application was modelled using only the Process, Store, and Actor.

Load Balancer and WordPress were modelled as two processes, while for the Client and Mysql database we have chosen the stencil of Actor and Store respectively. Each node of the graph communicates through a *DataFlow* relationship. As highlighted in the previous section, the tool considers both the nodes and the arcs of the graph as assets (i.e. resources to be protected). Each asset has a set of properties aimed at selecting the related threats, as shown in Table 14.

We modelled the Load Balancer service and WordPress application as a *web application*. In particular, we assumed that the WordPress-based website is an e-commerce (manages payment cards) that stores data and encrypted credentials in a MySQL database. Each communication is made on a public network with HTTP protocol. Considering the selected parameters, the tool automatically collects threats (i.e. threat name, description, and STRIDE classification) for each component of the application and suggests the related mitigations. A partial list of threats for each component is shown in Table 15.

As the user can access from a public network, a malicious user can exploit a *fingerprinting* threat against the data exchange between the client and the Load Balancer, sending specific requests to obtain information in order to profile the application. The WordPress-based web application on the other hand can be subject to Card Cracking threats since it manages payment cards. In this case, the malicious user can carry out a brute force attack on the payment process in order to identify the missing values of the card (i.e. expiry date, security code, etc.). A brute force attack prevention system can (partially) mitigate the threat.

Table 14 Parameters related to each component type

Component	Selected parameters
Client	Provide authentication
Load Balancer	Web application
	Handles goods and services
WordPress	Web application
	Handles goods and services
	Handles card payment
MySQL database	Stores credentials
	Is a stores inventory
	Is encrypted
Each data flow	Protocol: http
	Is over a public network

Table 15 Part of the threat model WordPress using Dragon TM

Asset	Threat	Description	STRIDE	Mitigation
Client-Load Balancer	Fingerprinting	Specific requests are sent to the application eliciting information in order to profile the application	Information Disclosures	Defence includes restricting what information is provided, for example, version numbers and package details
	Use encryption	Unencrypted data sent over a public network may be intercepted and read by an attacker	Information Disclosures	Data should be encrypted either at the message or transport level
Load Balancer	Sniping	Automated exploitation of system latencies in the form of timing attacks	Elevation of Privileges	Anti-automation and prevention of abuse of functionality
WordPress	Denial of Service	Usage may resemble legitimate application usage but leads to exhaustion of resources	Elevation of Privileges	Providing backoff, resource management and avoiding forced deadlock
	Card Cracking	Brute force attack against application payment card process to identify the missing values	Information Disclosures	Interaction frequency, preventing brute force attacks and anti-automation
MySQL database	Account Creation	Bulk account creation, and sometimes profile population, by using the application's account signup processes	Elevation of Privileges	Interaction frequency, enforcement of a single unique action and enforcement of behavioural workflow

5.4 SLA-generator analysis

Figure 8 shows the MACM model of our case study. Each label affects the colour of the nodes, while attributes are not visible in the picture. As anticipated, the system is composed of a cloud service provider (e.g. Azure or a private cloud) that *provides* three virtual machines. which are labelled as *IaaS*, and their asset type is *VM*, e.g. virtual machine. One VM *hosts* a Load Balancer service while the other two VMs *host* respectively a WordPress instance and a MySQL a database instance. We modelled the Load Balancer (LB) and WordPress (WP) as *SaaS* nodes and we set their asset type as *web application*. The MySQL instance, instead, was labelled as a *SaaS*, but with *database* (DB) value as asset type. The LB *uses* the WP that, in turn, uses the DB. The Client (modelled as a *CSC* node) uses the Load Balancer service, which acts as the application interface. Each virtual machine is connected to the public network.

Applying our threat modelling technique, we produce a list of threats but, for simplicity's sake, we report in Table 16 just one for each asset type. The full list of threats is not compatible with the length of the paper.

The results show how nodes labelled as *SERVICE.Web* can be subject to *Injection* threat in which an attacker legitimately sends commands to the exposed service without proper authorization. In order to mitigate this threat, we suggest the usage of NIST Control SI-10, *Invalid input validation*.

The tool also models the threats associated with the network, such as *Message Reply* threat for which an attacker can re-transmit some packets (previously intercepted) in order to obtain data.

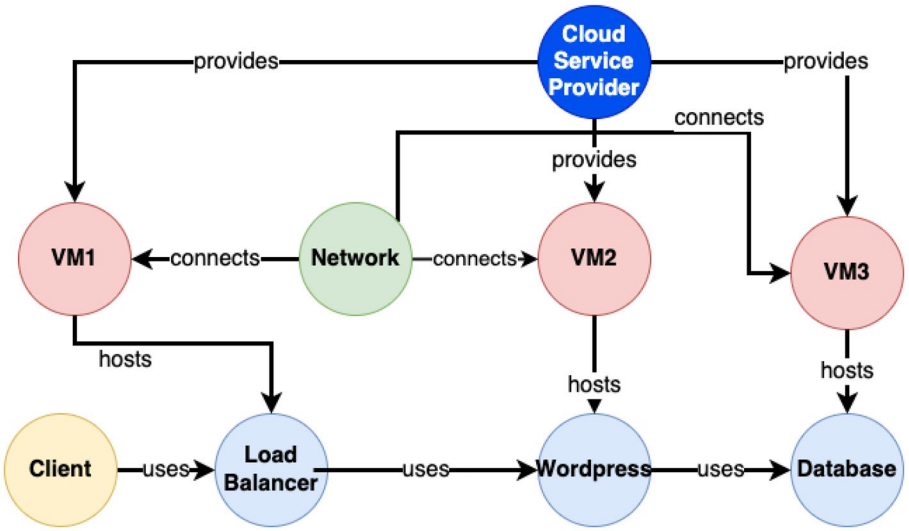


Fig. 8 WordPress MACM

Table 16 Part of the threat model WordPress using SLA-generator

Asset	Asset type	Threat	Description	STRIDE	NIST control
WordPress	SERVICE.Web	Injection	The attacker's hostile data can trick the interpreter into executing unintended commands	Tampering	SI-10 Invalid input validation
MySQL database	SERVICE.DB	Remote DoS	Made the DBMS inaccessible to remote clients	Denial of Service	SC-5, DoS Protection
VMs	SERVICE.VM	Authorization Abuse	An adversary is able to circumvent the authorization controls	Elevation of Privileges	CA-6 Authorization
Network	Network	Message Reply	An adversary can re-transmit the content of the packets coming from the asset at a later time	Spoofing	AC-12, Session Termination

5.5 PyTM analysis

We modelled the system using the Python code and the related documentation. Since the number of elements is limited, we used only two *Server* to model WordPress and the Load Balancer, the *Actor* to model the browser, and the *Datastore* to model the MySQL database. The DFD produced by the Python code is shown in Fig. 9.

The user also has modelled all the requests and the responses through the *Data-Flow* element. There is also the possibility to attach some data in the data flow (e.g. tokens, credentials), but, at the moment, the tool does not consider them to evaluate the threats. The tool produced a report containing 91 threats, each one associated with the compromised asset. A part of the threat model is shown as an example in Table 17.

From the analysis of the table, we can notice that the threats are described at a low level as well as the suggested mitigation. There are also some technological references that make the threat model less applicable (not generic). For example, an attacker can include some code into a file using PHP (assuming installed on the server). Also, the database can be subjected to brute force attack if it uses some weak encryption algorithm for the data.

5.6 Comparison

It is worth noting that, as highlighted above, most of the tools rely on a graph-based model to describe the target system, where the node represents the asset and the edge of their connections. Just PyTM relies on a code-based approach that allows the user to leverage a Python class to model the system, but it can produce a DFD from the code. Accordingly, a comparison can be carried out by considering the edges and the nodes produced by the code. The tools differ in the interpretation and metadata associated with nodes and edges of the graph. According to Microsoft's approach, there is a large variety of possible nodes, but the key role in the threat modelling is associated with the connection among them: in fact the threats are listed *per-connection*, taking into account the connected nodes and the connection attributes. According to OWASP, on the other hand, the Threat Dragon tool evaluates both the nodes and the arcs of the graph as assets, associating the threats to each element. However, the types of nodes and edges are very limited and the threats are selected according to a few attributes associated with both nodes and relationship. The SLA-generator, instead, focuses on system assets (the graph nodes) and identifies the possible threats relying on the *asset type* attribute, which offers a large variety of different values, similar to the Microsoft threat modelling tool. Moreover, relationships affect the possible threats to which each

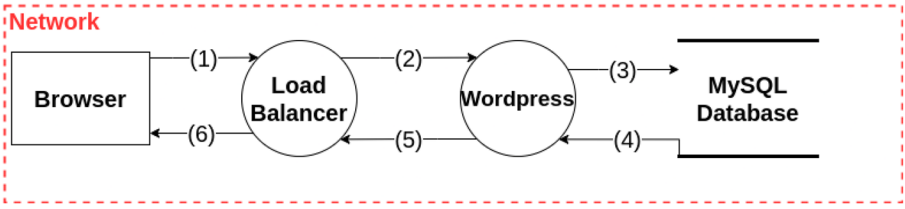


Fig. 9 PyTM DFD

Table 17 Part of the threat model WordPress using PyTM

Element	Element type	Threat	Description	Mitigation
WordPress	Server	INP03 Server Side Include (SSI) Injection	An attacker can use SSI Injection to send code to a web application that then gets executed by the web server.	Set the OPTIONS IncludesNOEXEC in the global web server configuration file to deny
Load Balancer	Server	INP16 — PHP Remote File Inclusion	In this pattern the adversary is able to load and execute arbitrary code remotely available from the application.	Perform input validation for all remote content, including remote and user-generated content Implementation.
MySQL database	Datastore	CR05 — Encryption Brute Forcing	An attacker can perform an exhaustive search on the key space to determine the key that decrypts the ciphertext to obtain the plaintext.	Use commonly accepted algorithms and recommended key sizes
Each data flow	Dataflow	DE01 — Interception	An adversary monitors data streams to or from the target for information gathering purposes	Leverage encryption to encode the transmission of data thus making it accessible only to authorized parties.

Table 18 Threat modelling comparison table

Asset	SLA-generator threat	Microsoft threat	OWASP threat	PyTM
WordPress	Data Leakage	Read web app's config files	Fingerprinting	Data leak
		Steal sensitive data like user credentials	Carding	
			Card Cracking	
WordPress	Injection	SQL injection through Web App	-	Many types of injection
Database	Read Injection	SQL injection	Account Creation	SQL Injection
	Insert Injection			
VM	Data Breach	Access to sensitive data from log files	-	-
VM	Denial of Service	-	-	-

node, but the threats are always listed as associated with nodes. A different approach is adopted by PyTM that considers all the Python elements as assets (even the data flows).

It is out of the scope of this work to say which approach is better (we aim at comparing the ideas not at making a rank of the tools), but it is worth noting that in the graph they made a completely different choice: one focuses on edge, one on nodes and the last on both of them. However, the final result, in all the cases, is a list of threats that contains an explicit description of the malicious behaviour (in natural language) and the classification of the threat according to STRIDE or respect to the impact on Confidentiality, Integrity, and Availability. The tools, even in the case of the WordPress application, which is pretty simple, produce a pretty long list of threats (88 for the MS threat modelling tool, 84 for SLA-generator, 31 for the Dragon tool, and 97 for PyTM). We, acting as experts, consider that the choice of listing threats only respects assets or only respects relationships (the choices done by SLA-generator and by MS threat modelling Tool) helps the expert work in the analysis of the results, but this is and remains a subjective choice. However, the number of threats outlined by the OWASP tool looks at the state of the art, with limited respect to the other tools. This is due to the limited set of parameters available for the selection and, probably, to the underlying threat catalogue dimension. PyTM instead has the greatest number of threats, but, as already mentioned, the threats are high-level attacks from CAPEC (i.e. attack patterns and in some cases, they depend on the technological implementation of the asset). Another interesting aspect is that the four techniques present threats at different levels of granularity, as shown in Table 18.

As an example, the SLA-generator tool underlines how WordPress can be subject to *data leakage*. The same threat is (partially) expressed by the Microsoft tool with *read configuration files* and *steal user credentials* threats and fully expressed by PyTM. According to OWASP, instead, data loss can be caused both by an application profiling technique (e.g. fingerprinting) and by techniques that aim at obtaining information on users' virtual cards.

In general, the threats affecting WordPress were 10 for both OWASP and SLA-generator, 25 according to Microsoft, and 42 for PyTM. It is important to note, however, that threats are expressed with different levels of detail. The analysis also shows how a *Injection* threat can affect both WordPress and the database. Considering the database

Table 19 Tool comparison features

Features	SLA-generator	Microsoft	OWASP	PyTM
Threats	84	88	30	97
Asset	Nodes	Archs	Both	All Python elements
Use case	Generic	Suitable for architecture using Microsoft components	Generic	Generic
Threat granularity	Variable	High	Low	High

as an asset, a Microsoft SQL injection can be as *SLA-generator Read/Insert injection* that takes into account that a malicious user wants to get information from the database or write to it (e.g. create an account), while PyTM describes many ways to inject data through more detailed threats. In this case, the threats according to the SLA-generator tool are 15, while OWASP and Microsoft consider only 8 and 12 for PyTM. Virtual machines, on the other hand, are not considered in the OWASP model and PyTM code, the table shows the comparison only between SLA-generator and Microsoft tool. One of the 13 threats described by the SLA-generator is that of Data Breach, partially mapped with *Access to sensitive data from log files* by Microsoft (which instead considers 6 threats). Network assets were modelled only by the SLA-generator and threat modelling reported 12 threats.[3] As a result, a generic tool comparison, shown in Table 19, summarizes the main differences between the selected tools.

6 Conclusions and future work

In this paper, we have analysed the state of the art of automated threat modelling techniques using a systematic literature review technique. According to the review results, we focused on three different graph-based methodologies and their tools: SLA-generator, Microsoft tool, and Threat Dragon by OWASP. The analysed tools require a very simplified graph-based model of the application in which the nodes represent the components of the system and the arcs represent the interactions between the various components. The simplicity of modelling allows the user in all three approaches to obtain security information in a fully automatic way. The approaches were applied to a case study involving WordPress, a content management system that allows you to manage a website. The results show that the threats are described at different levels of detail, but still compatible. In particular, OWASP Threat Dragon has proved to be the tool that produces a less complete threat model than the others. The number of threats related to the WordPress component was greater (25) with the Microsoft tool, while the threat model related to the database and virtual machines was more complete with SLA-generator. Furthermore, the tool also considered the network as an asset, highlighting 12 threats. As mentioned in the paper, risk analysis builds on threat modelling by assessing the likelihood of a threat occurring and the potential consequences. Accordingly, our future work is performing a complementary systematic literature review on risk analysis methodologies in the literature and the tools used to implement them.

[3] Full threat modelling comparison is available on request.

Author contribution Daniele Granata performed the systematic literature review and the tool analysis, wrote the manuscript text, and prepared all the figures; Massimiliano Rak contributed to review planning and validated the results, contributed to tools analysis, defined the paper structure, and reviewed the manuscript; all the authors contributed to the development of the methodology.

Funding Open access funding provided by Università degli Studi della Campania Luigi Vanvitelli within the CRUI-CARE Agreement. This work was partially supported by project SSeCeGOV funded by University of Campania under program Valere 2020.

Data availability The datasets generated during and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest All authors declare no competing interests.

References

Abela, R. (2020). Statistics show why WordPress is a popular hacker target. https://www.wpwhitesecurity.com/statistics-70-percent-wordpress-installations-vulnerable/

AbuEmera, E. A., ElZouka, H. A., & Saad, A. A. (2022). Security framework for identifying threats in smart manufacturing systems using stride approach. In: *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)* (pp. 605–612). https://doi.org/10.1109/ICCECE54139.2022.9712770

Althar, R. R., Samanta, D., Kaur, M., Singh, D., & Lee, H.-N. (2022). Automated risk management based software security vulnerabilities management. *IEEE Access, 10*, 90597–90608. https://doi.org/10.1109/ACCESS.2022.3185069

Almubairik, N. A., & Wills, G. (2016). Automated penetration testing based on a threat model. In: *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)* (pp. 413–414). https://doi.org/10.1109/ICITST.2016.7856742

Alwaheidi, M. K. S., & Islam, S. (2022). Data-driven threat analysis for ensuring security in cloud enabled systems. *Sensors, 22*(15). https://doi.org/10.3390/s22155726

Ansari, M. T. J., Pandey, D., & Alenezi, M. (2022). STORE: security threat oriented requirements engineering methodology. *Journal of King Saud University-Computer and Information Sciences, 34*(2), 191–203.

Asif, M. R. A., Hasan, K. F., Islam, M. Z., & Khondoker, R. (2021). STRIDE-based cyber security threat modeling for IoT-enabled precision agriculture systems. In: *2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI)* (pp. 1–6). https://doi.org/10.1109/STI53101.2021.9732597

Bernsmed, K., Cruzes, D., Jaatun, M., & Iovan, M. (2021). Adopting threat modelling in agile software development projects. *Journal of Systems and Software, 183*, 111090. https://doi.org/10.1016/j.jss.2021.111090

Bhattacharya, D. (2020). OWASP threat dragon review.

Brown, S., Fox, S., Hewage, C., & Khan, I. (2022). Threat modelling of cyber physical systems: A real case study based on window cleaning business. *SN Computer Science, 3*. https://doi.org/10.1007/s42979-022-01021-3

Casola, V., Benedictis, A. D., Mazzocca, C., & Montanari, R. (2021). Toward automated threat modeling of edge computing systems. In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)* (pp. 135–140). https://doi.org/10.1109/CSR51186.2021.9527937

Casola, V., De Benedictis, A., Rak, M., & Villano, U. (2019). Toward the automation of threat modeling and risk assessment in IoT systems. *Internet of Things, 7*.

Chen, H. (2019). Determining information security threats for an iot-based energy internet by adopting software engineering and risk management approaches. *Inventions, 4*, 53. https://doi.org/10.3390/inventions4030053

Chen, Y. -T. (2018). Modeling information security threats for smart grid applications by using software engineering and risk management. In: *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)* (pp. 128–132). https://doi.org/10.1109/SEGE.2018.8499431

Curzi, S. (2020). Threat Manager Studio. https://threatsmanager.com/

Danielis, P., Beckmann, M., & Skodzik, J. (2020). An ISO-compliant test procedure for technical risk analyses of IoT systems based on STRIDE. In: *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 499–504). https://doi.org/10.1109/COMPSAC48688.2020.0-203

DeMarco, T. (1979). Structured analysis and system specification. Prentice Hall PTR, USA.

Ding, J., Atif, Y., Andler, S., Lindström, B., & Jeusfeld, M. (2017). CPS-based threat modeling for critical infrastructure protection. *ACM SIGMETRICS Performance Evaluation Review, 45*, 129–132. https://doi.org/10.1145/3152042.3152080

Dominic, D., Chhawri, S., Eustice, R., Ma, D., & Weimerskirch, A. (2016). *Risk assessment for cooperative automated driving* (pp. 47–58). https://doi.org/10.1145/2994487.2994499

Fabbri, S., Silva, C., Hernandes, E., Octaviano, F., Di Thommazo, A., & Belgamo, A. (2016). Improvements in the start tool to better support the systematic review process. In: *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering. EASE '16*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2915970.2916013

Faily, S. (2018). *Designing usable and secure software with IRIS and CAIRIS*. Springer Cham - Computer Science.

Ficco, M., Granata, D., Rak, M., & Salzillo, G. (2021). Threat modeling of edge-based IoT applications. In: *International Conference on the Quality of Information and Communications Technology* (pp. 282–296). Springer.

Fraser Scott, M. R., & Smotrakov, A. (2019). Threat Spec. https://threatspec.org/

Frydman, M., Ruiz, G., Heymann, E., César, E., & Miller, B. P. (2014). Automating risk analysis of software design models. *The Scientific World Journal, 2014*.

Goodwin, M. (2020). OWASP Threat Dragon. Retrieved October 28, 2022, from https://github.com/owasp/threat-dragon/releases

Granata, D., & Rak., M. (2021). Design and development of a technique for the automation of the risk analysis process in IT security. In: *Proceedings of the 11th International Conference on Cloud Computing and Services Science - CLOSER* (pp. 87–98). SciTePress. https://doi.org/10.5220/0010455200870098.INSTICC

Granata, D., Rak, M., & Salzillo, G. (2022). Automated threat modeling approaches: Comparison of open source tools. In A. Vallecillo, J. Visser, & R. Pérez-Castillo (Eds.), *Quality of Information and Communications Technology* (pp. 250–265). Cham: Springer.

Granata, D., Rak, M., Salzillo, G., & Barbato, U. (2021). Security in IoT pairing & authentication protocols, a threat model, a case study analysis. *2490*, 207–218. CEUR-WS.

Haitao, Z., Lei, L., Ruikun, L., Jiajia, Y., Yun, L., & Lirong, C. (2022). Research and application of intelligent vehicle cybersecurity threat model. In: *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)* (pp. 102–109). https://doi.org/10.1109/DSC55868.2022.00021

Haji, S., Tan, Q., & Costa, R. (2019). A hybrid model for information security risk assessment. *International Journal of Advanced Trends in Computer Science and Engineering, 8*, 100–106. https://doi.org/10.30534/ijatcse/2019/1981.12019

Hasan, R., & Hasan, R. (2021). Towards a threat model and security analysis of video conferencing systems. In: *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)* (pp. 1–4). https://doi.org/10.1109/CCNC49032.2021.9369505

Hoque, M. A., & Hasan, R. (2019). *Towards a threat model for vehicular fog computing* (pp. 1051–1057).

Hussain, S., Kamal, A., Ahmad, S., Rasool, G., & Iqbal, S. (2014). Threat modelling methodologies: A survey. *26*, 1607–1609.

Ivanova, N. D., & Ivanenko, V. G. (2022). Modeling advanced persistent threats using risk matrix methods. *Journal of Computer Virology and Hacking Techniques,* 1–6.

Jamil, A. -M., Khan, S., Lee, J. K., & Ben Othmane, L. (2021). Towards automated threat modeling of cyber-physical systems. In: *2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)* (pp. 614–619). https://doi.org/10.1109/ICSECS52883.2021.00118

Joint Task Force Interagency Working Group. (2020, September). Security and privacy controls for information systems and organizations. Technical report, National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-53r5. Edition: Revision 5.

Khan, R., McLaughlin, K., Laverty, D., & Sezer, S. (2017). Stride-based threat modeling for cyber-physical systems. In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)* (pp. 1–6). https://doi.org/10.1109/ISGTEurope.2017.8260283

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and Software Technology, 51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009. Special Section - Most Cited Articles in 2002 and Regular Research Papers.

Kornecki, A. J., & Janusz, Z. (2015). Threat modeling for aviation computer security. *Crosstalk, 21*.

Kosachenko, T., Dudkin, D., Konev, A., & Sharamok, A. (2021). Threat model for trusted sensory information collection and processing platform. In P. K. Singh, G. Veselov, A. Pljonkin, Y. Kumar, M. Paprzycki, & Y. Zachinyaev (Eds.), *Futuristic trends in network and communication technologies* (pp. 296–304). Singapore: Springer.

Leander, B., Čaušević, A., & Hansson, H. (2019). Cybersecurity challenges in large industrial IoT systems. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (pp. 1035–1042). https://doi.org/10.1109/ETFA.2019.8869162

LINDDUN. (2020). LINDDUN privacy engineering. Retrieved October 28, 2022, from https://www.linddun.org/

Mahak, M., & Singh, Y. (2021). Threat modelling and risk assessment in internet of things: A review. In: P. K. Singh, S. T. Wierzchoń, S. Tanwar, M. Ganzha, & J. J. P. C. Rodrigues (Eds.), *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security* (pp. 293–305). Singapore: Springer.

Mahmood, S., Nguyen, H. N., & Shaikh, S. A. (2022). Systematic threat assessment and security testing of automotive over-the-air (OTA) updates. *Vehicular Communications, 35*, 100468. https://doi.org/10.1016/j.vehcom.2022.100468

Mani, P., & Venkat, M. (2017). A risk-centric defensive architecture for threat modeling in e-government application. *Electronic Government, an International Journal, 14*, 1. https://doi.org/10.1504/EG.2017.10008841

Martins, G., Bhatia, S., Koutsoukos, X., Stouffer, K., Tang, C., & Candell, R. (2015). Towards a systematic threat modeling approach for cyber-physical systems. In: *2015 Resilience Week (RWS)* (pp. 1–6). https://doi.org/10.1109/RWEEK.2015.7287428

Meland, P. H., Paja, E., Gjære, E. A., Paul, S., Dalpiaz, F., & Giorgini, P. (2014). Threat analysis in goal-oriented security requirements modelling. *International Journal of Secure Software Engineering, 5*, 1–19. https://doi.org/10.4018/ijsse.2014040101

Messe, N., Chiprianov, V., Belloir, N., El-Hachem, J., Fleurquin, R., & Sadou, S. (2020). Asset-oriented threat modeling. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 491–501). https://doi.org/10.1109/TrustCom50675.2020.00073

Microsoft. (2018). Microsoft threat modeling tool. Microsoft.

Monteuuis, J. -P., Boudguiga, A., Zhang, J., Labiod, H., Servel, A., & Urien, P. (2018). *SARA: Security automotive risk analysis method* (pp. 3–14). https://doi.org/10.1145/3198458.3198465

Moreira, A., Amaral, V., & De Faveri, C. (2016). Goal-driven deception tactics design. In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 264–275). https://doi.org/10.1109/ISSRE.2016.44

Naagas, M. (2018). A threat-driven approach to modeling a campus network security. https://doi.org/10.1145/3193092.3193096

Nweke, L., & Wolthusen, S. (2020). A review of asset-centric threat modelling approaches. *International Journal of Advanced Computer Science and Applications*, *11*, 1–6. https://doi.org/10.14569/IJACSA.2020.0110201

Omotunde, H., & Ibrahim, R. (2015). A review of threat modelling and its hybrid approaches to software security testing.

OWASP. OWASP automated threats to web applications.

pyTM. (2019). https://github.com/izar/pytm

Ramazanzadeh, M., Barzegar, B., & Motameni, H. (2022). ASATM: Automated security assistant of threat models in intelligent transportation systems. *IET Computers Digital Techniques, 16*. https://doi.org/10.1049/cdt2.12045

Radoglou-Grammatikis, P., Rompolos, K., Sarigiannidis, P., Argyriou, V., Lagkas, T., Sarigiannidis, A., Goudos, S., & Wan, S. (2022). Modeling, detecting, and mitigating threats against industrial healthcare systems: A combined software defined networking and reinforcement learning approach. *IEEE Transactions on Industrial Informatics, 18*(3), 2041–2052. https://doi.org/10.1109/TII.2021.3093905

Rak, M., Casola, V., De Benedictis, A., & Umberto, V. (2019). Automated risk analysis for IoT systems. In: *Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2018)* (pp. 265–275). https://doi.org/10.1007/978-3-030-02607-3_24

Rak, M., Salzillo, G., & Granata, D. (2022). ESSecA: An automated expert system for threat modelling and penetration testing for IoT ecosystems. *Computers and Electrical Engineering, 99*, 107721. https://doi.org/10.1016/j.compeleceng.2022.107721

Rak, M., Salzillo, G., & Romeo, C. (2020). *Systematic IoT penetration testing: Alexa case study, 2597*, 190–200. CEUR-WS.

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *Unified modeling language reference manual, The (2nd Edition)*. Pearson Higher Education.

Saatkamp, K., Krieger, C., Leymann, F., Sudendorf, J., & Wurster, M. (2019). Application threat modeling and automated VNF selection for mitigation using TOSCA. In: *2019 International Conference on Networked Systems (NetSys)* (pp. 1–6). https://doi.org/10.1109/NetSys.2019.8854524

Salzillo, G., Rak, M., & Moretta, F. (2020). Threat modeling based penetration testing: The open energy monitor case study. In: *13th International Conference on Security of Information and Networks. SIN 2020*. Association for Computing Machinery, New York, NY, USA.

Schaad, A., & Borozdin, M. (2012). TAM2: Automated threat analysis. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (pp. 1103–1108). Association for Computing Machinery.

Schlegel, R., Obermeier, S., & Schneider, J. (2015). Structured system threat modeling and mitigation analysis for industrial automation systems. In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)* (pp. 197–203). https://doi.org/10.1109/INDIN.2015.7281734

Shi, Z., Graffi, K., Starobinski, D., & Matyunin, N. (2021). Threat modeling tools: A taxonomy. *IEEE Security & Privacy, 20*(4), 29–39.

Shelupanov, A., & Konev, A. (2019). Threat model for IoT systems on the example of openUNB protocol. *International Journal of Emerging Trends in Engineering Research, 7*, 283–290. https://doi.org/10.30534/ijeter/2019/11792019

Sion, L., Landuyt, D., Yskout, K., & Joosen, W. (2018). *Sparta: Security & privacy architecture through risk-driven threat assessment* (pp. 89–92). https://doi.org/10.1109/ICSA-C.2018.00032

Sion, L., Van Landuyt, D., Yskout, K., Verreydt, S., & Joosen, W. (2021). Automated threat analysis and management in a continuous integration pipeline. In: *2021 IEEE Secure Development Conference (SecDev)* (pp. 30–37). https://doi.org/10.1109/SecDev51306.2021.00021

Sion, L., Wuyts, K., Yskout, K., Van Landuyt, D., & Joosen, W. (2018). Interaction-based privacy threat elicitation. In: *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS & PW)* (pp. 79–86). https://doi.org/10.1109/EuroSPW.2018.00017

Tan, K., & Garg, V. (2022). An analysis of open-source automated threat modeling tools and their extensibility from security into privacy.

Tatam, M., Shanmugam, B., Azam, S., & Kannoorpatti, K. (2021). A review of threat modelling approaches for APT-style attacks. *Heliyon, 7*(1). https://doi.org/10.1016/j.heliyon.2021.e05969

Threat Agile. (2020). Retrieved October 28, 2022, from https://github.com/Threagile/threagile

Valenza, F., Karafili, E., Steiner, R. V., & Lupu, E. C. (2022). A hybrid threat model for smart systems. *IEEE Transactions on Dependable and Secure Computing*, 1–14. https://doi.org/10.1109/TDSC.2022.3213577

Vallant, H., Stojanovic, B., Božić, J., & Hofer-Schmitz, K. (2021). Threat modelling and beyond-novel approaches to cyber secure the smart energy system. *Applied Sciences, 11*, 5149. https://doi.org/10.3390/app11115149

Verreydt, S., Sion, L., Yskout, K., & Joosen, W. (2022). Relationship-based threat modeling. In: *2022 IEEE/ACM 3rd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)* (pp. 41–48). https://doi.org/10.1145/3524489.3527303

Von Der Assen, J., Franco, M.F., Killer, C., Scheid, E.J., & Stiller, B. (2022). CoReTM: An approach enabling cross-functional collaborative threat modeling. In: *2022 IEEE International Conference on Cyber Security and Resilience (CSR)* (pp. 189–196). https://doi.org/10.1109/CSR54599.2022.9850283

Wirtz, R., & Heisel, M. (2020). *Risk identification: From requirements to threat models* (pp. 385–396). https://doi.org/10.5220/0008935803850396

Wuyts, K., Sion, L., & Joosen, W. (2020). Linddun go: A lightweight approach to privacy threat modeling. In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS &PW)* (pp. 302–309). https://doi.org/10.1109/EuroSPW51379.2020.00047

Wuyts, K., Van Landuyt, D., Hovsepyan, A., Joosen, W. (2018). Effective and efficient privacy threat modeling through domain refinements. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing. SAC '18* (pp. 1175–1178). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3167132.3167414

Zeinali, M., & Hadavi, M. A. (2018). Threat extraction method based on uml software description. In: *2018 15th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)* (pp. 1–8). https://doi.org/10.1109/ISCISC.2018.8546868