Asset-Oriented Threat Modeling

Nan Messe, Vanea Chiprianov, Nicolas Belloir, Jamal El-Hachem, Régis Fleurquin, Salah Sadou

Asset-Oriented Threat Modeling

Nan Messe*, Vanea Chiprianov*, Nicolas Belloir*, Jamal El-Hachem*, Régis Fleurquin*, Salah Sadou*
* *Université Bretagne Sud - IRISA, France*
Email: firstname.lastname@irisa.fr

***Abstract*—Threat modeling is recognized as one of the most important activities in software security. It helps to address security issues in software development. Several threat modeling processes are widely used in the industry such as the one of Microsoft SDL. In threat modeling, it is essential to first identify assets before enumerating threats, in order to diagnose the threat targets and spot the protection mechanisms. Asset identification and threat enumeration are collaborative activities involving many actors such as security experts and software architects. These activities are traditionally carried out in brainstorming sessions. Due to the lack of guidance, the lack of a sufficiently formalized process, the high dependence on actors' knowledge, and the variety of actors' background, these actors often have difficulties collaborating with each other. Brainstorming sessions are thus often conducted sub-optimally and require significant effort. To address this problem, we aim at structuring the asset identification phase by proposing a systematic asset identification process, which is based on a reference model. This process structures and identifies relevant assets, facilitating the threat enumeration during brainstorming. We illustrate the proposed process with a case study and show the usefulness of our process in supporting threat enumeration and improving existing threat modeling processes such as the Microsoft SDL one.**

***Index Terms*—threat modeling (process), asset-based reference model, asset identification, attack pattern**

I. Introduction

Threat modeling is recognized as one of the most important activities in software security [18]. It aims at identifying a coverage of all possible threats [4] and preventing and/or mitigating the effects of threats and attacks on a software system. Several threat modeling methods exist, reviewed for example in [34], [37]. As part of all these methods, threat enumeration is at its core [6], which is traditionally carried out in brainstorming sessions. Current widespread threat modeling methods (such as STRIDE [17], OCTAVE [2], PASTA [35], etc.) are coarse-grained and require in-depth security knowledge. There is no detailed description of a procedure to support the brainstorming sessions, and no reference model to be used by such a procedure [16], [28]. Due to the lack of guidance, the lack of sufficiently formalized process, the high dependence on actors' knowledge and the variety of actors' background, these sessions are often conducted sub-optimally and require significant effort [9].

Thus, several research challenges have been recently identified [16], such as 1) developing a reference model, which makes it possible to share threat modeling artifacts in a standardized manner for the reuse, education, and benchmark and 2) defining a process that better supports the interactions among threat modeling participants, consequently, allowing a better knowledge reuse across projects, experts, and organizational boundaries. To improve current threat modeling processes and rise to the identified research challenges, we propose an asset identification process, to help participants collaboratively identify assets, which are significant for both business stakeholders and product team members, as well as for the security experts. This structured process employs a number of concepts and relations, which we organise into a reference model. Moreover, to increase the knowledge reuse degree and reduce the reliance on subjective experience, we propose to construct a vulnerable asset library as part of a threat library.

The paper is structured as follows: Section II presents the background of threat modeling processes and the motivation of this paper. In Section III, we present our approach, including the need of reworking on the *asset* concept, structuring the threat modeling knowledge into an asset-based reference model and defining an asset identification process. In Section IV, we show the application of our approach, including a library of vulnerable assets, which we extract from common security knowledge bases such as CAPEC by applying several heuristic rules. We also illustrate in this section a case study by firstly applying the Microsoft SDL threat modeling process, and then integrating our asset identification process into it to improve its results. Then we discuss the advantages and limitations of our approach. Related works are discussed in Section V. Finally, we conclude the paper in Section VI.

II. Background and Motivation

In this section we first define what the threat modeling is and show its importance in secure software development. Then we make an inventory of the current threat modeling processes and highlight their limitations. Finally, we identify several needs or requirements that remain to be satisfied, which point out our motivation.

A. Threat modeling definition

There are numerous definitions of threat modeling in literature, used in different and perhaps incompatible ways [37]. For our purpose, we define the threat modeling as a systematic process of identifying and analyzing threats (i.e. potential attacks), which involves the understanding of threat agents goals and adversaries actions in attacking a system, based on that system's assets [36].

Threat modeling can occur at any time during the software development lifecycle, but it's more efficient to be

performed during the early requirement and the architecture/design phases [14], because fixing an issue that involves reworking on a conceptual model rather than significant re-engineering can save cost, development time and protect the system from high impact attacks [33], [36].

Threat modeling process is also a collaborative process where participants include: business stakeholders; product-team members from all product development phases, such as enterprise, software and application architects, development leads, IT infrastructure specialists, engineers; security experts such as security analysts, security architects, threat modeling experts [31], [33].

Threat modeling is important because it helps in 1) identifying business-logic flaws and other critical vulnerabilities that expose core business assets, 2) enriching assessments with new potential attack vectors, 3) prioritizing the types of attacks to address 4) mitigating the risks more effectively and 5) fixing issues early in the development process [31], [34].

B. Threat modeling process

Several threat modeling processes including various activities are proposed in literature [4], [15] and widely used in industry (Microsoft [25], [27], [33], CIGITAL [31] and EMC [6]), as shown in Table I. We summarize the threat modeling process into four main phases:

1) **Asset Identification phase**: It is centered on identifying security goals, modeling domains (by characterizing the system, usually by decomposing it and describing its components and data flows using (annotated) architecture/design diagrams) and identifying valuable assets.

2) **Threat Enumeration phase**: It is focused on identifying threats, together with attackers (their motivation and skill) and vulnerabilities, and enumerating and documenting resulted threats. This phase is often conducted in brainstorming meetings, sometimes guided by a threat library.

3) **Threat Prioritization phase**: It is based on the result of threat enumeration, to rate threats and assess risks. This phase can be either considered as an internal or external activity [34].

4) **Mitigation phase**: It aims at resolving threats by proposing security mitigations and by verifying them.

It is worth noting that not all the approaches in Table I include the asset identification phase, which is nonetheless an important step. The activity of identifying asset is a bridge between domain modeling and threat identification, however, only a few works address this activity. Identifying assets is essential because it takes both into account the modeling of the domain under consideration, and the will of stakeholders to protect valuable elements. Without this step, the later threat enumeration and prioritization phases would be less efficient. Some approaches mention the activity of "identifying asset" [4], [15], however no detailed guidance or formalized process about how to systematically conduct this activity is proposed.

As a prerequisite for the threat enumeration (which is at the core of the threat modeling process [6], [16]), the quality of the asset identification impacts directly the threat enumeration and indirectly later phases. Therefore, the early phases (asset identification and threat enumeration) of threat modeling are crucial for the success of threat modeling. However, the activities in these phases are often conducted in brainstorming sessions [28], the results of which depend highly on the human expertise, experiences and collaboration, even with the support of such methods as STRIDE (a coarse-grained guiding method used largely in industry).

Conducting threat modeling thus requires ideally a sound knowledge of a system's technical domain and sufficient security expertise to consider both generic and specific attacks for various specific contexts [36]. With the threat modeling process going on, more and more security expertise is required in each phase. However, the need of security knowledge can "leave most 'off-the-street' developers estranged" [36], with the result that threat modeling is performed sub-optimally or with significant effort involved, or not performed at all. Even after security training, the threat modeling process is still difficult to execute [31].

C. Motivation

Threat modeling is still at a low level of maturity [16] and several key research challenges and/or requirements have been identified:

1) It is important to hold a successful brainstorming meeting [28], which is still a subjective and unstructured activity. It should follow a methodical approach in enumerating threats, while still letting participants think about the problem creatively. It thus needs a guidance that is more prescriptive, formal, reusable and less dependent on the aptitudes and knowledge of the participants. Meanwhile, the cause of the high number of overlooked threats is also worthy of investigating [25].

2) The current threat modeling processes require a certain security knowledge level, making it a non-trivial task for participants with limited security knowledge. Proposed widespread threat modeling methods (such as STRIDE [17], OCTAVE [2], PASTA [35], etc.) are abstract, coarse-grained and require in-depth security knowledge. Wrong decisions are thus made based on insufficient knowledge about the security domain, threats, possible countermeasures and the own infrastructure. An in-depth reason is that security terminology is vaguely defined. This leads to confusion among experts as well as the people who should be counseled and served [7]. There is thus a need to propose a method that can be easily used or understandable by security novices.

3) Moreover, a successful communication among threat modeling participants requires that they share their knowledge and points of view with as little bias and as few misunderstandings or confusion as possible [9]. Without a shared terminology communication, especially in a complicated domain like security, threat modeling cannot be successful [8]. Incorrect results could thus be caused by the misinterpretation of some template threats in the checklists. Therefore, there is a need of a common language or a common concept that can be understood by all participants.

Phase	Asset Identification			Threat Enumeration				Threat Prioritization		Mitigation	
Activity / Paper	Identify security goal	Model domain	Identify asset	Identify threat	Enumerate &document threat	Describe attacker	Identify vulnerability	Rate threat	Assess risk	Mitigation	Verification
Torr (2005) [33]		x		x						x	x
Shostack (2008) [27]		x			x					x	x
Scandariato (2013) [25]		x		x	x						
Beckers (2013) [4]		x	x	x	x	x					
Dhillon (2011) [6]		x		x					x	x	
Steven (2010) [31]	x	x		x			x				
Kamatchi (2016) [15]		x	x	x	x			x			

TABLE I: An inventory of threat modeling processes

III. Structuring the asset identification phase

Our proposal addresses the 3 preceding needs. It aims to structure the asset identification phase: to associate it with a well-defined process promoting the manipulation of a set of precise and well-structured concepts and exploiting, if needed, a security knowledge base to limit the negative impact of the lack of experience in security. These elements can be used to guide the actors during the brainstorming sessions. This proposal is based on a novel refinement of the concept of *asset* which we describe at first. Secondly, we show how we use this refinement to structure the universe of threat modeling knowledge using a reference model. This reference model is used to structure the common language of the information handled by all participants during this phase. But also it can be used as a language with which one can capitalize in a knowledge base the state of the art in security. We next present a process based on this reference model to lead the asset identification phase.

A. Reworking on the asset concept

Our proposal is based on the *asset* concept. This concept can be easily understood by business stakeholders and by product team members [19], [23]. It is naturally well-known by security experts. It can therefore act as a shared concept between all participants in the threat modeling process. It also provides a solid common base for further activities, especially threat enumeration.

There are numerous definitions of the *asset* concept in literature. For example, ISO 21827 [1] defines *asset* as anything that has value to the organisation, such as data, hardware, software or networks. Similar definitions focusing on the value of an asset, which can be subjective, commercial, and vary in a wide range, are presented in [24]. Several definitions look at assets from the attackers' point of view, defining them as the things that an attacker tries to steal, modify, or disrupt, and considering their relations with threats [33]. Other definitions consider the relations of assets with vulnerabilities/exposures/weaknesses and countermeasures [22].

These definitions are too generic, too abstract and too wide-encompassing; entities that have various natures can be included in these definitions. Such multiple overlapping definitions, including things attackers want, things stakeholders are protecting, and stepping stones, can "trip you up" [28]. It may thus lead to misunderstandings and confusion among the threat modeling participants.

We therefore consider two viewpoints for the *asset* concept:

The domain expert's viewpoint: We consider as *domain experts*, participants in the threat modeling process who are not security experts, such as: business stakeholders and product-team members from all product development phases, such as enterprise, software and application architects, development leads, IT infrastructure specialists and engineers. They deal with **Domain Asset (DA):** Anything that has value for them, towards the fulfilment of the function and goal of the system, together with the assurance of its properties. DA is artifact of a particular system architecture.

The security expert's viewpoint: We consider as *security experts*, participants in the threat modeling process who have sufficient security knowledge, such as: security analysts, security architects and threat modeling experts. They deal with **Vulnerable Asset (VA):** Anything that has value for them. It has vulnerabilities that can be menaced by threats. Hence it is the direct, core target of the attacker. If it is compromised, it can impact relevant domain asset (DA). Therefore, they need protection to reduce threats and prevent attacks. VA is system artifact appearing in more or less abstract attack patterns or vulnerability bases.

VA and DA are all system artifacts but appear in a different context: respectively in an attack pattern description and in a system architecture model. They can also have different abstraction levels. However, they may also include elements which are exclusive to any one of them. As such, the domain assets may include assets which may not be vulnerable, and therefore not identified as vulnerable assets. Similarly, vulnerable assets may include assets which are not used in the domain models. Determining the common assets between the domain and vulnerable assets is not trivial. However, it is essential, because they represent the domain assets that are also vulnerable and therefore need protection.

The domain assets that are also vulnerable assets constitute therefore a new type of asset, understandable by both the domain and security experts. We call this new type of asset **Vulnerable Domain Asset** (VDA). The VDA is therefore anything that has value for the domain expert, but also has vulnerabilities that can be menaced by threats. As it is a domain asset, it is also domain specific. Our VDA concept has precursors in the literature. For example, [28] remarks that the most common usage of asset in discussing threat models seems to be a marriage of "things attackers want" and "things you want to protect". However, in previous works, this idea is not further developed to show how to differentiate and organize different types of assets.

B. Structuring the threat modeling knowledge

Now that we have refined the *asset* concept, we propose to structure the universe of information manipulated during brain-

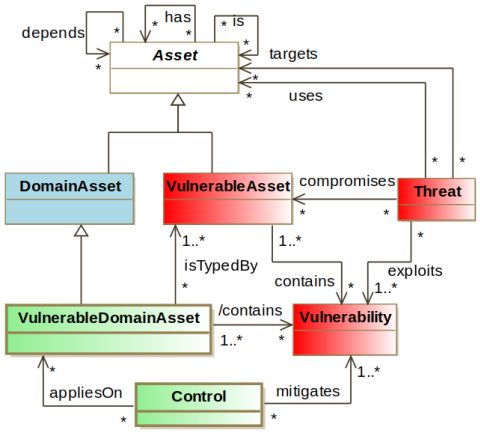

Fig. 1: Asset-based reference model

storming using a *reference model*. The definition of this model has two objectives: 1) fixing and structuring the discourse during brainstorming sessions and 2) allowing the capturing of security knowledge from the literature in a form which can then be reused during brainstorming. This reference model is presented in Figure 1. *Asset* is the core concept of our model. It is an abstract class. As discussed above, we specialize the concept of *Asset* into *DomainAsset* (DA) and *VulnerableAsset* (VA). The *VulnerableDomainAsset* concept (VDA) is a type of both *DomainAsset* and *VulnerableAsset*. Both *VulnerableAssets* and *VDA* can *have Vulnerabilities*, which can be *exploited* by *Threats*. Thus *Threats* can *compromise* VA and thus VDA. In its compromise actions, a *Threat* may *target* an *Asset* (both *Domain* and *Vulnerable*) *using* other compromised *Assets* in the process. To mitigate the *Vulnerabilities*, *Controls* can be *applied* on *VDA*.

Each *Asset* can have three relationships with other Assets: *is*, *has* and *depends*. The *is* relation captures the generalisation between *Assets* of different abstraction levels. It captures an iterative refinement of assets. For example, the domain experts define the list of domain assets coming from the domain architecture model. During design, they can progressively refine this list from more abstract assets to more concrete ones. Similarly, the security experts define an hierarchy from more abstract (coming from abstract attack pattern) to more concrete vulnerable assets. Moreover, an *Asset* may be composed of other *Assets*. We model this through the *has* relation. We also introduce the *dependency* relation between *Assets*. A dependency exists between two elements if changes to one element (the supplier) may cause changes to the other (the client) [11].

These three relationships (generalisation, composition and dependency) are very common in system architecture modeling. It is worth noting that this kind of modeling promotes a data structure similar to that of a B-tree [3], even if other data structures are also possible, such as class diagram. We choose B-Tree structure because it can be easily coupled with and extended from Attack Tree, as we deal with security aspect. A B-tree is a tree data structure where each level can have one or more *children* nodes. Each node may be thought of as a kind of list, containing several entities called *keys* (related to the origin of B-trees for databases). In our case, the *Assets* related by an *is*, are similar to the *children* nodes of a B-tree. For example, in Figure 2, VA2 and VA6 are *children* of VA1, and VA3 is a *child* of VA2. The *Assets* related by *has* and *depends*, correspond to the *keys* of a B-tree. VA4 and VA5 are keys related to VA2, related respectively by *has* and *depends* relations. In our data structure, we just take inspiration form the idea of B-trees, but are not interested in their properties, such as self-balancing. We choose B-Tree structure because it allows to show all the three relations of different dimensions inside one tree. More precisely, B-Tree allows showing the generalization relation vertically, and showing composition and dependence relations horizontally inside each child nodes, to align with the relations of different dimensions among different assets. This similarity may also enable a higher degree of automation for the asset identification process in the future. Moreover, this is close to the structure of existing security knowledge bases, such as CAPEC, facilitating extraction from them (cf. Section IV-A).

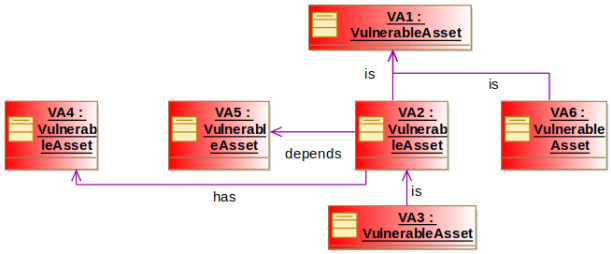

Fig. 2: Vulnerable asset B-tree

To reduce the level of security expertise required, threat modeling can be supported by threat libraries (structured or unstructured lists of threats), which have been found particularly effective in industry scenarios [36]. However, non-security experts, such as domain experts, have to be trained to better use threat libraries, as they require a minimum security knowledge to understand security jargon. Therefore, we think that it is useful to construct a vulnerable asset library, which can enrich the threat library. To help the asset identification process, we thus propose to construct a library of vulnerable assets. The VA library aims to classify a wide variety of abstract, system- and technique-independent VA, which keeps the asset identification and threat enumeration manageable, increases the VA library's applicability and reusability, and makes it both more practical and more useful for security novices and experts alike. For the library to be well integrated with the asset identification process, we propose that the library and the reference model presented follow the same structure for the VA. Part of construction process of the VA library is presented in Section IV-A.

C. Defining the asset identification process

After refining the *Asset* concept and proposing an asset-based reference model to structure the knowledge, we present a process to help actors in the identification of threats targeting the assets. This process is shown in Figure 4 and the general view is summarized in Figure 3. This process can be launched

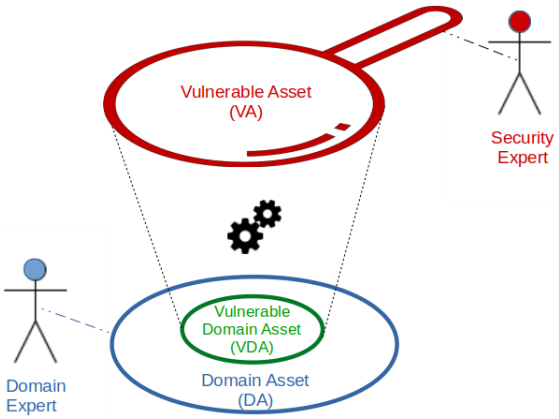

Fig. 3: General view of asset identification

regardless of the software development stage and therefore on more or less abstract models.

On one hand, DA, obtained from domain models such as enterprise, system and software architectures, is structured by relationships such as generalization, composition and dependency. On the other hand, security experts identify Vulnerable Assets (VA) relevant to the types of elements present in the model being designed. This list of VA can be populated from the security experts' knowledge, as well as be extracted from common security knowledge databases, thus promoting reuse. This extraction is a non-trivial process, involving threat libraries, attack patterns (e.g. CAPEC), attack trees, vulnerabilities, etc. Thus, we promote the setting up a VA library synthesizing current knowledge of the field in a format that respects our reference model.

Since DA and VA are similarly structured by the *is*, *has* and *depends* relationships defined in the reference model, the goal of the asset identification process is to bridge the gap between these two sets of assets (cf. Figure 3) and identify VDA (i.e. Domain Assets which are also Vulnerable). The security experts project or instantiate these VA on the DA. A comparison is made by actors to identify if a mapping occurs between VA and DA. If mappings are identified, they represent the VDA. It is therefore noteworthy that the matching process instantiates *abstract* VA into *concrete* VDA. Discovering the VDA further enables identifying security mitigations (i.e. controls), based on their vulnerabilities. In this way, our approach uses domain-independent, general, security threat and attack knowledge to identify and protect domain-specific VDA.

Figure 4 illustrates a fine-grained asset identification process. It takes as inputs a DA list and a VA tree. The DA list is a result of domain experts identifying assets specific to their domain. The VA tree results from the security experts using their knowledge to identify generic vulnerable assets, and it can be enriched with information extracted from security knowledge bases or a VA library.

The asset identification process can traverse the vulnerable asset tree (respecting to B-tree) either in a depth-first or in a breadth-first manner. In this paper, we choose to present a breadth-first strategy. As such, the process *selects* the VA tree children situated at the current vertical "i" level (i.e. all the VA linked through an *is* relation to the VA of the previous parent level). For instance, when considering the example in Figure 2, concerning level "i=1", the children are VA2 and VA6. For each VA child, the domain and security experts *compare* its syntactic and semantic similarity with each DA in the current domain asset list. If a VA_k, from the list of VA level i children, is found similar with a DA_j, from the DA list, further similarities are searched. For this, the VA tree is traversed horizontally, and the keys attached to that VA_k are *selected* (i.e. the VA linked through *has* and *depends* relations). Let us suppose that for the Figure 2 example, VA2 is found similar with a DA, then its key list containing VA4 and VA5 is *selected*.

The domain experts select among VA_k keys those which are involved in the domain. VA_k keys that are involved in the domain, discovered at this "i" iteration, are added to the current DA list, enriching the DA list for the next iteration. As they are initially VA, but also in the domain, they are actually VDA, and therefore are also added to the VDA list. Let us suppose that in our example VA4 is a key that is involved in the domain. At the end of the "i" iteration, the DA list additionally contains VA4 and the VDA list contains VA2 and VA4. Then, if no more VA_k key is found being involved in the domain, the process advances to the next VA tree level (i.e. "i=i+1"), until there are no more levels (i.e. "i==n"). The DA and VDA list are enriched with the iteration of each "i" level.

Once the VDA list has been enriched, it is used as a bridge towards threat enumeration. Security and domain experts may use it to propose security mitigations to the identified vulnerabilities.

IV. Case Study

In this section we want to highlight the gain obtained through the use of our approach. To do this, we proceed as follows: we first present a process of constructing a vulnerable asset library by leveraging CAPEC and respecting the B-tree structure of the reference model, in order to show the possibility of knowledge reuse. Secondly, we illustrate a case study by firstly applying the Microsoft SDL threat modeling process, and then integrating our asset identification process into it to improve its results. We believe that the integration of our asset identification process into the Microsoft process as a complementary step can improve the detection of more relevant threats. The advantages and limitations of the resulting enriched process, compared to those of the sole Microsoft process, are discussed at the end of this section.

A. The process of constructing a vulnerable asset library by leveraging CAPEC

As we mentioned before, Vulnerable Asset (VA) represents security experts' viewpoint and it is domain-independent. Therefore, VA can be extracted from existing security knowledge bases for the reuse in different contexts or domains. In this section, we stress the importance and reusability of this extraction and present part of extraction rules by leveraging

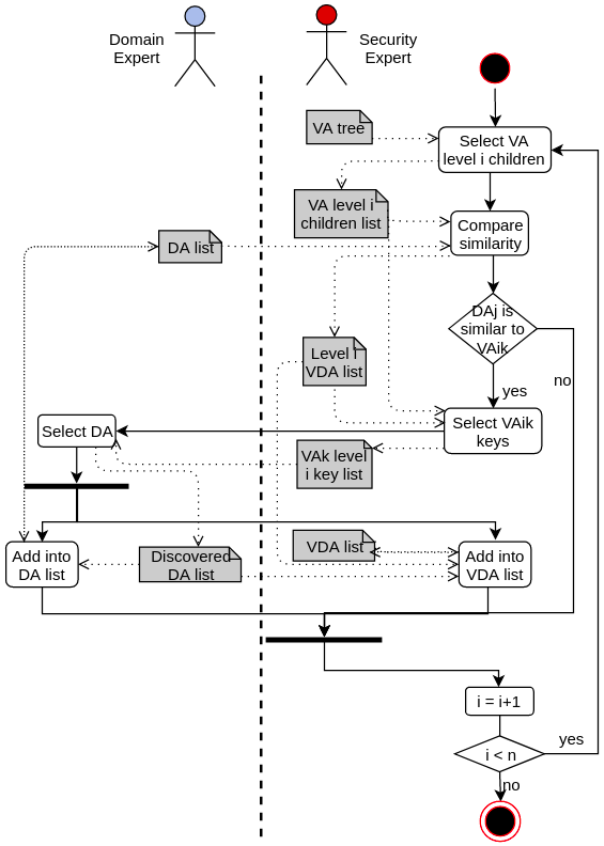

Fig. 4: Asset identification process

well-known attack pattern knowledge bases such as CAPEC and by respecting the B-tree structure.

Attacks are possible realisations of threats [30]. Therefore attack descriptions can be useful in enumerating threats. To construct this library, we can leverage existing attack databases such as CAPEC [20], OWASP [10] and ATT&CK [21]. However, they are defined in natural language and possibly ambiguous, which make them difficult to be processed automatically. At the current state of advancement, we identify a number of heuristic rules which can be enriched in the future. These rules help partially extract VA and relations between these VA that can be compromised by threats.

We show these extraction rules by leveraging CAPEC, which is one of the most popular and structured attack databases. In CAPEC, the attacks belong to different levels of abstraction: *view, category, meta, standard and detailed*. We focus on the *meta, standard and detailed* abstraction levels, because *view* and *category* levels are too abstract to be reused effectively. 1) A *meta* attack pattern is "an abstract characterization of a specific methodology or technique used in an attack", and "a generalization of related group of *standard* attack patterns". 2) A *standard* attack pattern is "focused on a specific methodology or technique used in an attack". 3) A *detailed* attack pattern "provides a low level of detail, typically leveraging a specific technique and targeting a specific technology, and expresses a complete execution flow". *Detailed* attack patterns are more specific than *meta* and *standard* attack patterns. The links between these abstraction levels are modeled through "*childOf*/*parentOf*" relations. This hierarchical attack/threat structure can help us identify *is* and *has* relations between VA. Moreover, there are also relations of "*canFollow*/*canPrecede*" between attacks/threats in CAPEC, which can help us identify *depends* relation between VA.

Based on CAPEC attack natural language descriptions, we define several VA extraction rules:

1) If the name of attack pattern contains the keyword "contaminate", or "poison", or "leverage", or "manipulate", or "abuse", or "exploit" or "misuse", then the noun set after any of these keywords is selected as a vulnerable asset (VA). For example, for the *detailed* attack pattern "Poison web service registry" (CAPEC-51), the "web service registry" is a vulnerable asset;

2) If the name of attack pattern contains the keyword "manipulation", or "poisoning", or "tampering" or "alteration", then the noun set before any of these keywords is extracted as a vulnerable asset (VA). For example, for the *standard* attack pattern "Web service protocol manipulation" (CAPEC-278), the "web service protocol" is a vulnerable asset;

3) If the name of attack pattern contains the keyword "injection", or "inclusion" or "insertion", then the noun set before any of these keywords is selected and we add the literal "Untested" before and "Input" after this noun set, the whole literal word is considered as a VA. For example, for the *standard* attack pattern "XML injection" (CAPEC-250), "XML" is selected and added by the above prefix and suffix. As a result, "UntestedXMLInput" is a vulnerable asset.

There are three possible relations (*is, has, depends*) between VA, as mentioned in Section III-B. By leveraging CAPEC, we can also extract the relations between VA.

4) The "childOf" relation between two attack patterns is translated into either "*is*" or "*has*" relation between two corresponding VA, because "ChildOf" in CAPEC can present either a specialisation or a decomposition relation. For example, on one hand, the "SOAP" VA extracted from the *detailed* attack pattern "SOAP Manipulation" (CAPEC ID 279), *is* a type of "Web Services protocol" VA. On the other hand, the "XML" VA *has* "DTD", "XPath" and "XQuery" VA, extracted respectively from three *detailed* attacks (CAPEC IDs respectively 228, 83, 84). Therefore, the reasoning about the decision comes from the security experts who extract VA;

5) The "canFollow" relation between two attack patterns is translated into *depends* relation between two relevant VA, because if asset A_a is compromised by an attack/threat T_a, then a threat T_b, which can follow T_a, can compromise asset A_b, therefore asset A_b *depends* on asset A_a.

These rules allow us to extract VA from attack/threat patterns. For each extraction, the relation between the threat and the VA is stored. This allows to later find all the threats that compromise the same VA. In this way, our library contains the information about VA and threats that compromise them. At the current state of this paper, the VA extraction process is conducted manually by the first author. We believe that this extraction process can be implemented using techniques

such as parsing, text mining and/or bash/sh scripting to allow automation.

The VA library, as a part of threat library, lightens the dependency on attack knowledge. It aims to be utilized by both security and non-security experts. Therefore, the construction of the VA library can satisfy to the requirement 2 in Section II-C.

B. Illustration of the process integration

In this part, we first illustrate a case study using the Microsoft SDL threat modeling process to enumerate threats. As we will see, this process lacks of an "identifying asset" activity, which is a bridging step between the "domain modeling" and the "threat identification" activities. Therefore, we then illustrate the integration of our asset identification process into the Microsoft process as a complementary step to improve the detection of relevant threats.

1) Microsoft SDL threat modeling process illustration: Microsoft SDL threat modeling process is based on STRIDE, which is currently the most mature threat modeling method [26], and is implemented with the SDL threat modeling tool, which is available online [5].

There are four steps, described in [27], to conduct threat modeling process in Microsoft: 1) diagramming (by applying Data Flow Diagram - DFD), 2) threat enumeration (by applying STRIDE), 3) mitigation and 4) verification. As our paper aims to support the threat enumeration, we only present the first two steps. Microsoft implements the SDL threat modeling process into Microsoft threat modeling tool. This tool provides predefined DFD elements, as well as allowing users create new templates containing stencils (new elements) and threat types. We illustrate a case study with the tool.

As to the case study, we take the example of Web Sphere 7.0 application server, which is a software framework that hosts java-based web applications, allowing deploying and managing applications ranging from simple Web sites to powerful on-demand solutions. It is architected as a distributed computing platform that could be installed on multiple operating system instances, collectively referred to as a WebSphere cell. Its configuration information are tracked in XML configuration files throughout the cell.

The Microsoft process begins by characterizing the software or system (Web sphere 7.0 in our case), by decomposing it and describing its components and data flows, using DFD. There are four types of elements in DFD: external entity, process, data flow and data store. An excerpt of a possible decomposition of the Web Sphere 7.0 application server is presented in Figure 5, obtained using the Microsoft tool. It is decomposed into a process called "web service" and a data store termed "configuration file". "Web service" interacts with "configuration file" through a "general data flow". The above three DFD elements are predefined by the tool. Further DFD modeling of the case study is not presented here for the sake of readability and space reasons.

The next phase is the threat enumeration, which is conducted in a brainstorming meeting guided by "STRIDE by elements", supported by the threat list generated automatically by the tool. The threat list contains the threats that menace each DFD element or a group of DFD elements. As shown in Figure 5, the tool has found two threats concerning "Web Service", "Configuration File" and their interaction: 1) Spoofing of destination data store configuration file (belonging to the threat category of Spoofing) and 2) potential excessive resource consumption for web service or configuration file (Denial of Service).

In the brainstorming meeting, participants generally discuss and find out potential threats belonging to six threat categories of STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) that can threaten the actual DFD element. At the current state of the work, we have not discussed with Microsoft SDL threat modeling experts. The quality and quantity of the results of the brainstorming meeting depends highly on participants. It is a highly subjective activity, the results of which are not reproducible. This makes it difficult for us to compare the brainstorming activity with our asset identification process. However, we believe that our process can help structure this activity, which we discuss in Section IV-C.

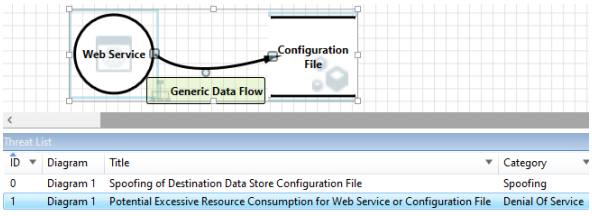

Fig. 5: Applying Microsoft SDL threat modeling tool

2) Integrating our process into Microsoft SDL threat modeling process: The Microsoft SDL threat modeling process begins by modeling the domain (by applying DFD), before identifying threats (by applying STRIDE). As we noted when discussing Table I, the Microsoft SDL process does not contain the activity of "identifying asset", which is a bridging step between "modeling domain" and "identify threats". Therefore, we present the integration of our asset identification process into the Microsoft SDL threat modeling process in the aim at discovering more relevant threats.

Based on the DFD model in Figure 5, we observe that there is a loss of information during the domain modeling: the "configuration document" is of the XML type. This information may be critical for threat enumeration. A reason for this loss of information is that XML document is not predefined by the tool.

Therefore, we add the "XML document" in the domain model for our asset identification process, based on the DFD modeled in Figure 5, in order to fill the gap between domain modeling and threat enumeration. We describe the domain model of the case study using UML class diagram. Other modeling languages can be used as well. The domain asset model is presented in Figure 6. Conforming to the description of "Web Sphere 7.0 application server", the "Configuration Document" is of type XML and is contained in the "Web Sphere Server", together with "Web Service".

To apply the asset identification process, on one hand, the domain experts produce the domain asset list, part of which is shown in Figure 6. The *WebSphere7.0* contains, among other components, a *ConfigurationDocument* and a *WebService*. These correspond to concrete (architecture) elements. The *ConfigurationDocument* can be generalized into an abstract (architecture) element *XMLDocument*.

On the other hand, the security experts use a vulnerable asset B-tree from the VA library (constructed using the heuristic rules presented in Section IV-A), part of which is presented in Figure 7. This vulnerable asset B-tree begins by a root called *VARoot* (VA_1), which is an artificial root to start the process and can be specialized by any of its children VA. In Figure 7, the VA UntestedCommandInput, UntestedCodeInput, UntestedXMLInput, UntestedSQLInput, UntestedDTDInput and UntestedXPATHInput are respectively extracted: from the *meta* attack patterns Command Injection (CAPEC-248) and Code Injection (CAPEC-242); from the *standard* attack patterns XML Injection (CAPEC-250) and SQL Injection (CAPEC-66); and from the *detailed* attack patterns DTD Injection (CAPEC-228) and XPATH Injection (CAPEC-83), respecting the rule 3 in Section IV-A. Other VA are omitted in this paper due to limited space.

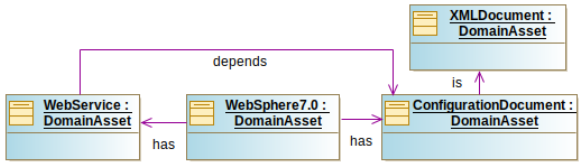

Fig. 6: An Excerpt of Domain Assets

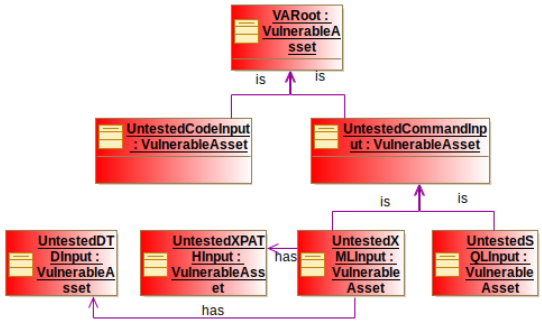

Fig. 7: An Excerpt of Vulnerable Asset Tree

We illustrate the asset identification process based on the VA tree in Figure 7. We initialize the process with i = 1, in this case, n is equal to 3. In the following, we illustrate each task of the process of Figure 4:

1) Select VA level i children: At the beginning of the process, i = 1, which is the *VARoot*. In our case, VA level 1 children are *UntestedCommandInput* (VA_{11}) and *UntestedCodeInput* (VA_{12});

2) Compare similarity: With the DA list provided by domain experts, security experts need to compare syntactical and semantic similarity between a vulnerable asset and a domain asset. Among the four domain assets *XMLDocument* (DA_1), *WebSphere7.0* (DA_2), *ConfigurationDocument* (DA_3) and *WebService* (DA_4), there is no similarity found when compared with *UntestededCommandInput* (VA_{11}) and *UntestedCodeInput* (VA_{12});

3) Therefore, for all DA_j, none of them is similar to VA_{1k}, which are children of VA level 1. In this case, i increments, now i is equal to 2, which is still lower than 3;

4) Select VA level i children: As no similarity is found from the upper level, the process advances to the lower level of the VA tree. For level i=2, there are two VA UntestedCommandInput and UntestedCodeInput, as shown in Figure 7. For the VA UntestedCommandInput, there are two children. Therefore, the VA level 2 children list contains UntestedXMLInput and UntestedSQLInput;

5) Compare similarity: *UntestedXMLInput* (VA_{21}) is found both syntactically and semantically similar to *XMLDocument* (DA_1);

6) Select VA_{ik} (VA_{21} in our case) keys: The process continues to search VA_{ik} keys. For our example, the VA_{21} key list contains *UntestedDTDInput* and *UntestedXPATHInput* (related to the *has* relation);

7) Select DA: Domain experts study if the domain model involves any of the assets which are in the VA_{21} key list, but have not yet been identified as domain assets. For the example in Figure 6, the domain experts realize that the XMLDocument DA does involve a DTD, which in this case can be manipulated by the user (attacker), without any intermediary tests. Possible impacts include that XML parsers, which process the DTD, consume excessive resources, resulting in resource depletion. Therefore, the *UntestedDTDInput* VA is a VDA that is vulnerable and involved in the domain;

8) Add into DA list: The domain experts add *UntestedDTDInput* to the DA list;

9) Add into VDA list: The security experts add as well, *UntestedDTDInput* to the VDA list. The VDA list for this example is shown in Figure 8. The same reasoning applies to other keys, which we do not detail here for the reason of readability;

10) After adding the discovered DA and VDA into each list, i increments, now i is equal to 3, which is not lower than n (=3 initially). Therefore, the process stops.

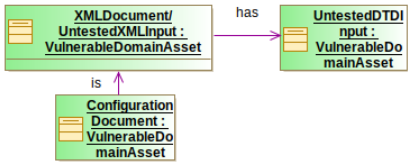

Fig. 8: An Excerpt of Vulnerable Domain Assets

As a result of this process illustration, we discovered the VDA *UntestedXMLInput* and *UntestedDTDInput*. *UntestedDTDInput* is not initially annotated as DA by domain experts. These two VDA are initially VA. Therefore, the threats that compromise these VA can be retrieved using the VA library presented in Section IV-A. As such, for *UntestedXMLInput*, the following 13 threats are identified: 1) XML Schema Poisoning (CAPEC-146), 2) XML Ping of the Death (CAPEC-147), 3) XML Entity Expansion (CAPEC-197), 4) XML Entity Linking (CAPEC-201), 5) Spoofing of UDDI/ebXML

Messages (CAPEC-218), 6) XML Routing Detour Attacks (CAPEC-219), 7) XML External Entities Blowup (CAPEC-221), 8) XML Attribute Blowup (CAPEC-229), 9) XML Nested Payloads (CAPEC-230), 10) XML Oversized Payloads (CAPEC-231), 11) XML Injection (CAPEC-250), 12) XML Quadratic Expansion (CAPEC-491) and 13) XML Flood (CAPEC-528). For *UntestedDTDInput*, there is only one threat identified: DTD Injection (CAPEC-228).

C. Discussion

1) Case study discussion: Whereas the Microsoft SDL threat modeling tool identified two threats for the case study, our asset identification process identified 14. Among these 14 threats, Spoofing of UDDI/ebXML Messages (CAPEC-218) and XML Routing Detour Attacks (CAPEC-219) belong to the same threat category of Spoofing, as the Spoofing of destination data store configuration file; Similarly, XML Flood (CAPEC-528), XML Ping of the Death (CAPEC-147), XML Nested Payloads (CAPEC-230), XML Entity Expansion (CAPEC-197), XML Quadratic Expansion (CAPEC-491), XML Oversized Payloads (CAPEC-231), XML Entity Linking (CAPEC-201), XML Attribute Blowup (CAPEC-229) and XML External Entities Blowup (CAPEC-221) belong to the same threat category Denial of Service, as the potential excessive resource consumption for web service or configuration file. As we can see, we identified more detailed threats comparing to the Microsoft SDL threat modeling tool. Moreover, our asset identification process discovered new XML Schema Poisoning (CAPEC-146) and XML Injection (CAPEC-250) threats, which are not found by the Microsoft tool.

A number of threats identified by our process come from proposing VA as new DA, for example *UnTestedDTDInput*. This enables identifying in-depth domain assets (that are also vulnerable), which otherwise may be overlooked.

As we can see, by integrating our asset identification process, we have found 14 threats, whereas sole Microsoft SDL threat modeling tool has found only 2 threat categories, which need further discussion and clarification during the brainstorming meeting. The average number of overlooked threats is very high as mentioned in [25], there is thus no guarantee that the brainstorming meeting can cover all 14 threats that we have found, because it depends highly on the security expertise, experiences and creativity of participants. The result of our asset identification process can thus be used as a checklist included in the brainstorming meeting to offer a guidance, to be complementary with Microsoft DFD and STRIDE based approach.

The DFD is data-centric, it focuses on the data flow between components of the same abstraction level. Each new template, which can be created with the tool, can represent a new abstraction level. However, DFD does not allow presenting **relations** among elements of **different abstraction levels**. That is why we model the case study with UML class diagram, because it allows modeling elements with different abstractions levels.

To integrate DFD into our process, the four DFD element types can be mapped to our asset reference model. As shown in Figures 5 and Figure 6, the process "web service" is mapped into the *WebService* domain asset, the data store "configuration file" is mapped into the domain asset *ConfigurationDocument*, and the "general data flow" is mapped into *depends* relation to show the interaction. The DFD diagram containing these three elements is mapped into the domain asset *Websphere7.0* together with *has* relations.

A limitation of this case study would be that we have not compare our asset identification process results with that of a real brainstorming meeting by industrial participants. This would be a future work to validate our approach.

2) General discussion: As we have seen in Section II, most of the existing threat modeling processes do not detail the asset identification phase. They usually consider it to be done through a discussion, usually of a non-structured, brainstorming type. The quality and quantity of the result of brainstorming meeting depends highly on participants. Moreover, such a discussion is highly creative and involves an important cognitive charge. By proposing a structured and detailed asset identification process together with the asset reference model, we help structure and guide this phase, which satisfies the Requirement 1 in Section II-C. This asset identification process can be reused in different domains, as the VA library contains VA that is domain-independent. It is worth noting that several activities of our asset identification process still need human expertise, such as "similarity comparison" and "search if a VA is involved in the domain", these two human tasks pose yet a much easier cognitive load than that of the entire brainstorming.

Other problems encountered in non-structured brainstorming sessions are that some details or system parts are overlooked, or the stakeholder input is not captured accurately. Hence, more in-depth threat modeling is typically performed afterwards by a security expert in isolation, which can be error-prone, as it is performed by manually iterating through a model, and with a lack of specific domain knowledge, such as a particular technology used in the system [16]. Our asset-based reference model can help consider both domain specific knowledge, by instantiating domain assets, and security knowledge, by extracting vulnerable assets. This two knowledge is shared by vulnerable domain assets (VDA), which can be established as a common vocabulary that can be understood by both experts, responding to the Requirement 3.

The concept of *asset* is easily understandable by non-security experts compared to the concept of *threat* together with that of *attack technique*. Identifying VA that can later derive threats thus helps bridging the gap during the collaboration between domain experts and security experts, satisfying Requirement 2.

V. Related work

In this section, we investigate existing works with a specific focus on the asset identification to deal with security issues, and compare them with our proposition in threat modeling.

Then, as the purpose of our proposition is to support collaboration between participants, we compare our study with other works focusing on collaboration in threat modeling.

A. Asset identification

Asset identification is an important step in numerous risk assessment (including threat modeling) methods, reviewed and compared by [13], [32], such as EBIOS, MEHARI, OCTAVE, IT-Grundschutz, MAGERIT, CRAMM, HTRA, NIST 800-30, RiskSafe Assessment and CORAS. For some of them, the concept of *asset* is defined very largely, rather vaguely, as anything that can have value to the organisation . Other methods try to separate the *asset* concept into several types, e.g. EBIOS into primary and supporting, or the ISSRM model into business and IT asset, the HERMENEUT approach [12] into tangible asset and intangible asset, etc. These separations help little, if any, the next phase of threat enumeration, while our approach does, because it considers the different perspectives between domain and security experts.

[4] proposes the notion of "secondary asset", the harm of which can cause harm to a "primary asset". This is captured by our reference model through the *depends* relation. In our case, the "supplier" is equivalent to the "secondary asset", and the "client" depending on the "supplier" is equivalent to the "primary asset". However, we can model extended chains of dependency relations between several assets, whereas the "secondary asset" does not allow this.

[19] proposes a security repository meta-model to store all the reusable elements. They add several concepts including "asset", based on the work of [29]. They indicate that the asset can be valuable or critical, but also vulnerable. However, they don't detail more about how to systematically distinguish each type of asset.

[23] identifies assets in the software architectural model, by mapping them from a system or organizational level. Their identification process is therefore focused on tracing assets from a development phase to another, whereas our identification process matches two different viewpoints.

B. Bridging the gap during the collaboration

[9] uses *anecdotes* and *scenarios* to express security knowledge and to reason about security in order to facilitate the communication among different stakeholders. *Anecdotes* are frequently used to communicate knowledge about real, concrete and specific security issues. However, *Scenarios* are even more widely used as a means of communicating security concepts, reasoning about security principles and justifying viewpoints. Weaknesses of *anecdotes* and *scenarios* are related to the difficulty in generalising their information content. That is to say that *anecdotes* and *scenarios* contain highly specific descriptions of particular events in a system, whereas security needs have to encompass the system as a whole. Yet, this approach only deals with requirements models, whereas ours may involve domain models from any and all phases of the development lifecycle. Moreover, the security details involved are fine-grained and difficult to generalise, whereas our reference model and process are aimed to be reused and deal with multiple levels of security abstraction.

[8] proposes a security ontology to resolve the communication problem. They took into account the entire infrastructure as asset which is physical and belongs to business domain. The ontology guarantees shared and accurate terminology in order to reduce misunderstandings. Comparing to their approach which aims at replacing the security expert, we introduce a collaborative process. As our abstract concept *asset* can be refined into domain asset (which can be understood by domain experts) and vulnerable asset (understood by security experts), the projection of general VA on DA, resulting VDA (understood by both), creates the possibility for better collaboration.

VI. Conclusion and Future Direction

Threat modeling is a result of a collaborative process involving many actors from different backgrounds. Despite its importance, the collaboration between domain and security experts to bridge the gap between domain modeling and threat enumeration phases is not trivial. One of the main reasons is that threat identification and enumeration is often a challenging task for non-security experts. Thus, domain experts have to rely on threat modeling processes, which may quickly turn into a complex task when these processes lack guidance and formalisation.

To address this limitation, we propose a reference model and a systematic asset identification process to facilitate the collaboration between actors. As a result, pertinent assets such as Vulnerable Assets are structured and Vulnerable Domain Assets are identified to improve the threat enumeration phase. Then, we have discussed how the proposed approach could be applied to structure the security knowledge base (CAPEC) and how the proposed process could be integrated with, and complementary to, the Microsoft SDL threat modeling process, using an appropriate case study. Results show the usefulness of our findings in identifying new assets and threats, and in bridging the gap between the domain and the security experts through the formalisation of the brainstorming activity.

The approach presented in this paper can be extended to become an aid system for the experts mentioned above, thus strengthening the bridge which facilitates their collaboration. To achieve such an aid system, the following objectives will have to be achieved, among others: 1) automating the security knowledge base extraction to offer appropriate guidelines to domain experts with modest security expertise; 2) proposing a semi-automatic assistance based on the formalised reference model and the structured process that we are proposing, in order to suggest possible attack mitigation and/or security controls to the domain experts. These two objectives guide our future work.

References

[1] ISO/IEC 21827:2008. Information technology – security techniques – systems security engineering – capability maturity model, 2008.

[2] Christopher J. Alberts, Audrey J. Dorofee, James Stevens, and Carol Woody. Introduction to the octave approach. 2003.

[3] R. Bayer and E. McCreight. Organization and maintenance of large ordered indices. New York, NY, USA, 1970. Association for Computing Machinery.

[4] K. Beckers, D. Hatebur, and M. Heisel. A problem-based threat analysis in compliance with common criteria. pages 111–120, 09 2013.

[5] Microsoft Corporation. Sdl threat modeling tool. security development lifecycle., July 2018.

[6] D. Dhillon. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security Privacy*, 9(4):41–47, 2011.

[7] Marc Donner. Toward a security ontology. *IEEE Security and Privacy*, 1(3):6–7, May 2003.

[8] A. Ekelhart, S. Fenz, M. Klemen, and E. Weippl. Security ontology: Simulating threats to corporate assets. pages 249–259, 12 2006.

[9] I. Flechais and A. Sasse. Stakeholder involvement, motivation, responsibility, communication: How to design usable security in e-science. *Int. Journal of Human-Computer Studies*, 67:281–296, 04 2009.

[10] The OWASP Foundation. Owasp attack list, 2017.

[11] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Object Technology Series. Addison-Wesley, Boston, MA, 3 edition, 2003.

[12] E. Frumento and C. Dambra. The role of intangible assets in the modern cyber threat landscape: the hermeneut project. 5:2019, 02 2019.

[13] D. Gritzalis, G. Iseppi, A. Mylonas, and V. Stavrou. Exiting the risk assessment maze: A meta-survey. *ACM Comput. Surv.*, 51(1):11:1–11:30, January 2018.

[14] Michael Howard and Steve Lipner. *The Security Development Lifecycle*, volume 34. 06 2006.

[15] R Kamatchi and Kimaya Ambekar. Analyzing impacts of cloud computing threats in attack based classification models. 2016.

[16] Y. Koen, H. Thomas, V. Dimitri, S. Laurens, W. Kim, and J. Wouter. Threat modeling: from infancy to maturity. *New Ideas and Emerging Results, ICSE*, 2020.

[17] Loren Kohnfelder and Praerit Garg. The threats to our products. *Microsoft Interface, Microsoft Corporation*, 33, 1999.

[18] Gary McGraw. *Software Security: Building Security In*. Addison-Wesley Professional, 2006.

[19] D. Mellado, E. Fernández-Medina, and M. Piattini. A common criteria based security requirements engineering process for the development of secure information systems. 29(2), 2007.

[20] MITRE. Common attack pattern enumeration and classification., 2007.

[21] MITRE. Attck matrix for enterprise, 2015.

[22] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood. Exploring software security approaches in software development lifecycle: A systematic mapping study. *Comp. Stand. & Int.*, 50:107–115, 2017.

[23] T. Rauter, A. Höller, J. Iber, and C. Kreiner. Asset-centric security risk assessment of software components. In *Workshop on MILS: Architecture and Assurance for Secure Systems*, 01 2016.

[24] Keunwoo Rhee, Dongho Won, Sang-Woon Jang, Sooyoung Chae, and Sangwoo Park. Threat modeling of a mobile device management system for secure smart work. *Electronic Commerce Research*, 13, 09 2013.

[25] R. Scandariato, K. Wuyts, and W. Joosen. A descriptive study of microsoft's threat modeling technique. *Requirements Engineering*, 20, 06 2013.

[26] N. Shevchenko, T. A Chick, P. O'riordan, Thomas P. Scanlon, and C. Woody. Threat modeling: a summary of available methods. *Software Engineering Institute. Carnegie Mellon University*, 2018.

[27] Adam Shostack. Experiences threat modeling at microsoft. 01 2008.

[28] Adam Shostack. *Threat Modeling: Designing for Security*. 2014.

[29] Guttorm Sindre and Andreas Opdahl. A reuse-based approach to determining security requirements. 05 2003.

[30] W. Stallings and L. Brown. *Computer Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2014.

[31] J. Steven. Threat modeling - perhaps it's time. *IEEE Security Privacy*, 8(3):83–86, May 2010.

[32] A. Syalim, Y. Hori, and K. Sakurai. Comparison of risk analysis methods: Mehari, magerit, nist800-30 and microsoft's security management guide. In *Int. Conf. on Availability, Reliability and Security*, 2009.

[33] P. Torr. Demystifying the threat modeling process. *IEEE Security Privacy*, 3(5):66–70, Sep. 2005.

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

[35] Tony UcedaVelez. Real world threat modeling using the pasta methodology. *OWASP App Sec EU*, 2012.

[36] Anton V. Uzunov and Eduardo B. Fernandez. An extensible pattern-based library and taxonomy of security threats for distributed systems. *Computer Standards Interfaces*, 36(4):734 – 747, 2014.

[37] Wenjun Xiong and Lagerström Robert. Threat modeling – a systematic literature review. *Computers Security*, 84:53–69, 03 2019.