Design and Development of a Technique for the Automation of the Risk Analysis Process in IT Security

Daniele Granata and Massimiliano Rak
Department of Engineering, University of Campania Luigi Vanvitelli, Aversa, Italy

Keywords: Threat Modeling, Risk Analysis, Threat Agent, Protocols.

Abstract: Cloud service architectures are very heterogeneous and commonly relies on components managed by third parties. As a consequence, the security verification process of these architectures is a complex and costly process. Moreover, development of application that runs in cloud should take into account the agile software design and development methodologies and a really short time-to market, which are often incompatible with deep security testing. This article aims at addressing such issues proposing a technique, compatible with Security-By-Design methodologies, that automates the threat modeling and risk evaluation of a system, reducing the costs and requiring a limited set of security skills. Through the proposed approach, the software system is analysed identifying the threats that affects the system technical assets, ranking the level of risk associated to each threat and suggesting a set of countermeasures in standard terms; the process requires a minimal user interaction. The proposed technique, was implemented through a dedicated tool and, correctly integrated in development processes, can significantly reduce the need of costly security experts and shorten the time needed to execute a full system security assessment. In order to validate the technique, we compared our results with approaches available in literature and existing tools.

1 INTRODUCTION

The Cloud Computing paradigm, that relies on service-based approach and on delegation of resources and services, affects the way in which applications are being developed: Cloud-native applications often rely on micro-services architectures and/or on integration of Commercial-off-the-shelf (*COTS*) components. Despite the great advantages that such an approach have in terms of costs and time-to-market, security of Cloud applications is an issue, due to the loss of control over resources and on the code, often property of third parties. At same time, the large exposition over internet of Cloud applications and the new regulations (e.g. GDPR, NIS Directive, Cybersecurity act) imposes to take into account strict security and privacy requirements. In order to address security issues the recommended best practice is *Security-by-Design*, which relies on the idea of taking into account security from the very early design phases. However, threat modeling, identification of countermeasures, weakness and vulnerability identification, security and penetration testing are time- and cost- expensive procedures, which hardly match with the market needs of fast adaptation to new requirements and release of new functionalities. In this paper we propose a technique that aims at automating as much as possible the Threat modeling, Risk Analysis and Countermeasure identification process, enabling its integration in agile development methodologies. The approach proposed on this paper relies on the experiences and the proposals made in (Casola et al., 2020b; Casola, 2019) and extends the approach, taking into consideration additional factors, simplifying and empowering the evaluation procedure. In particular this paper offers the following original contribution respect to the state of art: (i) the automated Threat Agent identification, (ii) the automated Threat identification according to assets and protocols involved, (iii) automated Risk ranking and countermeasures suggestion, (iv) a tool that implements the approach. Next Section will summarize the proposed methodology and introduces a simple example that will be used to illustrate the approach. Each of the following sections will describe in detail the proposed steps, demonstrating and validating the approach applying it to the illustrated example. The paper ends with a section on conclusions and future work.

Granata, D. and Rak, M.
Design and Development of a Technique for the Automation of the Risk Analysis Process in IT Security.
DOI: 10.5220/0010455200870098
In *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER 2021)*, pages 87-98
ISBN: 978-989-758-510-4

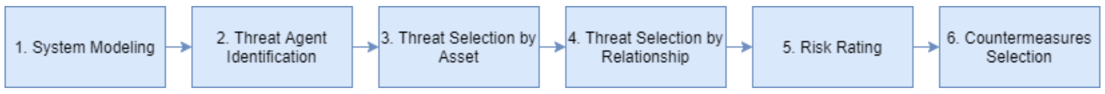

1. System Modeling
2. Threat Agent Identification
3. Threat Selection by Asset
4. Threat Selection by Relationship
5. Risk Rating
6. Countermeasures Selection

Figure 1: The Steps of the proposed methodology.

2 THE TECHNIQUE

According to security best practices, like the ones suggested by (Ross et al., 2016), development of security-critical application, relies on a clear Threat Modeling (Marback et al., 2013), defined as *a systematic process of identifying, analyzing, documenting, and mitigating security threats to a software system* .

The main goal of our approach is to automate as much as possible the Threat modeling and Risk Analysis process, in order to enable its integration in security-by-design methodologies, like the one proposed in (Casola et al., 2020b) and extended in (Casola et al., 2020a). As a matter of fact, we will produce a list of possible threats and suggest a list of countermeasures in terms of security controls that should be verified before deploying the application in production, after a very simplified modeling phase and with minimal user interaction. It is worth noticing that the list of threat is as complete as possible, a discussion about completeness validation will be done in section 5.

A Threat, according to NIST (Ross et al., 2016), is *An event or condition that has the potential for causing asset loss and the undesirable consequences or impact from such loss*. Identify all possible events is, clearly a very complex task. Moreover such threats may be related to design choice, to the involved technologies and/or to custom code development. In order to automate their identification in a generic cloud application, we model a threat as a triple: $ThreatAgent, Asset, Behaviour$.

Considering the concepts in literature ((Fraunholz et al.,) (Casey, 2007), (Dobrovoljc et al., 2017)), we define **Threat Agent** an actor that maliciously act in order to generate the threat event into the system under attack. An **Asset** is, according to NIST (Ross et al., 2016), *An item of value to achievement of organizational mission/business objectives*. In our methodology we focus on technical aspects, so we consider only the technical components of the system under attack as possible assets. We have identified different **Asset Types** that are the classes of asset that can be considered in a system. Our automation process will be able to consider all the threats respect to the asset types considered. In section 3, we will list the supported asset type, adopted in our model.

We define **Behaviour** the natural language description of the malicious event. It is worth noticing that this description is relevant, because gives to the security expert information that helps in clearly identify the issues to be mitigated. Our automation process relies on a catalogue of threats, uniquely identified by the above triple, enriched by a set of additional information: (i) for each threat we outline which security requirement in the CIA triad (Confidentiality, Integrity, Availability) is affected and (ii) we classify the threat according to the STRIDE classification (Kohnfelder and Garg, 1999).

Thanks to the collected information, we evaluate,for each identified Threat, the associated Risk, i.e. the probability that such a threat will take place. This value is calculated as the combination of probability and consequence value, as suggested by (Williams, 2020). Last, but no least, we suggest the list of standard countermeasures, in terms of NIST security controls (Group, 2020).

As illustrated in Figure 1, our approach relies on six steps, even if we ask for user interaction only at start of the procedure (during system modeling) and at end of the procedure (during the risk rating phase), as will be outlined in the following.

The first phase, Modeling, aims at identifying the application architecture and, in practice, the *Assets* involved in the system and their interactions, Section 3 summarizes the main steps involved in this phase, which is the one that needs major user interaction. The second phase, described in Section 4 aims at identifying the possible Threat Agents that will characterize the threats and will be needed to rank the risks associated to the threats.

Third and fourth steps, described together in section 5, identify the possible threats that affect the system taking into account the asset types and the relationship among them. The Risk Rating step evaluates the probability associated to each threat according to the OWASP methodology, estimating the needed parameters for the evaluation thanks to the information generated by the model and the collected catalogues.

Last, but no least, we generate a list of Security Controls, according to NIST control framework in SP 800-53 (Group, 2020), that should be verified on each asset; such list takes into account both the threat list and their risk evaluation.

It is worth noticing that the methodology does not assume any interaction with the end user, excluded at start of the process (modeling phase and a set of questions related to threat agent selection) and at end of the process (in order to evaluate the impact of threats on the business, that can be evaluated only by the system owner.

2.1 A Simple Case Study

In order to validate the proposed technique, we will use a very common system, typically executed on a cloud infrastructure: an e-commerce site developed on top of WordPress. WordPress (WP) 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 time, reusing always the same Database (that can scale only vertically, i.e. adding memory and/or CPU to the hosting VM). A Load Balancer (LB) distributes the Client requests to the connected WP instances. The developer simply customize the WP instances , through custom plugins and customizing the application behaviour.

Even if development of such systems is simple and commonly relies on very limited skills from the developer/system administrators, the application manage moneys 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.

3 MODELING

The first step of technique is the system Modeling that relies on the MACM (Multi-Application Cloud Composition Model) formalism (Rak, 2017), a graph-based model in which each node of the graph represents the component of the system, and each edge is a connection between the components.

Each Node has a different label that classify it respect to the system deployment. Model labels can be related to typical cloud roles (*CSP*, Cloud Service Provider, or *CSC*, Cloud Service Customer) or to service models, e.g. *IaaS* (Infrastructure-as-a-Service), *PaaS* (Platform-as-a-Service), and *SaaS* (Software-as-a-Service),. Labels affect the relationships (graph edges) in which the nodes can be involved.

Moreover, each node has a set of properties that

Table 1: Assets and their types in the case Study.

Node	Label	Asset Type
Client	CSC	CSC
CSP	CSP	CSP
LB (Load Balancer)	SaaS	WebApplication
WP (WorkPress)	SaaS	WebApplication
DB (Database)	SaaS	Database
VM (Virtual Machine)	IaaS	VirtualMachine

Table 2: Relation between components in case study.

Start Node	Relation	End Node	Protocol
Client	uses	LB	https
LB	uses	Wordpress	https
Wordpress	uses	DB	mysql
VM	hosts	Wordpress	-
VM	hosts	LB	-
VM	hosts	DB	-
CSP	provides	VMs	-

specifies the characteristic of the node. A mandatory property, for the services, is the *type*, that specifies the *Asset Type*, according to the concept expressed in the previous section. As an example *IaaS* nodes can be of *VM* or *container* asset type, while *SaaS* can be of *Web Application*, *Database* or *IDM* (Identity Management System), asset types. New asset types can be easily added, as an example the MACM model was extended to support IoT systems in (Casola, 2019).

The (directed) edges of the graph represent the relationship among the nodes. The model adopts few different kind of relationship, namely: *provides*, *hosts* and *uses*. The relationship outlines the way in which the different type of components may interact, as an example the *uses* relationship among two services, outlines that a service uses the capabilities offered by the other. The model allows to associate properties to relationships, e.g. it is possible to specify a *protocol* attribute to a *uses* relationship, whose value indicates the protocols involved in the interaction.

Figure 2 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 by a Cloud Service Provider (e.g. Amazon 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 *hosts* 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 Customer(s) (modeled as a *CSC* node) uses the LB node, that acts as application interface. Tables 1 and 2 summarize the model.

In order to manage the MACM model our tools represent them in a graph database, namely Neo4j [1]. Our Tools, available at link[2], are able to automatically manage and generate MACM models in many different ways, however, in this paper, we assume that the MACM model is simply available to the tool and stored in the graph database.

4 THREAT AGENT SELECTION

As already outlined, Threat Agents are *any person or thing that has the power to act to cause, carry, transmit, or support a threat*. In a security assessment process the identification of Threat Agents is a fundamental part and its results influence the Risk Rating activities. In order to identify possible threat agents, we adopted the taxonomy proposed in (Casey, 2007), that suggests the categories described in table 3.

The classification proposed in (Casey, 2007) associate to each class of threat agents a set of attributes and proposes a mapping that outlines which are the attribute values associated to each Threat Agent. Table 4 summarizes the attributes and the possible values of the attribute.

In order to select the Threat Agents (TAs) and evaluate the risk associate to each of them, we classify the proposed attributes in two classes: (i) attributes useful to identify the TAs that are meaningful for the system under analysis and (ii) attributes useful to risk evaluation.

In the selection process we concentrate on the first class, which, according to our consideration are *Intent*, *Access*, *Outcome*, *Objective*. Accordingly we identified four questions to which the system owner should answer, that enable us to select the TAs.

- Q1: Are there someone who can gain an advantage implementing a cyber threat against your system?
- Q2: Do you trust all employees and do you assume that they are not a possible Threat agent?
- Q3: What are the goals of the attackers that represent the most threat to you?
- Q4: What could be the expected results of a possible attacker in the phases of a possible attack on the software system?

Q1 identifies the threat agent's hostility. Q2 is used to consider the threat agents that have internal access to the application. For example, if the user marks the employees (or partners who have access to confidential data) as “trusted”, these categories are removed from the final threat agent list. Q3 and Q4, instead, allow multiple answers and apply the filter to the threat agents in relation to the steps they take to attack the system and the desired result of the attacker. The answers to these questions are used to select the threat agent categories. For example, Terrorist or Data miner can not be selected if the user consider ”Embarrassment” the only outcome. The result of the first step is a threat agent list containing a description of the threat agent and common actions he takes to attack the target.

4.1 Application to Our Case Study

In our case study we assume that the system's cyber-security officer considers only the hostile threat agents. This may due to the exposure of threats that can bring financial advantage to the agent, especially on Wordpress site such as e-commerce. We also assumed that the director has complete trust of the employees, which excludes them from the results. Having made these answers to the questionnaire, the categories interested in attacking the system are described by the table 5.

It should be noted that some categories are always considered as outputs of the algorithm as they represent a set of threat agents that do not have a specific reason for attacking the system, so they should be considered to every kind of systems. In our case the user provides a multiple answer to questions Q3 and Q4. He answers ”Copy” to Q3 question and ”Tech Advantage” and ”Business Advantage to Q4 question. When an user gives multiple answer to Q3 or Q4 question, a set of threat agent categories is calculated for each answer and the final result is the union of each set. For example, in our case study the answers considers both a Technological Advantage and a Business Advantage. The result of the algorithm is the union of the sets obtained from the categories associated with the technological advantage and the business advantage. In our case study, therefore, five categories are detected to be consider as possible threat agents and these categories will be used for risk rating phase, as shown in table 5.

[1]https://neo4j.com
[2]https://bitbucket.org/daniele94/sla-generator

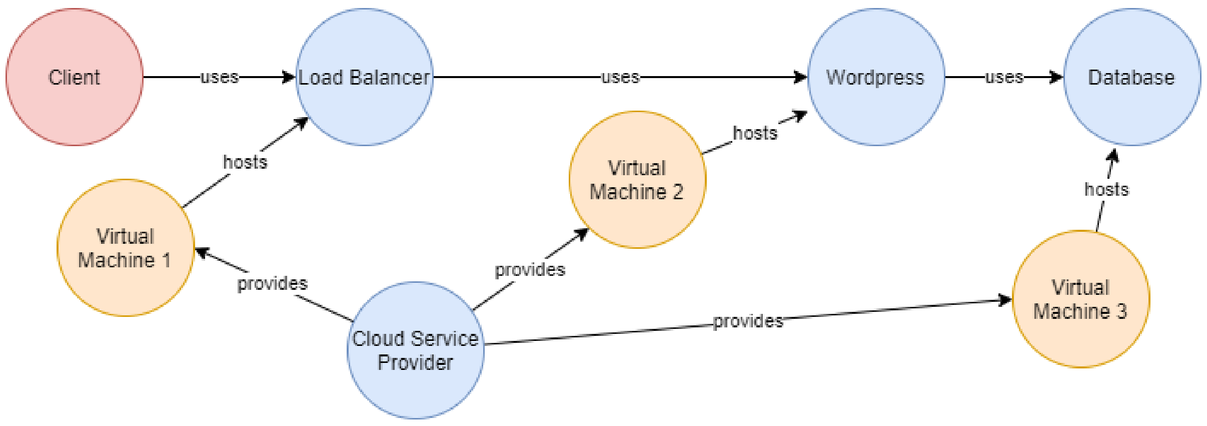


Figure 2: MACM Wordpress Case study.

Table 3: Threat Agent Taxonomy.

Category	**Description**
Employee Reckless	Employee who circumvents safeguards for expediency,but intends no harm
Employee Untrained	Employee with harmless intent but unknowingly misuses system or safeguards
Info Partner	Someone with whom the company has voluntarily shared sensitive data
Anarchist	Someone who rejects all forms of structure and acts with few constraints
Civil Activist	Highly motivated but non-violent supporter of cause
Competitor	Business adversary who competes for revenues or resources (acquisitions,etc.)
Corrupt Government	Person who uses his position within the government to acquire resources
Data Miner	Professional data gatherer external to the company (includes cyber methods)
Employee Disgruntled	Current or former employee with intent to harm the company
Government Cyberwarrior	State-sponsored attacker with significant resources to affect major disruption
Government Spy	State-sponsored spy as a trusted insider,supporting idealistic goal
Internal Spy	Professional data gatherer as a trusted insider
Irrational Individual	Someone with illogical purpose and irrational behavior
Legal Adversary	Adversary in legal proceedings against the company, warranted or not
Mobster	Manager of organized crime organization with significant resources
Radical Activist	Highly motivated,potentially destructive supporter of cause
Sensationalist	Attention-grabber who may employ any method for notoriety
Terrorist	Person who relies on the use of violence for socio-political purposes
Thief	Opportunistic individual with simple profit motive
Cyber Vandal	Derives thrills from intrusion or destruction of property, without strong agenda
Vendor	Business partner who seeks inside information for financial advantage

4.2 Implementation

We implemented the Selection Process as a wizard in which all the information necessary to obtain the list of threat agents is obtained from the user, submitting the answers to the above proposed questions.

The Threat Agent selection process involves the use of a specific data model, which allows us to perform the described operations, which is logically organized in two parts:

- A static part that contains a set of data already defined upstream of the process.
- A dynamic part that contains the results of the wizard related to the selected application.

Figure 3, illustrates the E-R model of data model. Blue tables are the static part of data model, made of 6 tables, devoted to collect on one side the *Threat Agent Categories*, the *Attributes* and their mapping, according to the referenced paper (Casey, 2007) and , on the other side, *Questions*, *Replies* of questionnaire and a table, which links each answer to a category that satisfies it.
It is worth noticing that it will be easy, in future works and after additional validation processes to enhance

Table 4: Threat Agent Attributes.

Attribute	Description	Attribute Values
Access	Privileged position of the target infrastructure	Internal,External
Intent	Whether the agent intends harm	Hostile, Not Hostile
Limits	Legal and ethical limits of threat agent	Code of Conduct, Legal, Extra-legal Minor, Extra-legal Major
Outcome	Primary goal of threat agent	Acquisition, Business Advantage, Damage, Embarrassment, Technical Advantage
Objective	Method agent uses for achieving goals	Copy, Deny, Destroy, Damage, Take, All
Resources	Available time, money and technological means	Individual, Club, Contest, Team, Organization, Government
Skills	Special training and expertise	None, Minimal, Operational, Adept
Visibility	How hidden are identity and actions	Overt, Covert, Clandestine, Don't Care

Table 5: Results of Threat Agent Selection phase in the example Scenario.

Category	Description	Common Actions
Competitor	Business adversary who competes for resources	Theft of IP or business data
Cyber Vandal	Derives thrills from intrusion or destruction of property, without strong agenda	Network/Computing disruption, web hijacking, malware
Irrational Individual	Someone with illogical purpose and irrational behavior	Personal violence resulting in physical business disruption
Data Miner	Professional data gatherer external to the company	Theft of Personally Identifiable Information, IP or business data
Sensationalist	Attention-grabber who may employ any method for notoriety of fame	Public announcements for PR crises, theft of business data

the approach, adding new categories, attributes and, eventually, additional questions.

The dynamic part,shown in figure 3 in red, is created at run-time, and refers to each single user. The *Applications* table contains all information about the application to be subjected to security assessment. At the end of the questionnaire, a specific application is associated with a table, called *questionThreatAgentOnApplications*, which contains all the answers of the questionnaire. *ThreatAgentCategoryAppliesOnApplication* table contains all OWASP parameters calculated by the algorithm. Finally,the algorithm calculates some Threat Agent scores associated to the application and save them in *ThreatAgentScores* table, according to the algorithm shown in section 6.

5 THREATS SELECTION PHASE

As anticipated, the core of the selection phase, already presented in (Casola et al., 2020b) (Casola et al., 2020a) (Casola et al., 2016), relies on a Threat Catalogue, built in the context of the MUSA H2020 project, that organize the threats according to their asset type. The threats of the catalogue were collected using a set of well known sources, like OWASP top threats or referenced scientific paper, maintaining the link to the adopted source in the catalogue. The threat catalogue is available as open data.[3]

However, in this paper, we enhanced our approach, considering not only the asset types, but even the relationships among the assets. As outlined in section 3, the attribute *protocol* associated to the MACM relationships outlines the protocol(s) adopted in communication among different nodes. Moreover, we use the direction of the edge among the node (MACM relies a directed graph) in order to assign a *role* to each asset involved in the communication. As an example, if a Customer (CSC) and a web application through the HTTP protocol, the MACM model will add the HTTP attributes to the *uses* relationship among them, assigning the the CSC the role of HTTP client, while the application assumes the role of server. At state of art, the proposed approach supports only client-server relationships among assets, in future works we aims at extending the approach to different paradigms.

Accordingly we enriched our threat catalogue with protocol-related threats. At state of art we sup-

[3]In case you are interested send an email to massimiliano.rak@unicampania.it to request the latest version.

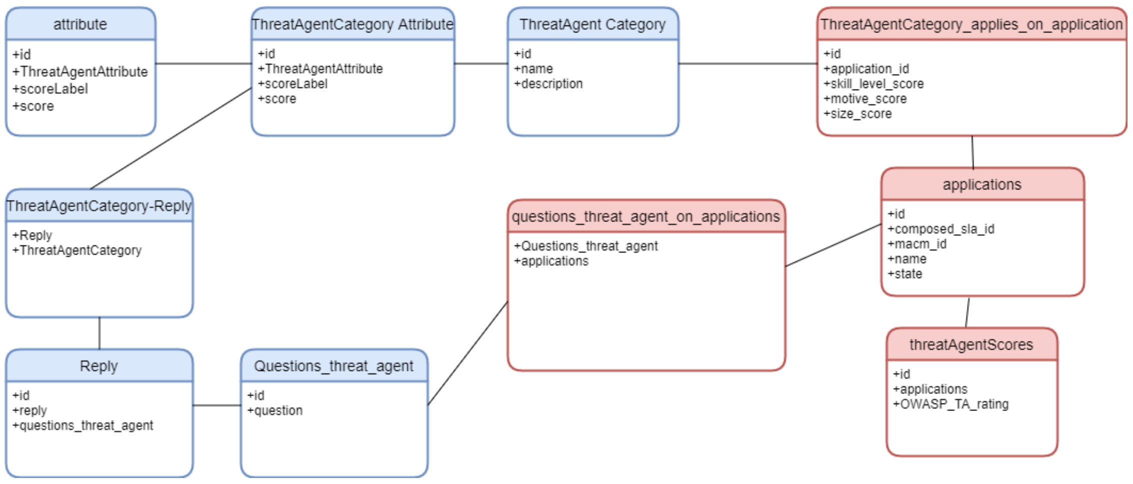


Figure 3: Data model.

ported the threats related to TCP, HTTP, SSL/TLS, HTTPS and OAUTH using standard and well-known threat models for protocols like (Qualys, 2013) and (T. Lodderstedt, 2012).

The proposed approach produce a pretty large list of threats: one for each triple $< ThreatAgent, Asset, Behaviour >$. Accordingly some of the threat behaviours collected in the catalogue may appear more than one time, being applied to different assets. Moreover, it should be noted that, even if the threats are selected according to the protocols, we associate them to each single asset, according to the role they have in the protocol. This enable us to focus on different countermeasures for each asset , taking all the threats in consideration.

5.1 Implementation and Data Model

The enhancement of the proposed approach has implied an improvement of the Data Model needed to maintain the Threat catalogue, which is summarized in Figure 4.

The proposed data model provides a dynamic table named ”components” linked through an N:M relationship to a table containing the catalog of threats. Threats due to protocols are mapped to *Protocols* table through a table named *threatprotocol*. All protocol information is extracted from MACM and relation are stored in database through a *Relation* table witch maps component to the used protocol and also defines the role that component plays in that communication. In this step *role* attribute is stored in *Relation* table based on the orientation of the arch. All Threats are extracted from database using a query that select all threats linked to a specific protocol filtering by *role* attribute. In this way we distinguish threats related to two components of the same type and which communicate with a specific protocol, but which have different roles in the communication.

5.2 Threat Selection Example

According to our case study, the assets are the ones already anticipated in the previous sections and summarized in tables 1 and 2 of the Modeling section. Applying the proposed approach we produce a list of threats that we have summarized, for simplicity’ sake we summarized some of the threats due to HTTPS in Table 7 and some of the Threats obtain per asset in Table 6. A list of Threats is not compatible with the length of the paper. Note that we manage it directly through our tools, that enable easy exploration of threats through a web interface.

Analyzing the tables it is possible to outline, as an example, that Wordpress can be exposed to threats such as Injection. Other threats that affects the VMs are linked to data and policy violation such as Data Breaches and Unauthorized access to admin interface. Moreover, data can be sniffed by threat agents in communication channel between SQL-database and Wordpress or the data could even be physically destroyed.

5.3 Validation through Comparison with Microsoft Tool

Validation of threat modeling processes is always an issue, because there are no common ways to demonstrate completeness (i.e. that all threats are considered) and consistency (i.e. that the threats considered are concretely applicable to the system under analysis). We made a comparison with the results of

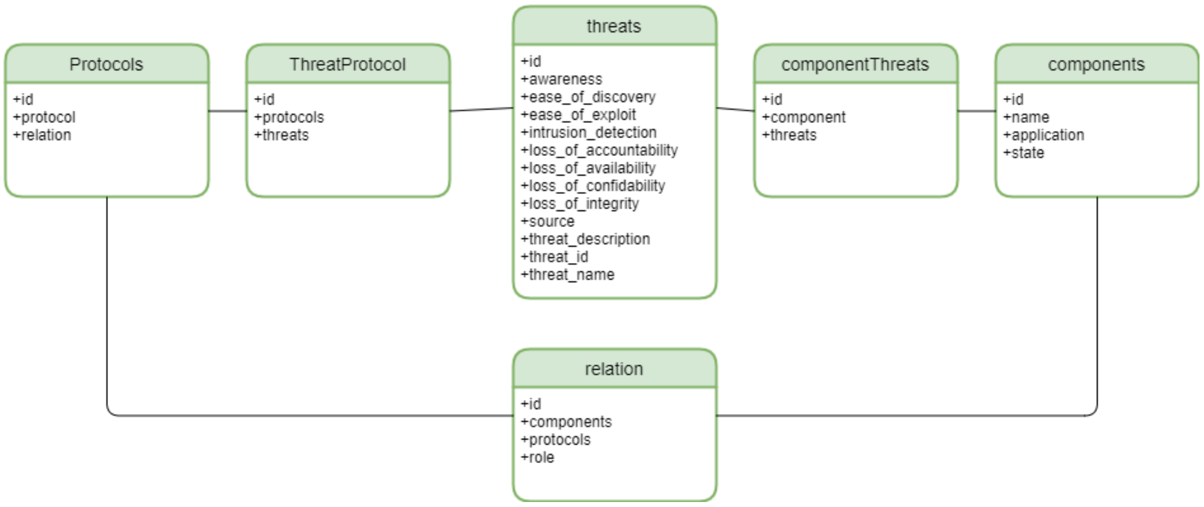


Figure 4: Table Protocols, Threats and Components in data model.

Table 6: Threat Selection results per Asset (snippet).

Threat ID	Threat Name	Asset
T94	Injection	Wordpress
T98	Security Misconfiguration	Wordpress
T99	Cross Site Scripting	Wordpress
T70	Sniffing Storage Traffic	DB
T87	Physical Destruction of Storage Media	DB
T77	Exhausting Log Data and Metadata Space	DB
T59	Data Breaches	VM
T59	Unauthorized access to admin interface	VM
T59	Weak Identity, Credential and Access Management	VM
...	...	...

Table 7: Threats due to HTTPS in LB-Wordpress relation.

Threat ID	Behavior	Asset
T117	Eavesdropping	WP
T118	Message Reply	LB
T119	Message Injection	WP
T120	Message Deleting	LB
T121	Spoofing users	WP
T122	Gaining unauthorized entry to a server or account	WP
T124	Denying service to other entities	WP
T126	Denial Of Services	WP
T127	Truncation	LB
T128	Message Modification	WP
T128	Message Modification	LB
..	..	..

a similar tool, The Microsoft Threat modeling Tool, that relies on similar approaches, even if it does not support threat agent selection and risk ranking evaluation. The Microsoft application model relies on a graph-based very similar to MACM. The main difference among our approach the MS one, is that the Microsoft tool associates the threats to the arcs of the graph, while the proposed technique provides a list of threats for each component. Microsoft threat model per-Component is extrapolated considering threats related to arcs in-going or out-going the component.

As an example, considering the WordPress component, are listed in 8 (for space motivation we report only the table related to this asset). Our approach proposes 18 threats, against 27 using Microsoft tool. The main difference among the approaches is that the MS tool suggestes more *low level* threats, which supports a technician in a technology oriented problem solution, while our tool suggests *high level* threats, more prone to policy definition and audit verification procedures. Moreover, table 8 outlines how a set of threats proposed by MS tool are specific cases of the threats suggested by our approach, as an example we outlines the threat of security misconfigurations (whose countermeasure could be the adoption of an audit procedure and/or of automatic tools based on SCAP (D. Waltermire, 2018)), while the MS tools suggests a specific list of security misconfiguration cases. For what regard the detected threats affecting the MYSQL-Database, we suggested 22 threats, against the 7 suggested by the Microsoft tool. For VM asset, instead, we considered 17 different threats while Microsoft shows only 2 threats. In both cases the threats suggested by MS are subcases of the ones identified by us.

The difference in the last two asset types is probably due to the asset-oriented approach we adopt, against the communication oriented suggested by Microsoft. These results demonstrated the reliability of the proposed threat modeling approach, considering

Table 8: Comparison Table.

Asset	**Our Threat**	**Microsoft Threat**
Wordpress	Injection	SQL injection through Web App
Wordpress	Security Misconfiguration	Gaining access to sensitive data from log files
		Gaining access to sensitive information through error
		Spoofing due to insecure TLS certificate configuration
		Improper logout or timeout configuration
		Insecure coding practices
Wordpress	Cross Site Scripting	XSS
DB	Sniffing Storage Traffic	Sniffing traffic to database
DB	Exhausting Log Data and Metadata Space	-
VM	Data breaches	-
VM	Unauthorized access to admin interface	Unauthorized admin privileges
	Weak Identity, Credential and Access Management	

that the commercial tool was not able to identify a threat that we are not able to consider (at least at an higher level of generality). As a matter of fact, the MS tool is a powerful tool that directly suggest technician on possible security issues and helps them in directly apply countermeasures, while our technique is more prone to identify an higher level security policy and identify development procedures that enables to solve the security problems.

6 RISK EVALUATION

Threat Modeling offers a clear idea of the possible menaces that threaten the system under analysis, however addressing them all could be a (costly) overkill, not compatible with time-to-market and application costs. Risk Analysis aims at offering a (rough) evaluation of the risk (i.e. the probability that a threat happens) in order to prioritize the implementation of the countermeasures. We adopted the Risk Rating Methodology proposed by OWASP (Williams, 2020) that evaluate (as commonly happen) the risk through the composition of two indicators: *Likelihood* and *Impact*. *Likelihood* is an indicator that expresses how likely a threat agent is to implement a threat. OWASP quantitatively models it considering two set of factors: Threat Agent Factors and Vulnerability Factors. The first ones are related to the group of threat agents, considering : *Skill Level* , *Motive*, *Opportunity* and *Size* While *Vulnerability Factors* are related to the vulnerabilities needed to exploit a specific threat agent and take into account *Ease of Discovery*, *Ease of Exploit* , *Awareness* and *Intrusion Detection*.

Even *Impact* relies on two set of factors, namely *technical* and *business* factors. The technical impact is estimated taking into account how threat affect the security requirement of the asset in terms of the *Loss of Confidentiality* , *Loss of Integrity* , *Loss of Availability* and *Loss of Accountability* .

Last but no least *Business factors* take into account what is important to the company running the application, evaluating *Financial damage* *Reputation damage* , *Non-compliance* and *Privacy violation*

The OWASP methodology offers a descriptive criteria in order to assign to each of the above factors a number between 1 and 10. Likelihood and Impact will assume a level of risk (*Low*, *Medium* or *High*) if the average of the values of their factors is respectively in the range $1-3$, $4-6$ or $7-10$. The risk value of a threat is assigned through a table that assign the final risk level according to the Likelihood and Impact levels.

Thanks to our approach, Threat Agents and the behaviour of the threats are well known in advance, so it is possible to evaluate 12 on 16 of the factors without any user involvement. The only factors that we will ask as an input are related to business impact, which are strictly related to the context of execution of the application and its market considerations.

Table 9: Mapping OWASP Threat Agent parameters with TAL parameters.

OWASP parameter	**TAL Parameter**
Skill Level	Skill
Motive	Limits, Outcome, Intent
Opportunity	Access, Resources, Visibility
Size	Resources

6.1 Threat Agent Rating

As a consequence, Threat Agent Factors are calculated on the basis of the considerations made in section 4. We considered these values to be associated with the system and therefore they are the same for every threat in threat model.

The evaluation of the OWASP parameters relating to threat agents is based on the categories resulting from the questionnaire described in the chapter 5 Each category has a TAL attribute set that describe it. Each attribute has a score which we use to calculate the OWASP values. At this step, both attribute classes, described in chapter 5 are considered. For example, we assume that OWASP Motive depends only on *Intent*, *Outcome* and *Limits* and it is calculated as follows:

$$OWASPMotive = \frac{\frac{Intent}{2} + \frac{Outcome}{5} + \frac{Limits}{4}}{3} x10 \quad (1)$$

For multiple responses related to attributes (such as Outcome values) , the considered value is the average of the single associated ones.

The process in this way calculates four OWASP parameters for each category, using the mapping described in the table 9. In order to produce the parameters relating to the system, it is necessary to evaluate the combination of all the OWASP parameters for each category of the Wizard result. For example, the total OWASP Motive value is given by the formula:

$$OWASPMotiveTOT = \frac{\sum_{i=1}^{N} A_i M_i}{A_{tot}} \quad (2)$$

N represents the number of categories resulting from the Wizard and M_i the OWASP score of the reason relating to the i-th category and d A_i instead represents the i-th weight assigned by the user to the single output category through a Low-Medium-High qualitative approach. In this way, the user decides which output category is more dangerous for the application.

6.2 Vulnerability and Technical Impact Factors

In order to evaluate the risk of a threat, we still miss the factors related to Vulnerability and Technical Impact. According to OWASP, a security expert, should make such evaluations, taking into account (i) how to implement an attack, exploiting the vulnerabilities to implement the threat, and (ii) which are the effect of the threat over the asset. However, as outlined in section 5, we collected in the catalogue the full set of possible threats in terms of the couple $< Asset, Behaviour >$ together with additional information, namely the security requirements affected and the STRIDE classification of the threat. This enabled us to made an evaluation of each pair $< Asset, Behaviour >$, and identify the default value for the eight factors, associated to such a pair. In practice, we evaluated in advance all the possible threats assigning the values in order to avoid any additional request. In future works we aims at improving such an evaluation, through additional attributes to the MACM model: knowing the technologies that implement a node, as an example, it is possible to make a search over threat intelligence knowledge bases (like NVD [4] or the MITRE CVE, CWE, ATT&CK or CAPEC, in order to make an additional automated evaluation.

6.3 Overall Evaluation and Input Collection

As a summary, for each threat, given by the triple $< ThreatAgent, Asset, Behaviour >$, we are automatically able to estimate 12 over 16 of the factors requested by the OWASP methodology. However, the number of possible threats remains pretty long and involving the user in order to collect the last four factor for each of the (tens, if not hundreds) identified possible threats is unpractical. Accordingly we followed a simplified approach,: each threat was associated to one or more STRIDE (Kohnfelder and Garg, 1999) categories, Spoofing, Tampering, Reputation, Information Disclosure, Denial of Services and Elevation of Privileges. Then we ask the user to express the impact factor of each STRIDE category over the overall system (it is possible to make a per-asset request if the user is interested to a finer grain evaluation). Finally, we evaluate the impact of a threat on an asset considering the higher impact among the ones that the user declared for each STRIDE category to which the

[4]National Vulnerabiity Database NVD, https://nvd.nist.gov

threat belong to.

As a final result we are able to make a Detailed risk analysis, outlining, for each threat, the associated Risk Level (as a value Low, Medium or High).

Validation of the proposed evaluation is, however, almost impossible, as often happens in Risk analysis approaches. We relies on a well-established methodology (the OWASP one) that assign a relevant role to security-experts. Our technique anticipated the work of the security experts, so that it is possible to automate the process, delegating it to developers with limited security skills. As a consequence, it is possible to involve the (costly) experts only for validating the results on the specific case studies or in given phases of the project development.

6.4 Evaluation of the Case Study

In our case study, we selected five different Threat Agents, listed in table 5. Let's consider, for example, the OWASP Factor Motive for the *Data Miner* agent:

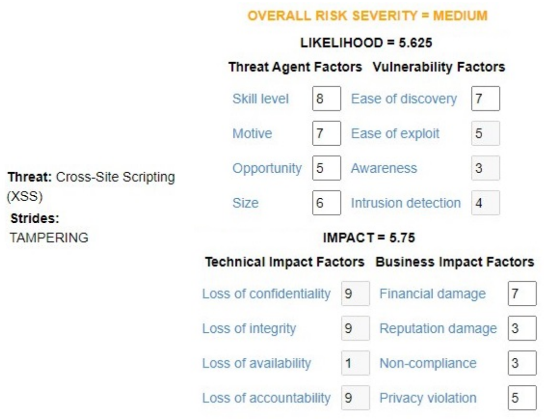
$$DataMinerMotive = \frac{\frac{2}{2}+\frac{3}{5}+\frac{3}{4}}{3}x10 = 7 \qquad (3)$$

Accordingly, the OWASP Motive values calculated by 3 give the value 7, it means that threat agents might be strongly motivated to attack the system. A similar approach is used for the OWASP Size, Motive and Opportunity parameters.

The calculation of threat agent parameters is unique for each resulting threat described by tables 7 and 6. Other OWASP parameters depends on threat, so for greater readability we take into account some example of threats in table 6. Evaluating the Cross-site Scripting threat, all OWASP related values are shown in figure 5. Considering the resulting threat agents and Cross-site scripting threat, the related overall risk is considered MEDIUM (that typically implies that countermeasures must be applied in a short amount of time)

7 SECURITY CONTROL SELECTION

In order to close the process we propose a set of (standard) countermeasures to mitigate the threats and enhance the overall level of security associated to the system. This step of the technique were already available in previous work, (Casola et al., 2020b; Casola et al., 2020a) and simply relies on the additional information collected in the threat catalogue. Countermeasures are modeled according to the NIST security control framework ((Group, 2020)) which provides a



OVERALL RISK SEVERITY = MEDIUM

LIKELIHOOD = 5.625

Threat: Cross-Site Scripting (XSS)
Strides: TAMPERING

Threat Agent Factors		Vulnerability Factors	
Skill level	8	Ease of discovery	7
Motive	7	Ease of exploit	5
Opportunity	5	Awareness	3
Size	6	Intrusion detection	4

IMPACT = 5.75

Technical Impact Factors		Business Impact Factors	
Loss of confidentiality	9	Financial damage	7
Loss of integrity	9	Reputation damage	3
Loss of availability	1	Non-compliance	3
Loss of accountability	9	Privacy violation	5

Figure 5: Risk values of Cross-site Scripting Threat.

Table 10: Controls suggested for some threats in Threat Model.

Control Name	**NIST Control ID**	**Threat Name**
Information input validation	SI-10	Injection
Information input validation	SI-10	Cross Site Scripting
Transmission confidentiality and integrity	SC-8	Sniffing Storage Traffic
Protection of information at rest	SC-18	Sniffing Storage Traffic

large catalogue of security controls (e.g. countermeasures) organized in families, controls and control enhancements. For each threat provided by the threat model, we query the threat catalogue that contains, as additional information, a list of nist security controls able to mitigate each couple *Asset*, *behaviour*, available in the catalogue. It is worth noticing that the NIST framework suggests a reference baseline, associating to each control the security level that require the control as a mandatory countermeasures, accordingly we select all the security controls, associated to the identified threats, that are needed when the risk is higher than the one we have evaluated according to the process illustrated in previous sections. Table 10, illustrated some of the NIST controls suggested for our case study. For example to avoid Injection threats for Web Application, this phase of methodology suggest to use a validation system for input and output values. Being a previous work, we does not spend additional space in the details related to such control selection.

8 CONCLUSIONS

In this paper we presented a technique that aims at automating thee process of threat modeling, risk analysis and security policy definition. The proposed approach, thought to be easily adopted in security-by-design development methodologies, enables non-security experts to automate many of the typical steps performed by security experts. It is worth noticing that our goal is not to substitute the experts, but enabling their (costly) involvement only when strictly necessary, in order to validate and eventually enrich the policies, and during the development in order to verify the correctness of the automated decisions.

The proposed approach require a very simplified model of the application, similar to state of art tools, like the Microsoft Threat modeling tool, and the reply to few very simple questions, that are needed to make the risk evaluation. The proposed technique was validated comparing its result with existing tools (like the Microsoft one) and relying on standard and existing methodologies for the more subjective phases of the procedure. The comparison with the MS threat modeling tool, as already outlined in section 5, outlined that we identified all the threats suggested by the competitor, sometime suggesting a more general threat respect to the one proposed by the tool. At best of author's knowledge, no other tool and technique is able to support in a coherent and homogeneous way threat agents identification, threat modeling, risk analysis and countermeasures identification without any user interaction of not few initial questions and the starting model we require. In future works we aims at studying a technique to validate risk analysis techniques like the one proposed, adopting threat intelligence data-sets, in order to both, offer additional grants on our results and improve the quality of our risk level evaluations.

Moreover we aims at automating, as much as possible, the process of enrichment of the threat catalogue, collecting data from open data sets and enriching the risk factor evaluation through dedicated analysis and testing procedures.

REFERENCES

Abela, R. (2020). Statistics show why wordpress is a popular hacker target.

Casey, T. (2007). Threat agent library helps identify information security risks. page 12.

Casola, V. (2019). Toward the automation of threat modeling and risk assessment in IoT systems. *Internet of Things*, page 13.

Casola, V., De Benedictis, A., Rak, M., and Rios, E. (2016). Security-by-design in clouds: A security-sla driven methodology to build secure cloud applications. *Procedia Computer Science*, 97:53 – 62. 2nd International Conference on Cloud Forward: From Distributed to Complete Computing.

Casola, V., De Benedictis, A., Rak, M., and Salzillo, G. (2020a). A cloud secdevops methodology: From design to testing. In *International Conference on the Quality of Information and Communications Technology*, pages 317–331. Springer, Cham.

Casola, V., De Benedictis, A., Rak, M., and Villano, U. (2020b). A novel Security-by-Design methodology: Modeling and assessing security by SLAs with a quantitative approach. *Journal of Systems and Software*, 163:110537.

D. Waltermire, J.-M. (2018). Transitioning to the Security Content Automation Protocol (SCAP) Version 2. White Paper, NIST.

Dobrovoljc, A., Trček, D., and Likar, B. (2017). Predicting Exploitations of Information Systems Vulnerabilities Through Attackers' Characteristics. 5:13.

Fraunholz, D., Anton, S. D., and Schotten, H. D. Introducing GAMfIS: A Generic Attacker Model for Information Security. page 6.

Group, J. T. F. I. W. (2020). *Security and Privacy Controls for Information Systems and Organizations*. NIST.

Kohnfelder, L. and Garg, P. (1999). The threats to our products. *Microsoft Interface, Microsoft Corporation*, 33.

Marback, A., Do, H., He, K., Kondamarri, S., and Xu, D. (2013). A threat model-based approach to security testing. *Software: Practice and Experience*, 43.

Qualys (2013). SSL Threat Model. https://www.ssllabs.com/projects/ssl-threat-mode.

Rak, M. (2017). Security assurance of (multi-)cloud application with security sla composition. In Au, M. H. A., Castiglione, A., Choo, K.-K. R., Palmieri, F., and Li, K.-C., editors, *Green, Pervasive, and Cloud Computing*, pages 786–799, Cham. Springer International Publishing.

Ross, R., McEvilley, M., and Oren, J. C. (2016). Systems security engineering considerations for a multidisciplinary approach in the engineering of trustworthy secure systems.

T. Lodderstedt, M. McGloin, P. H. (2012). OAuth 2.0 Threat Model and Security Considerations draft-ietf-oauth-v2-threatmodel-08. RFC 6819, RFC Editor.

Williams, J. (2020). OWASP Risk Rating Methodology. https://owasp.org/www-community/OWASP_Risk_Rating_Methodology.