Automated Threat Modeling Approaches: Comparison of Open Source Tools

Daniele Granata(✉), Massimiliano Rak, and Giovanni Salzillo

Department of Engineering, University of Campania Luigi Vanvitelli, via Roma 29, 81031 Aversa, CE, Italy
{daniele.granata,massimiliano.rak,giovanni.salzillo}@unicampania.it

Abstract. The software systems of modern architectures are characterized by high heterogeneity and by the use of a model that delegates the control of individual components to third parties, making these systems more vulnerable to cyber-attacks. As a consequence, best practices, such as the Security-by-Design development methodologies, suggest taking into account security all over the systems life cycle, starting from the very early stages (e.g. from initial requirement analysis). Thus, one of the most relevant practices is Threat Modeling (TM), i.e. the activity devoted to identifying the possible threats that may affect the system. According to most security-related best practices, TM should be done as early as possible, in order to help in the requirement elicitation. Threat Modeling is a complex activity, that requires security experts with consolidated skills, able to predict and anticipate the possible issues: as a consequence, it is a costly activity, both in terms of time and money. Due to the continuous need of enforcing security, the effect of new regulation and the wide diffusion of ICT systems, there is a recent growth of tools and techniques that support and aims at automatizing Threat modelling activities. This work illustrates the approach adopted by our research team and compares the results of our technique with two other existing tools, in order to offer a brief overview of the state of the art of threat modelling automation techniques and of state of art limits and open research topics. It is worth noting that our comparison does not aims at being complete and focuses only on open tools (or on their free/community version), but offers a basis for understanding the progress of security automation processes in terms of threat modelling.

Keywords: Security · Threat modelling · Security assessment

1 Introduction

Nowadays, Software systems are more and more complex and heterogeneous. Due to the widespread use of such IT solutions, the process of ensuring their security has become a strong requirement, as evidenced by the new regulations (e.g. GDPR, Cybersecurity Act) that impose hard privacy and security requirements. However, it is not easy to take into account security in application development and the problem grows up in emerging paradigms, like the Cloud, that

A. Vallecillo et al. (Eds.): QUATIC 2022, CCIS 1621, pp. 250–265, 2022.
https://doi.org/10.1007/978-3-031-14179-9_17

delegate resources and services to third parties. Cloud-native applications, as an example, often rely on micro-services architectures and/or on the integration of Commercial-Off-The-Shelf (COTS) components, an approach that has great advantages in terms of costs and time-to-market, but heavily affects the security. In order to address the security, best practices suggest the adoption of threat modeling 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 software life cycle. The adoption of these procedures increases awareness of cybersecurity issues in all the involved personnel and allows the security administrator to evaluate and accordingly manage the risk, applying mitigation strategies. However, at the state of the art, there is a lack of standards and consolidated practices devoted to helping security administrators systematically apply security checks and mitigate threats [22]. As a consequence, there are many tools that offer support to threat modeling, but each relies on (i) different modeling approaches and (ii) describing threats and threat models in different ways. This paper aims at offering a simple description of the main open source tools, describing the threat modeling approaches on which the tools rely and the results they are able to produce. In particular, this paper focuses on three different tools, that adopt different modeling techniques: Threat Modeling tool by Microsoft [1], Threat Dragon by OWASP [11] and Sla-Generator [12] which implements the methodology proposed in MUSA H2020 European research project. The comparison was carried out on a simple, but significant case study, involving a well-known Content Management System (CMS) platform: Wordpress. The remainder of the paper is organized as follows: Section 2 briefly summarizes the threat modeling techniques proposed in literature, while Sect. 3 describes the three tools that we addressed in our analysis. Section 4 compares the tools, using the Wordpress application as a basis for the comparison. Finally, Sect. 5 summarizes our conclusions and future work.

2 Threat Modeling Practices

Threat modelling processes give an organized representation of all the information that influences the security of an application and it can be related to a wide run of things, as stated by OWASP [9], *threat modeling works to identify, communicate, and understand threats and mitigations within the context of protecting something of value.* As highlighted by the OWASP threat modelling manifesto [17], threat modeling aims at achieving thoroughness and reproducibility by applying security and privacy knowledge in a structured manner. It is also supported by some tools that allow you to enable repeatability and provide measurability. The use of threat modelling provides a structured representation of all the information that affects the security of an application and it can be applied to a wide range of things, including software, applications, systems, networks, distributed systems, Internet of Things (IoT) devices, and business processes.

At the state of the art, there are various methodologies that aim at modelling the threats applicable to a system, one of the related problems is that

these practices are (often) carried out by a human and require a lot of execution time [21]. In order to reduce the time of these practices, there are, at the state of the art, some automated (or semi-automated) approaches: Schaad et al. [19] proposed a STRIDE-based threat modelling technique for software architecture diagrams. They introduced their own conceptual data model, consisting of assets, asset shapes and components. These concepts can be used to describe software systems and perform security evaluations. Additionally, they implemented a supporting tool, TAM, that performs an automated threat analysis, based on the described assets. Casola et al. [6–8] proposed a Security-by-Design methodology to evaluate the security of IoT systems by the means of an almost automated process for threat modeling and risk assessment. Their approach also helps at identifying the security controls to implement in order to mitigate the existing security risks. We invite the interested reader to deepen the existing automated [10,14,15] and semi-automated [4,20] threat modeling approaches. As a summary, the state of the art highlights, as can be seen especially from the literature review in [22], that there is a wide need for automating the full process, leaving to humans only the role of final control result and evaluation.

3 Threat Modeling Tools

As outlined in the previous section, even if threat modeling is a well accepted practice, it commonly relies on security expert skills and experience. As a consequence, there are no standards that aim at listing possible threats. As an example, the standard ISO 15408, Common Criteria, imposes a structured section for security threats and security objectives (the countermeasures to address the threats), but, while security requirements are catalogued in long documents, threats should be defined case-by-case in the standard documents (security profiles and/or security target) by the expert for the specific product or class of products. However, Security experts are costly and the human-driven threat modeling is costly both in terms of money and time. Accordingly, there are now a few tools that aim at offering support to experts in threat modeling, simplifying the work, requiring less experienced experts (most of the threats are catalogued) and offering solution that produce the models in limited time. According to our studies, three of this tools are the most interesting ones: *Microsoft Threat Modeling Tool*, which is probably the most largely adopted one, the *OWASP Threat Dragon*, supported by the OWASP consortium, and the *SLAgenerator* tool, developed in the H2020 MUSA European project. In the following we briefly outline the threat modeling approaches they support.

3.1 Microsoft Threat Modeling Tool

Microsoft Threat Modeling Tool we tested was released in September 2018 [1]. 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 modeled 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 Modeling 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 1. 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.

Table 1. 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 Gatewat

Once the application is modeled, the tool generates a threat report automatically. Threats are associated with each interaction between components. Each threat is selected from a proprietary catalog 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.

3.2 OWASP Threat Dragon

Threat Dragon is a free, open-source, cross platform threat modelling application based on diagram models and rule engine to auto-generate threats and mitigations [5]. It supports STRIDE [3] 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 [11]. The tool requires the application to be modelled through a graph-based model in which the nodes represent the components, while the edges 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 the Table 2.

Table 2. 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

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 catalog [16] 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.

3.3 SLAGenerator

The SLAGenerator threat modelling technique [12,18] relies on MACM (Multi-purpose Application Composition Model) 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 depends on the labels. *Labels* and supported *Asset Types* are listed and described in Table 3.

Table 3. 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	Personal 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

The possible relationships between the nodes are *uses, hosts, provides, connects*, described extensively in some works, cited above. In order to manage the MACM model the tool represent them in a graph database, namely Neo4j. The MACM is preliminary produced by the user in Neo4j and then requested by the tool (available at link[1]) for the threat modeling phase. The tool communicates with the graph database, obtaining the correctly modeled 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 4.

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

Table 4. 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 [3]
Compromised	Which assets the malicious behaviour compromises

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 threats are then obtained by the tool 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 [13] controls.

4 Tool Comparison

In this chapter we want to compare the different threat modeling 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 modeled the system with the three different tools and documented the threat modelling results each tool offered.

4.1 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 contents, 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 [2]), due to incorrect security planning and management.

4.2 Microsoft Tool Analysis

The Microsoft tool allowed us to describe the Wordpress application in 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. 1).

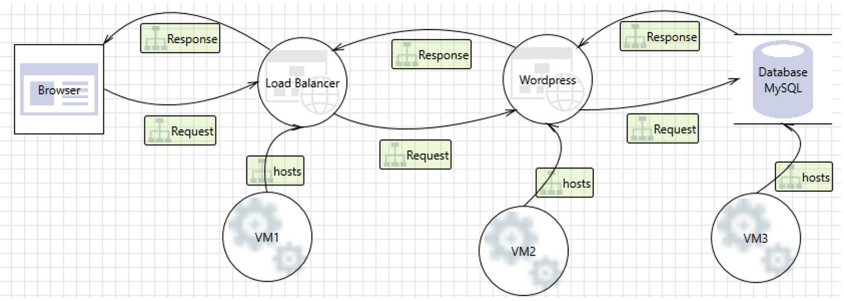

Fig. 1. Microsoft tool model Wordpress

In order to model the application, the client was modeled as a *Browser*, while Wordpress and Load Balancer as a *Web Application*. MySQL Database instead was modelled as a *Database* component value. Each service is running on a *Host* node. Once the used 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 the Table 5.

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

Table 5. 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

data. The sending of the access credentials by the user to the service can also be compromised. In fact, a malicious user can steal these data in different ways. In order to reduce the risk that this threat happens, Microsoft tool suggests some countermeasures. Ad 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.

4.3 Dragon Analysis

We modeled the system using Threat Dragon diagram tool. The number of stencils available is limited, so, as shown in Fig. 2, The Wordpress application was modeled using only the Process, Store and Actor.

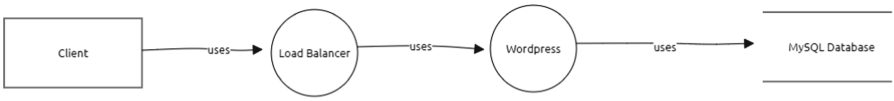

Fig. 2. Threat Dragon model Wordpress

Load Balancer and Wordpress were modeled 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 communicate 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. resource to be protected). Each asset has a set of properties aimed at selecting the related threats, as shown in the Table 6. We modeled 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 the Table 7. 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 it can be subject to Card Cracking threat 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 6. 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 7. 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 Disclosure	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 Disclosure	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 Disclosure	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 a action and enforcement of behavioral workflow

4.4 SLAgenerator Analysis

Figure 3 shows the MACM model of our case study. Each label affect the color 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 labeled 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 modeled the Load Balancer (LB) and WordPress (WP) as *SaaS* nodes and we set their Asset Type as *Web Application*. The MySQL instance, instead, was labeled as a *SaaS*, but with *Database* (DB) value as Asset Type. The LB *uses* the WP that, in turn, uses the DB. The Client (modeled as a *CSC* node) uses the Load Balancer service, that acts as application interface. Each SaaS service is connected to the public Network. Applying our threat modelling technique we produce a list of threats but, for simplicity' sake we report in Table 8 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.

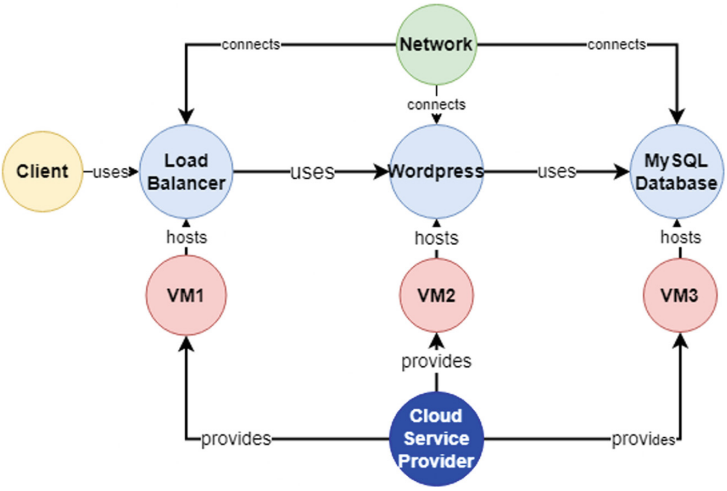

Fig. 3. Wordpress MACM

Table 8. Part of the Threat Model Wordpress using SlaGenerator.

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 unaccessible 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

4.5 Comparison

It is worth noting that, as highlighted above, all the tool rely on a graph-based model to describe the target system, where the node represent the asset and the edge their connections. However, the tools differ on the interpretation and metadata associated to both 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 modeling is associated to the connection among them: in fact the threat 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 type of nodes and edges are very limited and the threats are selected according to few attributes associated to both nodes and relationship. The SLAGenerator, on the other hand, focuses on system assets (the graph nodes) and identifies the possible threats relying on the *asset type* attribute, which offer a large variety of different values, similarly to the Microsoft Threat Modeling tool. Moreover, relationships affect the possible threats to which each node, but the threats are always listed as associated to nodes. 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 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 threat impact on Confidentiality, Integrity and Availability. The three 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 Modeling tool, 84 for SLAgenerator and 31 for the Dragon tool). We, acting as experts, consider that the choice of listing threats only respect to assets or only respect to relationships (the choices done by SLAGenerator and by MS Threat Modeling Tool) helps the expert work in the analysis of the results, but this is

and remain a subjective choice. However, the number of threat outlined by the OWASP tool looks, at state of art, 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.

Table 9. Comparison table.

Asset	SLAGenerator threat	Microsoft threat	OWASP threat
Wordpress	Data Leakage	Read web app's config files	Fingerprinting
		Steal sensitive data like user credentials	Carding
			Card cracking
Wordpress	Injection	SQL injection through Web App	–
Database	Read Injection	SQL injection	–
Database	Insert Injection		Account Creation
VM	Data Breach	Access to sensitive data from log files	–
VM	Denial of Service	–	

Another interesting aspect is that the three techniques present threats at different levels of granularity, as shown in the Table 9. As an example, the SlaGenerator 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. 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 SlaGenerator and 25 according to Microsoft. 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 Database as an asset, a Microsoft SQL injection can be as *SlaGenerator 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). In this case, the threats according to SlaGenerator tool are 15, while OWASP and Microsoft consider only 8. Virtual machines, on the other hand, are not considered in the OWASP model, the table shows the comparison only between SLAGenerator and Microsoft tool. One of the 13 threats described by the SLAGenerator 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 modeled only by the SLAGenerator and threat modeling reported 12 threats[2].

5 Conclusion

In this paper we have analyzed three threat modeling techniques that make use of different models in order to select the threats applicable to the system. The tools analyzed were SlaGenerator, 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 modeling 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 SlaGenerator. Furthermore, the tool also considered the network as an asset, highlighting 12 threats.

References

1. Microsoft threat modeling tool (2018). https://docs.microsoft.com/it-it/azure/security/develop/threat-modeling-tool
2. Abela, R.: Statistics show why WordPress is a popular hacker target (2020)
3. Ansari, M.T., Pandey, D., Alenezi, M.: STORE: security threat oriented requirements engineering methodology (2019)
4. Arsac, W., Bella, G., Chantry, X., Compagna, L.: Multi-attacker protocol validation. J. Autom. Reason. **46**(3–4), 353–388 (2011)
5. Bhattacharya, D.: OWASP threat dragon review (2020)
6. Casola, V., Benedictis, A.D., Rak, M., Villano, U.: Preliminary design of a platform-as-a-service to provide security in cloud. In: Proceedings of the 4th International Conference on Cloud Computing and Services Science - CLOSER, pp. 752–757 (2014)
7. Casola, V., De Benedictis, A., Rak, M., Rios, E.: Security-by-design in clouds: a security-SLA driven methodology to build secure cloud applications. Procedia Comput. Sci. **97**, 53–62 (2016). 2nd International Conference on Cloud Forward: From Distributed to Complete Computing
8. Casola, V., De Benedictis, A., Rak, M., Villano, U.: Toward the automation of threat modeling and risk assessment in IoT systems. Internet Things **7**, 100056 (2019)
9. Drake: Threat Modeling. https://owasp.org/www-community/Threat_Modeling

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

10. Frydman, M., Ruiz, G., Heymann, E., César, E., Miller, B.P.: Automating risk analysis of software design models. Sci. World J. **2014**, 805856 (2014)
11. Goodwin, M.: OWASP Threat Dragon. https://github.com/owasp/threat-dragon/releases
12. Granata, D., Rak, M.: Design and development of a technique for the automation of the risk analysis process in IT Security, p. 14 (2021)
13. Joint Task Force Interagency Working Group: Security and privacy controls for information systems and organizations. NIST (2020)
14. Kornecki, A.J., Janusz, Z.: Threat modeling for aviation computer security. CrossTalk **28**, 21–27 (2015)
15. Musman, S., Turner, A.J.: A game oriented approach to minimizing cybersecurity risk. Saf. Secur. Stud. **8**, 212–222 (2018)
16. OWASP: OWASP Automated Threats to Web Applications (2018)
17. OWASP: Threat Modeling Manifesto. https://www.threatmodelingmanifesto.org/
18. Rak, M., Salzillo, G., Granata, D.: EssecA: an automated expert system for threat modelling and penetration testing for IoT ecosystems. Comput. Electr. Eng. **99**, 107721 (2022)
19. Schaad, A., Borozdin, M.: TAM: automated threat analysis. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC 2012, pp. 1103–1108. Association for Computing Machinery, New York (2012)
20. Singh, S., Tu, H., Allanach, J., Areta, J., Willett, P., Pattipati, K.: Modeling threats. IEEE Potentials **23**(3), 18–21 (2004)
21. Tatam, M., Shanmugam, B., Azam, S., Kannoorpatti, K.: A review of threat modelling approaches for apt-style attacks. Heliyon **7**(1), e05969 (2021)
22. Xiong, W., Lagerström, R.: Threat modeling - a systematic literature review. Comput. Secur. **84**, 53–69 (2019)