# Model the System from Adversary Viewpoint: Threats Identification & Modeling

Muhammad Sabir Idrees[1], Yves Roudier[2], and Ludovic Apvrille[3]

[1] Telecom Bretagne, France, muhammad.idrees@telecom-bretagne.eu,
[2] EURECOM, France, yves.rouider@eurecom.fr
[3] Telecom ParisTech, France, ludovic.apvrille@telecom-paristech.fr

**Abstract.** Security attacks are hard to understand, often expressed with unfriendly and limited details, making it difficult for security experts and for security analysts to create intelligible security specifications. For instance, to explain "Why" (attack objective), "What" (i.e., system assets, goals, etc.), and "How" (attack method), adversary achieved his attack goals. We introduce in this paper a security attack meta-model for our SysML-Sec framework [17], developed to improve the threat identification and modeling through the explicit representation of security concerns with knowledge representation techniques. Our proposed meta-model enables the specification of these concerns through ontological concepts which define the semantics of the security artifacts and introduced using SysML-Sec diagrams. This meta-model also enables representing the relationships that tie several such concepts together. This representation is then used for reasoning about the knowledge introduced by system designers as well as security experts through the graphical environment of the SysML-Sec framework.

## 1 Introduction

Security attack, whatever objective it has, is deontology that derives the wrongness of one's conduct which compromise the security objectives. Frequent reports about security vulnerabilities show that still many deficits exist in the development of secure software systems. The problem is even more pressing as the adversary activity and the destructiveness of attacks have increased over the last years. It is generally agreed that a central problem in the design of secure systems and the security analysis of existing systems is the danger of overlooking the system from particular standpoints [23]. This corresponds to situations in which security of the system is analyzed and described in terms of making the system secure by preventing weak links. In such a context, it is not sufficient to discover security attacks only at overlooked weak point of the system; there is also a need to analyze the information flow control issues, especially when the underlying platforms and infrastructures are also made of services themselves. Security analysts also need to consider threats to these underlying infrastructure and middleware for a particular security realization, as the assets to be protected originate both from the horizontal (i.e., between different entities and components) and vertical (i.e., multiple layers) compositions.

A related problem is that it is easier to analyze the protection level at each separate layer in the system architecture stack, but become vulnerable to various security exploits and flaws in a coordinated manner [8,18,22]. Because of their complexity and of the varying degrees in which system assets are deployed and executed, it is often the case that a system is compromised through a path its developers never have thought of. What is worse, a local security attack and vulnerability or a mismatch between the security mechanisms adopted at different locations can have dire consequences, potentially putting the security of large system at stake. Most of such security attacks stem from the limited knowledge shared between various security-engineering activities that collaborate with each other and the expression of their interdependencies.
One thing is that it is not easy to discover all parts of a system that are relevant for its security. In mainstream practice, this knowledge is often spread across different architecture layers, and correspond to various system development activities such as system architecture design, goal specification, In general, for a thorough security evaluation, one needs to take into account these different knowledge perspectives. In this context, in this paper, we aim at proposing a security analysis model derived from the conceptual constructs of security ontologies that will serve as the common knowledge repository for discovering, analyzing, and sharing attack knowledge with other system development activities. Thus, it will offer means to analyze the security of the system in such a way that it is possible to discover simple and complex security attacks and vulnerabilities at different levels of system abstraction. Furthermore, the concept of attack tree, modeled in SysML Attack Tree Diagrams, is brought in as the foundational graphical representation for modeling and embedding the collected security attacks knowledge into the security attack ontology. In this manner, the attack trees are completely parameterized by the ontological concepts so that it is possible to handle simultaneously several knowledge bases associated with security attacks and vulnerabilities. In particular, the knowledge based attack trees ease the process of keeping security attack specifications clear and understandable, minimizing the inconsistencies and helping to achieve maintainability – even when security attacks are drafted cooperatively by several entities as well as at different system development stages.

## 2 Collecting Knowledge about Adversary

In IT security engineering, to be on the safe side, we must assume that each attack scenario that is possible and promises whatsoever small benefit will definitely be carried out by someone. In this regard we first need to make a clear distinguish about: what does an adversary look like in distributed systems when different entities are involved such as when a client is a user, an owner, and a service provider and when some or all of the entities in the system can become adversaries?, Is there a hierarchy of adversaries when attacking such heterogeneous systems?, and so on. More specifically, security attacks are hard to understand, often expressed with unfriendly and limited details, making it difficult for secu-

rity experts and for security analysts to create intelligible security specifications. For instance, to explain "Why" (attack objective), "What" (i.e., system assets, goals, etc.), and "How" (attack method), adversary achieved his attack goals. We introduced security attack ontology by taking into account security standards and security dictionaries and deriving the features, in terms of classes and subclasses that were needed in such situations. Security attack ontology has been designed to enable the specification of security attacks in a concise, readable, and extensible way. Following, we detail different types of knowledge that an adversary require or use to perform an attack.

### 2.1 Adversary Profile

The adversary profile depicts the attack potential that is a measure of the minimum effort to be expended in an attack to be successful. In ISO/IEC 15408:2009 the attack potential is defined as a "measure of the effort to be expended in attacking a TOE, expressed in terms of an adversary's expertise, resources and motivation". Essentially, the attack potential for an attack corresponds to the effort required creating and carrying out the attack. The higher the adversary's motivation is the higher efforts they may be willing to exert. After having performed a comparative analysis of several security specifications and standards, we suggest the following abstract level taxonomy (see Figure 1) to be considered during an analysis of the attack potential:

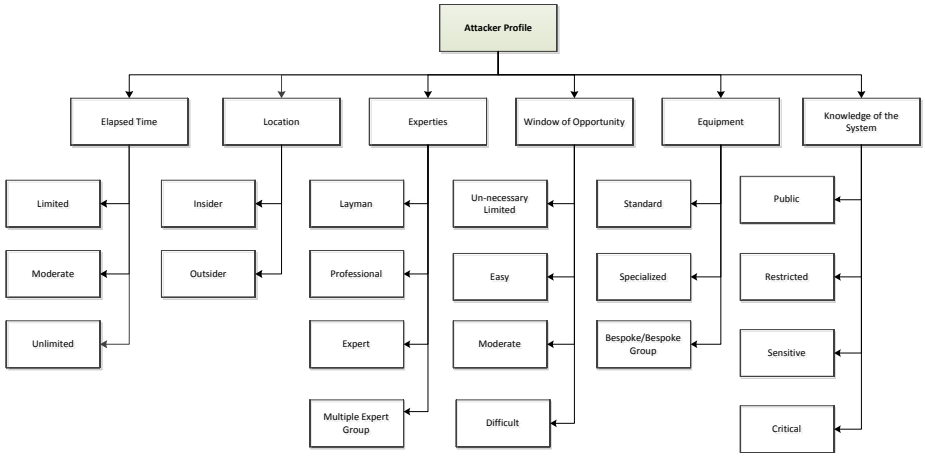


Fig. 1: Adversary Profile Taxonomy

– **Elasped Time:** This is the total amount of time taken by an adver- sary to identify that a particular potential vulnerability may exist, to develop an attack method and to sustain the effort required for mounting the attack.

- **Expertise:** This refers to the required level of general knowledge of the underlying principles for mounting an attack (i.e., system archi- tecture, security components, etc.), product types or attack methods.
- **Location:** This refers to the knowledge and the capabilities, which an attacker may have, depending of his/her location; this is typically reflected by the terminology for an Insider or Outsider attacker. For instance, insider attack agents are likely to have specific attack objectives, potential, and have legitimate knowledge and access to the system.
- **Window of opportunity:** This concept has a relationship with the elapsed time factor. Identification and exploitation of vulnerability may require considerable amounts of accesses to a system that may increase the likelihood of detection of the attack. In contract, some attack methods may require considerable effort off-line, and only brief access to the target to exploit.
- **IT hardware/software or other equipment:** This refers to the equipment required to identify and exploit vulnerability.
- **Knowledge of the system under investigation:** This refers to the specific expertise required in relation to the system under investigation. Though it is related to general expertise, it is distinct from it.

### 2.2 Adversary Objectives

Attack objective suggest particular types of adversary and his capabilities, as well as associated attack motivation. At the abstract level of specification attack motivations can be broadly categorized as:

- **Individual Benefits:** Personal advantages can be gained in different ways and for different purposes. For instance, gain reputation as hacker, financial gain fraudulent commercial transactions, etc.
- **Economical Benefits:** These motivations and underlying objectives should be envisaged at an organizational scale.
- **Political Benefits:** The main goal of the attacker is to destroy the reputation of an organization or an individual system asset. For example, acquiring system design information or for the purposes of fraud, industrial/state espionage or sabotage.
- **Criminal Benefits:** An augmentation of the attack motivation to harm an individual for the purposes of criminal or terrorist activity, destroy or financial harm, destructive attacks or intellectual property attacks, etc.

### 2.3 Attack Mode used by an Adversary

The attack mode refers to the actions that an adversary takes during the execution of an attack and that can be labeled as active or passive attacks:

- **Active Attacks:** modifying the behavior of the system.
- **Passive Attacks:** aiming at information retrieval without modifying the behavior.

### 2.4 Attack Method used by an Adversary

The attack methods are related to the attack mode class. The attack method can be classified into either functional (logical) attacks or physical attack methods:

- **Physical Attacks:** Attacks physically modifying the behavior of the system.
- **Functional Attacks:** From the functional point of view, attacks aiming at logical manipulation of information without physically modifying the system behavior.

### 2.5 Attack Consequence

Attack Consequences refers to an impact of security breach or outcomes that are not the ones intended by a purposeful system action. The attack consequences can be classified as:

- **Usurpation:** is a derogatory term used to describe either a misappropriation or misuse of the system functionalities.
- **Disruption:** is an event that causes an incapacitation, corruption, obstruction, and unplanned deviation from the expected system behavior, according to the functional and non-functional objectives.
- **Deception:** is defined as masquerade, falsification, and repudiation actions taken by an adversary, to thereby causes a system to accept as true a specific incorrect version of reality.
- **Disclosure:** enables an adversary to gain valuable information about a system and its functionalities either by exposure, interception, inference, intrusion, etc. that tries to uncover the details of a system.

## 3 Security Ontology

In this section, we define the security ontology – Security Attack Ontology – for modeling different types of adversary's knowledge. Security ontology constitutes a knowledge repository for capturing, classifying, and sharing security related information. More specifically, the definition of a security attack ontology aims at building knowledge vocabulary for security attacks that could be described including their type, mode, consequences, and such details as described above. Figure 2 sums up our analysis with respect to extracting different constructs and concepts defined in well-known security standards (i.e., ISO/IEC 15408:2009, ISO/IEC 18045, ISO/IEC 27000: 2012, ISO/IEC 17799:2005, NIST SP-800:30, etc.) and security dictionaries (i.e., CVE, CAPEC, OWASP, CLASP, etc.) in order to build the security attack ontology. This has been modeled with the Ontology Web Language [2] using OWL classes. Our security ontology use a flexible and easily extendable structure, which makes it possible to seamlessly add new concepts.

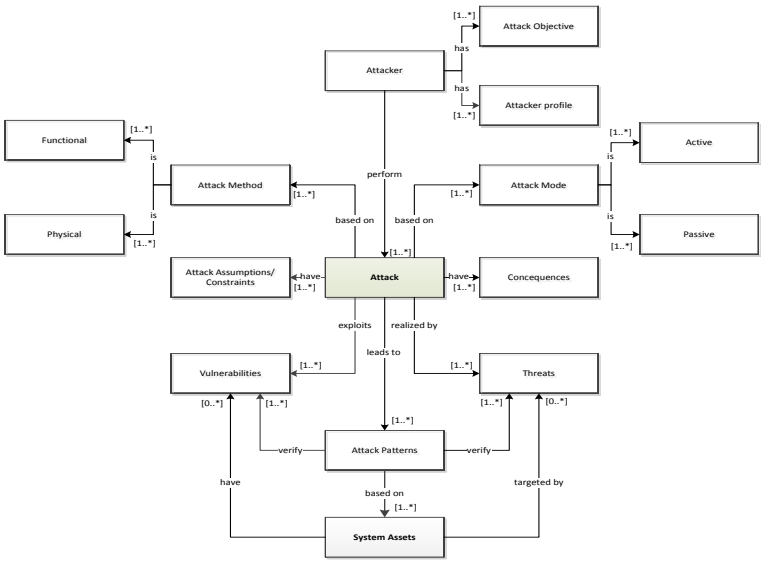


Fig. 2: Security Attack Ontology

## 4 Attack Modeling

The concept of security analysis is similar to the concept of trade-off analysis in that there is also more than one way to attack system assets, and an adversary may be trying them simultaneously or just a subset of them. More precisely, an adversary can use distinct attack paths or alternative approaches until reaching his attack objectives. This is often illustrated through the attack trees, which form a convenient way to systematically categorize the different. Basically; attack trees (the term was introduced by Schneier [19]) are multi-levelled diagrams consisting of one-root, leaves, and children nodes. In addition, different node values can be combined with AND, OR relationship to learn even more about a system's security flaws and weaknesses. Specifically, the purpose of an attack tree is to define and analyze possible attacks on a system in a structured way. This structure is expressed in the node hierarchy as well as in the form of logical operators (i.e., conjunctive (aggregation) or disjunctive (choice), etc.) for expressing interrelationship between different attack tree nodes. Thus, using both logical operators and node definition retains the natural way security experts build the attack trees or fault trees [22,18,20,1,27]. Actually, these two building blocks (nodes and logical operators) of an attack tree can be modeled with the definition of constraint block with a object functions and the part element of the parametric diagram. Thus, at a conceptual level we can use parametric diagrams to model attack trees. Let us present how we suggest representing attack trees in SysML using the above-mentioned modeling constructs.

### 4.1 Parametric Diagram

The parametric diagram is the second new type of diagram introduced to describe constraints on system properties to support engineering analysis. The parametric diagram is a specialized variant of an internal block diagram that restricts diagram elements to represent constraint blocks, their parameters and the properties of block that they bind to. Parametric diagrams are made up of one or more constraint blocks, zero or more part, and one or more connectors [6]. The constraint block is used to show which constraints are being used. The SysML specification describes constraint blocks in terms of conditions that are represented by mathematical equations. More precisely, the constraints block contains an equation, expression or rule that relates together the parameters given in the parameters block. The concepts behind constraints can be extended to cover general rules that constrain system properties and behavior such as authentication should be performed BEFORE authorizing entity to access system resources, etc. The use of a constraint block is called a constraint property and is depicted on a parametric diagram. The interconnection between constraint blocks and part or constraint blocks is shown on a parametric diagram using zero or more binding connectors. Binding connectors depict an equality relationship between the two connected parameters or between a parameter and a value property. In the parametric diagrams, a standard part element includes properties to specify its unique identifier and text description.

### 4.2 Attack Trees in Parametric Diagram

Let us first focus on the extension of the parametric metamodel (see Figure 3) that is necessary for modeling attack trees. Following the extension mechanism suggested in the SysML specification where the stereotype mechanism is defined to extend the existing SysML classes, we create a new stereotype to represent security attacks: the ”attack tree” This is illustrated in Figure 3. As mentioned earlier, we mainly focus in this paper on expressing security knowledge to be shared and reused throughout the system development process to design a secure system. In order to integrate attack related knowledge, we extend the parametric diagram's ”part” element with ontological concepts and properties from the attack tree ontology, presented in Section 3. We argue that such a representation is indispensable to precisely understand how attack trees can be manipulated during their construction and analysis. More details are given in Section 5 about the introduction of security reasoning into SysML models. We use the ”constraint block” element for the definition of set of constraints such as mathematical expressions (i.e., AND, OR, etc.) among the pieces of the security attack nodes. The objective of these operators is to show the relationship among difference attack nodes. More precisely, we use OR operator to represent alternatives ways an adversary tries to achieve his attack objectives. For instance, an adversary has to perform either one of the attacks ”hijack authenticated session” OR ”disconnect client” to accomplish his attack goal. AND relationship represent different steps toward achieving the same goal, for example, by assuming an adversary can gain

root access of vehicle Communication Unit (CU) if and only if he can tamper the on-board communication unit. In our attack tree modeling approach, rather than considering only these two types of logical operators, we also consider temporal operators (i.e., AFTER, BEFORE, SEQUENCE, etc.). We in particular allow security experts to capture temporal dependencies between attack nodes and sequences in an attack. For instance, in order to install the bogus authority keys an adversary first have to switch an ECU into a re-programming mode. Furthermore, we can represent the ordering between attacks by using the SEQUENCE relationship. We use a "connector" element to link zero or more "part" with constraint block. The use of a "constraint block", "part", and "connector" element for building attack trees is shown in Figure 4.

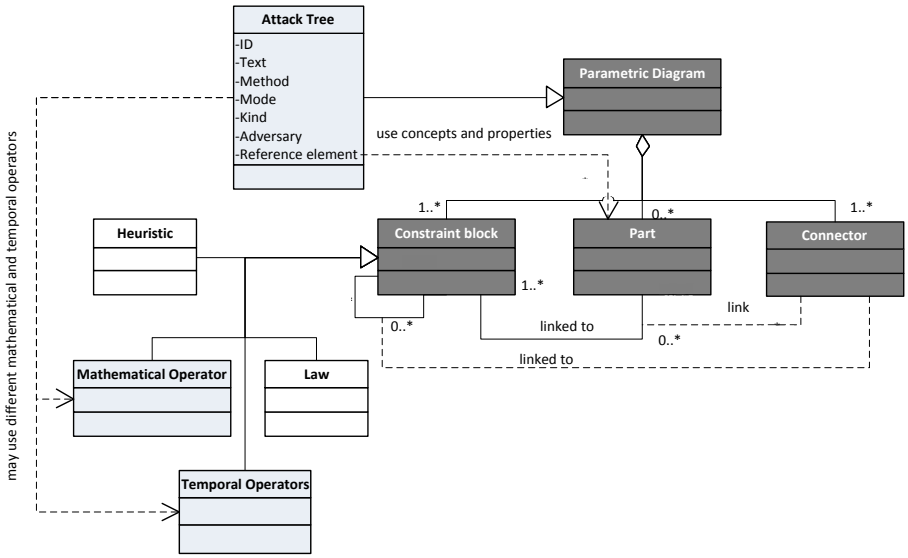


Fig. 3: Metamodel for the SysML attack tree diagram

### 4.3 Knowledge-Centric Attack Tree Modeling

An overall procedure for attack tree modeling looks like this:

1. Build attack tree rooted (Level 0) on an abstract "attack objective". We use the "part" element to model each attack tree node.
2. Its child nodes (Level 1) represent different "attack goals" that could satisfy this attack objective. Attack goals and attack objectives are linked via a binding "connector".
3. For each attack goal node:

- Decompose into a number of "attack methods" (Level 3) that could be employed to achieve the attack objective.
- Specify the logical relationships (Level 2) between different attack methods, if there are. We use the "constraint block" to specify these logical expressions. At this stage, we also consider intermediate steps that represent attack method at a certain level of abstraction.

4. The attack tree terminates when leaf conditions (basic operations are described that gives all details of the attack) are reached that meet the adversary's capabilities.

The attack tree modeling approach that we advocate provides a bridge between the typical attack trees modeling approaches [20], and the anti-goal models approaches [25]. More precisely, the first two steps of our attack tree modeling approach are equivalent to the KAOS anti-goal model [25], which provides the top down approach for modeling attacks. The next two steps correspond to the standard attack tree modeling approach, where attacks are identified from bottom up perspective. Figure 4 sums up these two approaches.

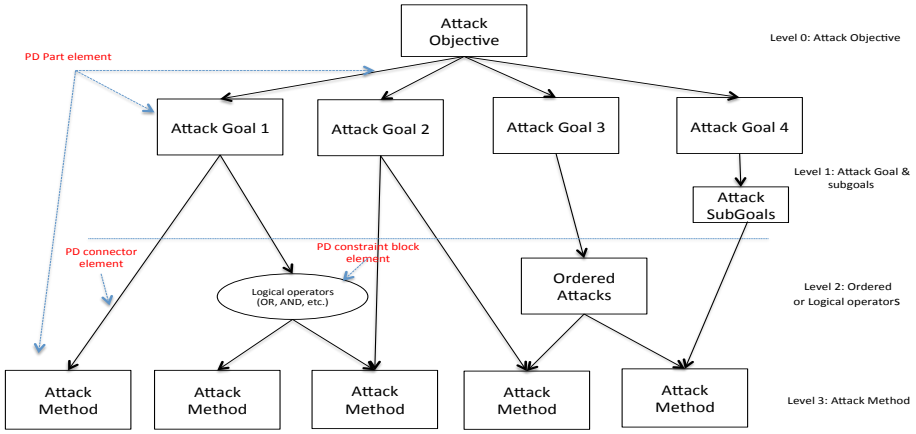


Fig. 4: Generic knowledge-centric attack tree structure

### 4.4 Tooling

The aforementioned approach has been applied with TTool [12]. TTool is an open-source toolkit that supports several UML/SysML profiles, including TURTLE [3] , DIPLODOCUS [4], and AVATAR [27,18]. TTool includes diagramming facilities, formal and non-formal code generators, model simulators, model-checkers and analysis tools. DIPLODOCUS is focused on the design space exploration of complex embedded systems. TURTLE is now deprecated and replaced with AVATAR. the latter targets the design of embedded applications. The main

strength of TTool is to hide knowledge of the underlying simulation or formal proofs techniques, thus offering a press-button approach to perform safety or security proofs. We have used the SysML-Sec profile as a part of the AVATAR profile to build attack trees.

## 5 Integrating Knowledge Bases and Reasoning into SysML

Syntactically, SysML and ontology languages (i.e., OWL, OIL, etc.) have a lot of similarities. While SysML makes use of a graphical formalism, it also aims at defining the semantics of a system with constructs like blocks, associations, part properties, and relationships between models and sets of model elements. Ontology languages use classes, properties, relationships, and individuals as basic knowledge constructs. For instance, OWL defines classes by appropriate and implicit logical constraints on properties of their subclasses and concepts. The integration of both approaches enables engineers to add reasoning arguments to the explicit documentation of system models, and to define more precise relationships in the course of a typical model-based development process. We discuss in the following how ontologies and inferences on ontologies are used to enrich the SysML-Sec framework.

### 5.1 Annotating SysML Diagrams with Ontological concepts

In this section, we focus on annotating security concepts and terms defined in the security attack ontology (cf. Section 2) with attack tree diagrams. Figure 5, presents an overview of an integration approach for embedding ontological concepts and terms into SysMLsec models. As previously stated in section 4 we can add the ontological concepts and terms into SysML models by extending the SysML metamodel by including the user defined stereotypes or properties and tagged values. Let us first consider the Attack Tree diagram (presented in section 4.3), and map adversary related ontological classes (cf. section 3) to the Attack Tree diagram. We build the integration approach based on three core ideas:

- We have defined the "Attack" stereotypes (see Figure 6) to represent the security attack ontology in the SysML Attack Tree diagram.
- We integrate high-level ontology classes (see Level-1 in Figure 6.b) as a SysML Attack Tree diagram properties (i.e., type, kind, etc.) as shown in Figure 6.a.
- We use ontology subclasses (see Level-2/n Figure 6.b), as tag values of the SysML attack tree diagram property element. This is illustrated in Figure 6.a. These values constitute a controlled vocabulary. Thus, it provides a canonical set of mapping mechanism in order to deal with integration of ontological concepts into the SysML.

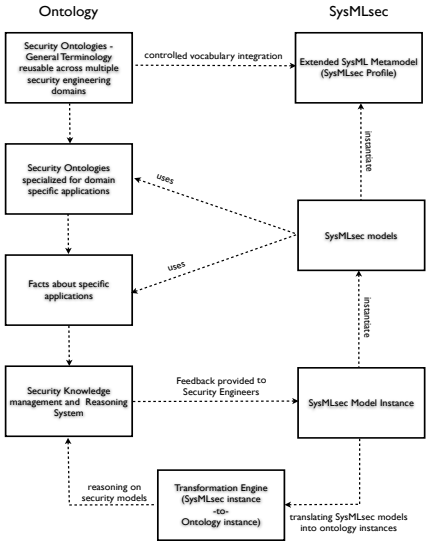


Fig. 5: Integration of ontology reasoning on security with SysML

According to these rules, every Attack Tree diagram extended with an "Attack" stereotype is also associated with ontology concepts and terms as shown in Figure 6.a. The diagram consists of two parts; standard SysML parametric properties (e.g., id, text) and extended ontological properties (e.g., kind, type, classification, etc.). The discussion here will be limited to the extended attack's property constructions that can be directly translated to ontology classes. It can be seen that, for each high-level class of the security attack ontology (see Figure 6.b.) we basically define a new property element in the attack diagram. This is illustrated in the Figure 6.a. These properties are then populated with the subclasses and concepts defined in the security attack ontology as its tagged values. In particular, the properties and tagged values are specified in the same manner as the classes and subclasses concept described in the security attack ontology. For example the "Adversary" class takes Layman, Expert, and Professional as its tagged value. Accordingly, we map all other concepts and terms defined in the security attack ontology into the "part" element.

### 5.2 Reasoning with SysMLsec Models

In this section, we describe the extent to which we can use the capabilities of ontologies to reason about these different security concepts defined in Attack Tree diagram. In particular, our objective is to enable the security engineers to have access to various ontological concepts and terms, and to reason on these models. Although, with integration of ontology classes and subclasses into the SysML Attack Tree diagram, we already provided the partial reasoning capabilities to reason about different security concept within the SysML models. More pre-

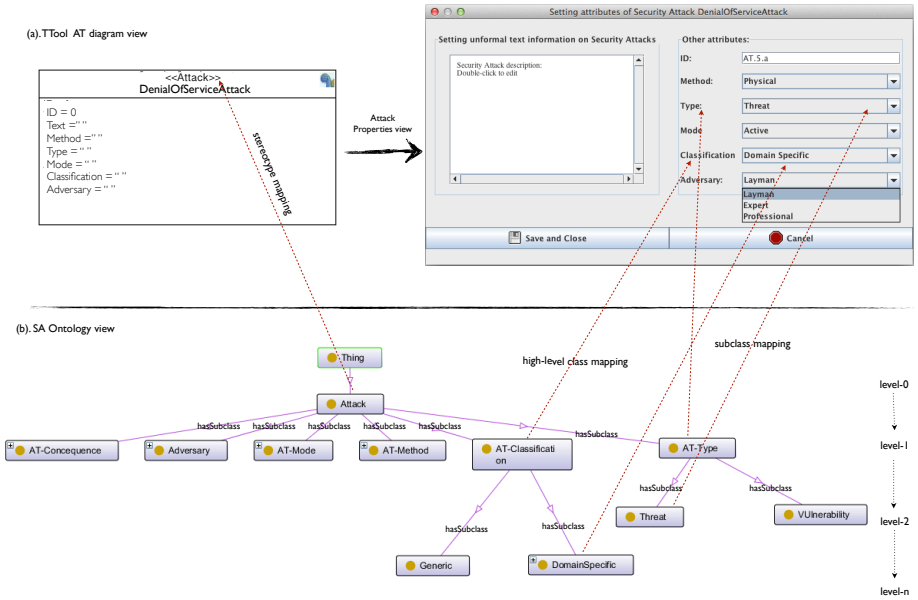


Fig. 6: Mapping of the security attack ontology concepts into the SysML Attack Tree diagram

cisely, when security engineer select a particular concept in the SysML diagram, for instance, Attack is a "Domain specific" threat, we annotate the structure of the sub-classification with the tagged values that belong to the domain specific class such as an application specific, middleware specific, etc, as shown in the property view of Figure 6. In a similar way, for each ontology class we apply the same approach and limit the knowledge space for security engineers to specify only those concepts and terms that belong to the super class or the parent class. Thus, provide means to reason about security concepts within the SysML models, which brings additional power to the development of security models like consistency checking (i), concept satisfiability (ii), and concept classification. The shortcoming is that we cannot specify the reasoner calls in relation to one another or in relation with other security models such as security goals, attack, system architecture, etc., which is our core objective. In order to fulfil this design objective, we have implemented the "SysMLsec-to-Ontology" translation engine as shown in the Figure 7. The translation engine, we have implemented for mapping from SysMLsec models to the OWL description, contains a set of rules that match security constructs and transform them into equivalent instance of ontology.

The primary purpose of this translation engine is to make the security engineers able to reason about their security models using well known and efficient reasoning engines such as SPARQL [16], OWL-QL [4], RACER [5], SQWRL [15], etc. Engineers can directly make use of reasoning capabilities of these engines

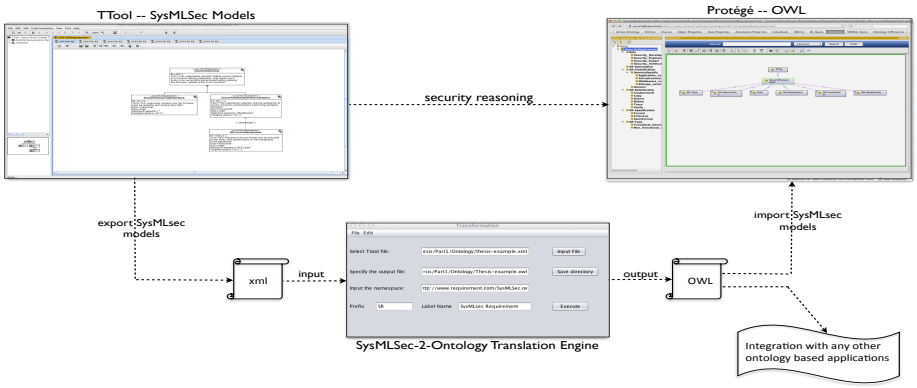


Fig. 7: Integration of ontology reasoning on security with SysML

within the context of current engineering practice and tools without building and using some separate ontological models. In particular, our objective is to give the security engineering a more precise way to employ reasoning in the course of a typical model-based development process. On the other hand, since we obtain an OWL described document, we can integrate our SysMLsec models with any other ontology based applications such as integrating the security requirement knowledge with security resource annotating approaches like [11], or providing input to the ontology based risk analysis approaches [3] in order to compute risk metrics.

In this paper, we refer to SQWRL (Semantic Query Web Language) [15] as a query language because of its concise, readable, and semantically robust semantic. SQWRL is a SWRL-based [7] query language that can be used to query OWL ontologies and provided in Protégé 4.2 beta version [24]. To retrieve knowledge from OWL ontology, SQWRL provide SQL-like operation. The form of rule is

$$antecedent \rightarrow consequent$$

In this rule an antecedent part is referred to as the body, and a consequent part is referred to as the head. Both the body and head consist of positive conjunctions of atoms:

$$Atom \ \wedge \ Atom \ \rightarrow Atom \wedge \ Atom$$

This rule can be read as if all the atoms in the antecedent are true then the consequent must also be true. Here, an atom is an expression of the form P($\arg_1$, $\arg_2$, ...$\arg_n$), where p is a predicate symbol and $\arg_1$, $\arg_2$, ..., $\arg_n$ are the terms or arguments of the expression. In our approach, the predicate symbols can include security ontology classes (i.e., asset, goal, attack, security requirement, etc.), properties (i.e., hasFunction, hasSequence, hasAvoidGoal, etc.) or data types. Arguments can be class individuals (i.e., type, classification, adversary,

etc.) or data values, or variables referring to them. In the further course of this thesis, we will use the above-mentioned SQWRL query expression to retrieve, manipulate, and reason about different security-related information.

## 6 Related Work

In the following we will introduce different threat modelling profiles. These modelling profiles capture a certain types of information and results in different types of threats and vulnerability models and security design solutions. A number of extensions of UML (i.e., UMLsec [10], Anti-Goal [25], Misuse cases [21], Abuse cases [13], etc.), allow to express security relevant information within the diagrams in a system specification have been proposed. For instance, Abuse Frames are based on the Jackson's problem frames approach [9] and is intended to analyze security problems in order to determine security vulnerabilities and to derive security requirements. This approach introduces the notion of anti-requirement (similar to the concept of an anti-goal [25]) to describe the behavior of a malicious user that can subvert an existing requirement. The basic idea behind the definition of abuse frames is to bind the scope of a security problem with anti-requirements in order to derive security requirements. Such explicit and precise descriptions facilitate the identification and analysis of threats, which in turn drive the elicitation and elaboration of security requirements. Possible ways of misusing system functionality can be specified by an extension of UML. Misuse case diagrams not only shows regular actor/use case relations but also can model threats that threaten use cases, and countermeasures that mitigate these threats. Misuse cases extend the traditional use case approach to also consider misuse cases, which represent behavior not wanted in the system to be developed. A misuse case diagram contains both, use cases and actors, as well as misuse cases and misusers. Development of misuse cases allows the identification of security attacks and associate security requirements during application development. In [28], misuse cases are further analyzed and author's presents a formal representation of misuse cases and provide an intuitive way to executable misuse case model. Although misuses cases are not entirely problem-oriented as they represent aspect of both problems and solutions, they have become popular as a means of representing security concerns in the early stages of software development. Yet, to best of our knowledge, none of them provides the expressivity required to deal effectively with system-wide security attacks. Another major group of contributions to the conceptual modeling of security attacks like KAOS [26] and Secure Tropos [14], etc., have defined their own graphical formalism each of which allows to express security relevant information (i.e., goal, anti-goals, requirements, obstacles, etc.). However, security issues involve special concerns that these traditional software engineering languages do not consider. Consider, for example, a general behavior modeling notation that expresses interactions of entities in the system without considering the harmful behavior of an adversary. Thus, the models do not convey the impacts of the malicious

behavior of the adversary on requirements, design, and architecture to the next phases of system development lifecycle.

## 7 Conclusion

Given an input for our knowledge centric design methodology, the security analysis process helps to both classify identified attacks, but also to think about new ones, given a category. Knowledge centric attack tree is a combination of both top-down and bottom-up approach to provide a support tool to security analysts. The purpose of developing the ontology driven attack trees is to identify possible security threats and to allow aspects such as the desirability (to the adversary), opportunity, probability and severity of attacks to be assessed in order to share knowledge among various system development activities (i.e., security requirements engineering, protocol design, testing, etc.). We believe that, on the one hand, ontology based security analysis is expressive enough to describe several real-world security attacks with a multi faceted approach; at the same time, it provides constructs to map and relate security attacks with other system development activities.

## References

1. S. A. Camtepe and B. Yener. Modeling and Detection of Complex Attacks. *In 3rd International Conference on Security and Privacy in Communications Networks, SecureComm*, September 2007.
2. M. Dean, G. S. (eds.), F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL Web Ontology Language Reference. 2003.
3. A. Ekelhart, S. Fenz, M. Klemen, and E. Weippl. Security Ontologies: Improving Quantitative Risk Analysis. In *40th Annual Hawaii International Conference on System Sciences*, HICSS'07, page 156a, Jan 2007.
4. R. Fikes, P. Hayes, and I. Horrocks. OWL-QL- A language for deductive query answering on the Semantic Web. *Journal Web Semantics: Science, Services and Agents on the World Wide Web*, pages 19–29, December 2004.
5. V. Haarslev and R. Moller. RACER: An OWL Reasoning Agent for the Semantic Web. In *1st International Workkshop on Applications, Products and Services of Web-based Support Systems*, WCC'03, pages 91–95, 2003.
6. J. Holt and S. Perry. *SysML for System Engineering (Professional Applications of Computing)*, volume 7. IET, 2007.
7. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Available at `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`, May 2004.
8. M. S. Idrees, Y. Roudier, L. Apvrille, and G. Pedroza. Test Results. Technical Report D4.4.2, EVITA Project, 2011.
9. M. Jackson. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Professional, 2001.
10. J. Jüerjens. *Secure Systems Development with UML*. Springer, 2003.

11. A. Kim, J. Luo, and M. Kang. Security Ontology for Annotating Resources. In *Proc. of the 2005 OTM Confederated international conference on the Move to Meaningful Internet Systems: CoopIS, COA, and ODBASE*, OTM'05, pages 1483–1499, 2005.
12. T. P. LabSoc. The TURTLE Toolkit - TTool. Available at `http://labsoc.comelec.enst.fr/turtle/ttool.html`.
13. L. Lin, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett. Introducing Abuse Frames for Analysing Security Requirements. In *Proc. of the 11th IEEE International Conference on Requirements Engineering*, RE'03, pages 371–372, September 2003.
14. H. Mouratidis, P. Giorgini, G. Manson, and I. Philp. A Natural Extension of Tropos Methodology for Modelling Security. In *Proc. of the Agent Oriented Methodologies Workshop*, OOPSLA'02, 2002.
15. M. J. O'Connor and A. K. Das. SQWRL: A Query Language for OWL. In *Proc. of the 5th International Workshop on OWL: Experiences and Directions*, OWLED'09, October 2009.
16. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Available at `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`, January 2008.
17. Y. Roudier, M. S. Idrees, and L. Apvrille. Towards the model-driven engineering of security requirements for embedded systems. In *MODRE 2013, International Workshop on Model-Driven Requirements Engineering, 15 July 2013, Rio de Janeiro, Brazil*, Rio de Janeiro, BRAZIL, 07 2013.
18. A. Ruddle, D. Ward, B. Weyl, S. Idrees, Y. Roudier, M. Friedewald, T. Leimbach, A. Fuchs, S. Gürgens, O. Henniger, R. Rieke, M. Ritscher, H. Broberg, L. Apvrille, R. Pacalet, and G. Pedroza. Security requirements for automotive on-board networks based on dark-side scenarios. Technical Report Deliverable D2.3, EVITA Project, 2009.
19. B. Schneier. Attack trees. *Dr. Dobb's Journal: Software Tools for the Professional Programmer, Ebsco*, (24):21–27, 1999.
20. B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.
21. G. Sindre and A. L. Opdahl. Eliciting Security Requirements by Misuse Cases. In *Proc. of the 37th International Conference on Technology of Object-Oriented Languages and Systems*, TOOLS-Pacific'00, pages 120–131, 2000.
22. J. Stefan and M. Schumacher. Collaborative Attack Modeling. In *Proc. of the ACM Symposium on Applied Computing*, SAC'02, pages 253–259, March 2002.
23. J. Steffan and M. Schumacher. Collaborative attack modeling. In *Proceedings of the 2002 ACM symposium on Applied computing*, SAC '02, pages 253–259, New York, NY, USA, 2002. ACM.
24. S. University. Protégé Ontology Editor and Knowledge-base Framework. Available at `http://protege.stanford.edu/`.
25. A. van Lamsweerde. Engineering Requirements for System Reliability and Security. In *Software System Reliability and Security*, NATO Security through Science Series - Information and Communicarion Security, pages 196–238, 2007.
26. A. van Lamsweerde and E. Letier. From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. In *Radical Innovations of Software and System Engineering*, pages 4–8, 2003.
27. G. Vigna, S. Eckmann, and R. Kemmerer. Attack Languages. In *Proc. of the IEEE Information Survivability Workshop*, ISW'00, pages 163–166, October 2000.
28. J. Whittle, D. Wijesekera, and M. Hartong. Executable misuse cases for modeling security concerns. In *Proc. of the 30th international conference on Software engineering*, ICSE'08, pages 121–130, 2008.