Towards a Model-driven based Security Framework

Rouwaida Abdallah, Nataliya Yakymets, Agnes Lanusse

Towards a Model-driven based Security Framework

Rouwaida Abdallah, Nataliya Yakymets and Agnes Lanusse
CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, Gif-sur-Yvette Cedex, France
{rouwaida.abdallah, nataliya.yakymets, agnes.lanusse}@cea.fr

Keywords: Security, Model-driven, UML Profiles, EBIOS, Attack Trees, Papyrus Tool.

Abstract: In this paper, we propose a model-driven framework for security analysis. We present a security analysis process that begins from the design phase of the system architecture then allows performing several security analysis methods. Our approach presents mainly two advantages: First, it allows the traceability of the security analysis methods with the system architecture. Second, this framework can include several security analysis methods. Moreover it allows information reuse which is complicated when we use separate methods dedicated tools. Thus, we can have more consistent and accurate security analysis results for a system. We chose to implement two methods: A qualitative method named EBIOS which is simple and helps to identify areas of focus within the system. Then, to get more accurate results, we implement a quantitative method, the Attack trees. Attack trees can be automatically generated from the Ebios analysis phase and can be completed later on to get more specific results.

1 INTRODUCTION

Today Model Driven Engineering (MDE) has proven its efficiency to cope with the ever-growing system complexity (Bernardi, 2013), (Bran and Gérard, 2014). In particular, there has been substantial research on model-based security analysis (Basin, 2011). The use of models in security engineering offers more focused views of complex systems, and several levels of abstraction to assist non-security experts to implement security efficiently.

Several risk management methods have been established to improve the security of information systems. These methods identify areas of focus within the project that need special attention and security and privacy measures. These methods can be broken down mainly into two essential types: qualitative (e.g. NIST SP 800-30 (Stoneburner, 2002), CORAS (den Braber, 2007), OCTAVE (Alberts, 2003), EBIOS (Secrétariat Général de la Défense Nationale,2004), etc.) and quantitative (e.g. CORA (International Security Technology, 2002), ISRAM (Karabacaka and Songukpinar, 2005), AttackTree (Schneier, 1999), etc.). Some approaches can be applied to all types of risks, while others are specific to particular risks like for instance risks related to information security. Qualitative methods implement no mathematical computations in general and thus they are considered as simpler but less precise than quantitative methods. Then, if an organization is concerned with simplicity rather than accuracy, qualitative methods are good fit, otherwise the choice will be quantitative methods (Behnia, 2102).

In this paper, we present the first steps towards a model-driven process for risk analysis in order to get accurate results but in a simpler way. This process is twofold: First, we proceed by a qualitative method to assess the risks that we consider as dangerous or unacceptable relatively to the threshold we have preliminary fixed for the system. This step will reduce the perimeter of the risk analysis. Then we apply a quantitative method considering only the reduced perimeter to get more precise and accurate results for all the system.

Our approach presents many advantages: First, as it is a model driven approach we can benefit from the architecture of the system realized at the design phase to apply the risk analysis methods. Second, having the two methods in the same environment allows us to reuse information which is complicated when we use special methods dedicated tools. Most existing risk analysis methods rely on separate tools or models etc. Using separate tools requires a lot of experience and extra effort and may lead to inconsistencies between the different analyses. Similar challenges have been solved in software

engineering by model-driven approaches. The core idea is to reuse as much information as possible from earlier design stages in later stages of the development cycle. This idea is already very successfully being used for the development of software intensive systems (Gudemann and Ortmeier, 2011)

In this paper we present the first steps towards a model-driven process for security analysis. It is structured as follows: Section 2 provides the background of this work. Section 3 describes the process and the model driven approach. Section 4 presents related works. Finally, the conclusions are in section 5.

2 BACKGROUND

In this section, we present first the two security analysis methods: EBIOS and Attack Tree. Then we give the motivation of this work.

2.1 EBIOS Method

One of the well-known qualitative methods dedicated to manage risks in information systems operating in steady environments is called EBIOS (Expression of Needs and Identification of Security Objectives). EBIOS is used by many organizations in both public and private sectors to conduct Information System Security (ISS) risk analyses. EBIOS method (Secrétariat Général de la Défense Nationale, 2010) provides uniform vocabulary and concepts that allows attending security objectives. It can be adapted to the context of each organization (its tools and methodologies) and then used to develop either a complete global study of the information system or a detailed study of a particular system. Some efforts have been made to automate the EBIOS method. It helps the user to perform risk analysis and management steps according to the five EBIOS phases and to automatically generate reports:

- Phase 1 deals with context analysis. It establishes the environment, purpose and operation of the target system and identifies the essential elements (assets) on which they are based.
- Phase 2 conducts the security needs analysis. The identified security needs of the essential elements are evaluated in terms of availability, integrity and confidentiality, etc. In other terms we identify the Feared Events of the system. We also define a severity level to this Feared Event based on the harm that it may induce.
- Phase 3 consists of identifying and describing the threats affecting the system. This is done by studying the attack methods and threat agents using them by exploiting existing vulnerabilities of the elements of the system. We associate to each threat the likelihood of occurrence.
- Phase 4 contributes to risk evaluation and treatment. It formalizes the real risks affecting the system by comparing the threats with the security needs.
- Phase 5 determines how to specify security countermeasures allowing the security objective to be fulfilled and how to validate these measures and the residual risk. Actually, for each Feared Event we associate a risk level. This level is computed based on the severity "*sev*" (possible values for *sev* : *Negligible*, *Limited*, *Important*, etc.) of the Feared event concerned by the risk and the maximal value "*lik*" (possible values for *lik* : *Minimal*, *Significant*, *Heavy*, etc.) between the likelihoods of the threats that lead to this Feared Event (possible values for the risk level: *Negligible*, *Significant*, *Intolerable*, etc.). The risk level is deduced from a predefined matrix *RiskLevel(sev,lik)*. A residual risk is the risk after applying existing or new countermeasures what may decrease the severity and/or the likelihoods and so it may decrease the risk level.

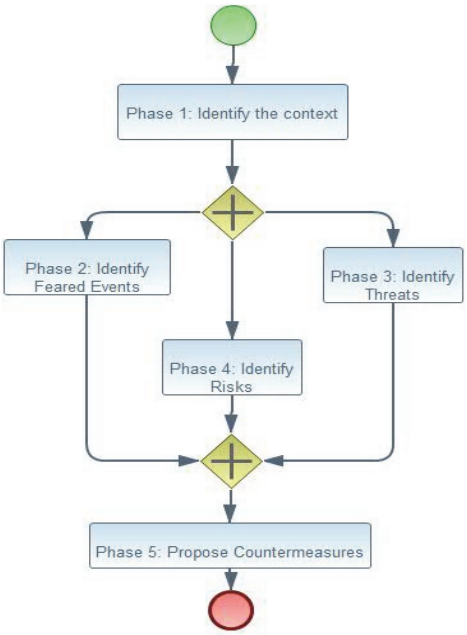

Figure 1: EBIOS analysis method.

Some efforts have been made to automate the EBIOS method. It helps the user to perform risk

analysis and management steps according the five EBIOS phases and to automatically generate reports. However, such tools are disconnected from the design model and do not provide an overview of the system or its layers. Besides, they do not offer the possibility to perform any further security analysis methods which can be important to get a more complete security analysis of the system.

Although EBIOS is dedicated to ISS, it can be adapted to different contexts such as (Mcdonald et al., 2011). Besides, EBIOS meets the risk management described in ISO 27001 and supports entirely ISO 27005 and ISO 31000 (Secrétariat Général de la Défense Nationale, 2010).

2.2 Attack Tree Quantitative Method

Attack trees (the term is introduced by Schneier in (Schneier, 1999)) can be used to model potential attacks on a system and corresponding risks associated with each attack path.

Attack trees describe attacks towards any system as a logical function of atomic attacks. The top node of an Attack tree is the ultimate goal with combinations of sub-goals. Children of a node are refinements of this goal, and leaf nodes therefore represent attacks that can no longer be refined. An Attack leaf can be an element of different intrusion scenarios, depending on the node connectivity associated with it. We have two types of node connectivity: "OR" nodes represent different ways to achieving the same goal (in Figure 2 the top node is an OR node). "And" nodes represent different steps in achieving a goal (in Figure 2 the node labeled by "Eavesdrop" is an AND node).

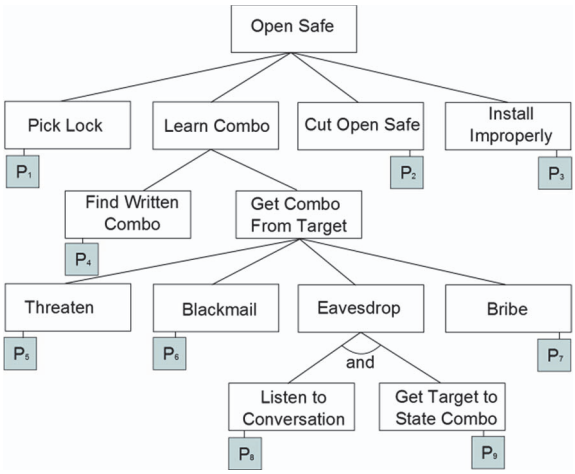

Figure 2: Example of an Attack Tree (Schneier, 1999).

Figure 3 presents the procedure that we follow in general in an Attack tree based analysis inspired from (Ten et al., 2008). Once a tree is created, we can compute all the scenarios that lead to its top node. Moreover, Attack trees allow several parameters values to be associated to leaf nodes (cost, time to achieve, likelihood of occurrence, etc., or qualitative statements such as "possible" "impossible", etc.). These parameters are used to compute the value of the vulnerability index (or index) associated to an Attack tree.

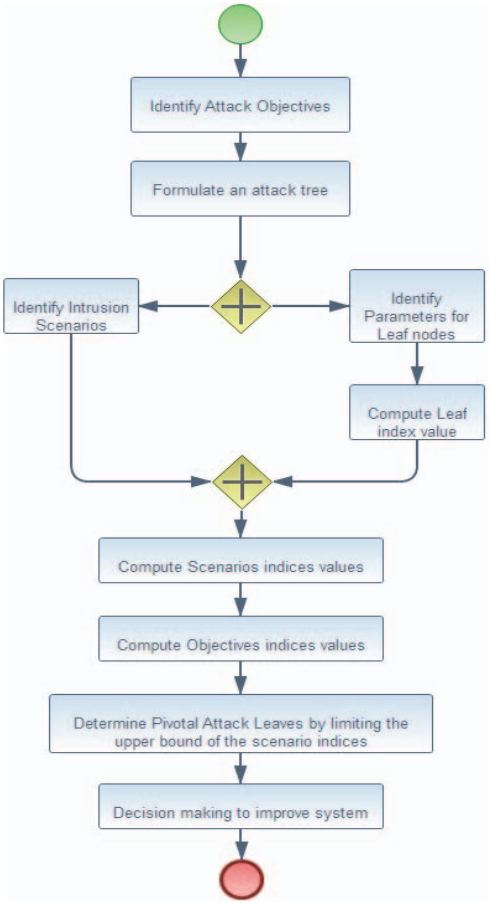

Figure 3: General Attack tree analysis process.

We distinguish between three types of indices: First, the index v (l_i) associated to a leaf node l_i which is computed based on a formula F that we choose and the set of parameters values P_i of this as shown in (1). Second, the index $V(s_m)$ of an Attack scenario s_m concerning a subset $L_m = \{l_1,l_2,..,l_p\}$ of p leaf nodes is computed as shown in (2). Finally, the index of a top node V_t is determined from the scenarios as shown in (3) where $S= \{s_1, s_2,...,s_k\}$ is the set of the scenarios and k is the total number of scenarios.

$$v(l_i) = F(P_i) \quad (1)$$

$$V(s_m) = \prod_{j \le p, lj \in Lm} v(lj) \quad (2)$$

$$V_t = \max\{ V(s_1), V(s_2), \ldots, V(s_k) \} \quad (3)$$

2.3 Motivation

In many engineering disciplines, model building is at the heart of any system design. But model building is not an end in itself and certainly does not come for free. There is an important added value so that this effort becomes worthwhile. In (Basin, 2011), the authors summarize in four activities what models can be used for in the development of secure systems: (1) Precisely documenting security requirements together with design requirements; (2) Analyzing security requirements; (3) Model-based transformation, such as migrating security policies on application data to policies for other system layers or artifacts; (4) Generating code, including complete, configured security infrastructures. These four activities can be used to automate EBIOS and Attack trees in the following way.

(1) Models, especially graphical ones, give a clear overview of the system or a part of it. The core idea is to reuse as much information as possible from earlier design stages in later stages of the system development cycle. Existing EBIOS tools do not present any graphical representation of the system, the security analysis performed with EBIOS is not based on any other document or support used in the system design phase. Moreover, the contextual information about the system and its environment are entered from scratch. For Attack trees we find several graphical tools (e.g. SecureITree (Saini *et al*., 2008)) however they are not related to the design phase either.

(2) EBIOS presents a very robust process for analyzing security requirements. However, we believe that a global overview and several viewpoints of the system, or subsystems can offer to the security engineer a better recognition of the hazards. Attack trees method is not as robust as the EBIOS method at this point.

(3) To perform a robust security analysis we need to run several analysis security methods. Each method has its own and independent model. Consequently, to conduct several methods, the model transformation remains very important. Model transformation plays a key role in model-based software development. It describes the relationship between models, more specifically the mapping of information from one model to another which allows traceability. This traceability allows tracking changes in models and how it affects other models.

(4) Some model-based environments (for instance, Papyrus (Gérard et al., 2011)) offer the possibility of code generation.

On the other side, implementing several methods in the same environment allows the reuse of information. This remains particularly difficult, when we use separate method dedicated tools as it requires a lot of experience and extra effort and may lead to inconsistencies between the different analyses. Besides, this approach allows to automatically generating skeletons or even a full body for the next methods to be applied. This is what we will show in the next section where we explain how to automatically generate partial Attack trees from an EBIOS analysis study.

3 MODEL DRIVEN APPROACH

In this section we present the process that we propose and the way to implement it.

3.1 Process Description

The process that we propose (see Figure 4) is described as follows:

Phase 1: We design or use an existing architecture design of the system.

Phase 2: We apply the EBIOS analysis by following the five phases described in the previous section (section 2.1): We first define the context. Second we define the Feared Events. Third, we describe the threats and relate them to the existing Feared Events so that for each defined Feared Event we have a list of threats where each of them can lead to the occurrence of this Feared Event. Fourth we appreciate the risk level for each Feared Event. Fifth, we consider the existing security countermeasures in the system to compute the residual risk level for each Feared Event. At this phase, we do not only apply a classical EBIOS analysis but we also relate it to the design architecture of the system. Actually, we relate the Feared Events, threats and vulnerabilities to the concerned assets (a component in the system architecture: function, software, hardware, etc.) in the system architecture. This allows visualizing the critical components in a system, and keeps the traceability between all the phases.

Phase 3: We automatically retrieve the Feared Events that have the residual risk level higher than a threshold that we have already fixed.

Phase 4: We generate an Attack tree for each Feared Event as follows: The top node of the tree is the Feared Event. This node is an OR node, and its children are the threats that lead to this Feared Event as defined in the EBIOS Analysis step (It is an OR node because as it is defined in EBIOS method any of these threats can lead to the Feared Event). Then those threats nodes are AND nodes based on the description in the EBIOS Analysis: Each Threat can exploit one or several vulnerabilities of an asset. Besides the generation of the structure of the tree we also generate some parameters that might help in the evaluation of the Attack tree: For instance for each vulnerability corresponding node we keep the concerned asset information (parameter *A*). This allows us to keep the traceability of the Attack tree with the system architecture. Second, in the EBIOS analysis we associate likelihood for each threat. But, as in Attack trees, parameters are associated only to leaf nodes that represent vulnerabilities in our case, we will associate to each vulnerability node a new parameter which is the set of likelihoods of all threats that can exploit this vulnerability (parameter *P*). Finally, another parameter is the countermeasures that are related to the assets in the EBIOS analysis. Then, as in an Attack tree, a vulnerability node is related to one asset, we can deduce the set of countermeasures that are concerned (parameter *M*). The structure of a generated tree is presented in Figure 5.

Phase 5: We can complete the tree in three manners. First we might add some new nodes: to add new threats that may be missed in the EBIOS Analysis step, or to detail some nodes, or maybe to go deeper in the description of the Attack tree (more than 3 levels depth). Second, we can add new parameters to the nodes. Actually, in the EBIOS analysis level the evaluation of risk is computed as described in section 2. It is based on the severity level parameter associated to a Feared Event and the likelihood parameters associated to threats. However we might define new formulas to have more precise evaluation or even to consider other criteria to evaluate the Attack tree. For instance, in the Magerit (Ministerio de Administraciones Publicas, 2006) method they also consider a third parameter to compute the risk which is vulnerability level, as for (Ten et al., 2008) their evaluation of the Attack tree is based on the existing countermeasures in the system only. Third we define the formula to compute the evaluation of the tree. We notice that we didn't define any formula by default to compute the evaluation of the tree we only extract the parameters from the EBIOS analysis phase that we consider that might be useful for the evaluation.

Phase 6: At this level we can compute the different indices based on the Attack tree and the parameters that we have associated to their nodes. Then we propose countermeasures to minimize indices that are higher than the threshold we decided.

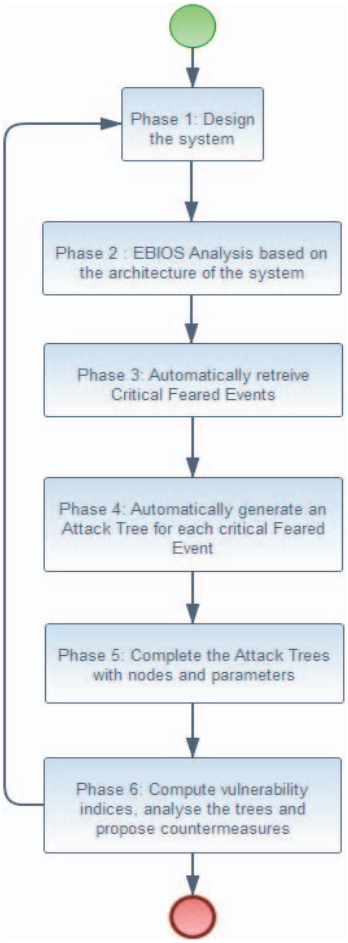

Figure 4: The structure of a generated Attack tree.

We model the system architecture in the Papyrus environment which is an Eclipse based environment supporting UML and SysML modelling standards. Besides, Papyrus can serve as a modelling platform for other tools dedicated to different analysis and specific domains and this by using UML profile (e.g. RobotML (Dhouib et al., 2012)). Profile is a powerful extension mechanism for UML that allows introducing specific concepts to the model.

We implement a framework that allows integrating security analysis methods into uniform

Papyrus. This framework includes profiles and tools to automate typical security analysis methods. Thus, we propose to implement EBIOS method and Attack Tree method based on UML profiles. We can also generate tables and documents as in the EBIOS Tool. Besides the complete analysis study and the architecture of the system can also be generated in documents that we can customize.

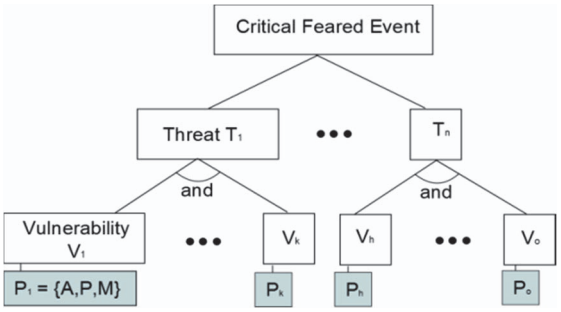

Figure 5: Generated Attack Tree structure.

4 RELATED WORKS

There are several model-based technologies and tools for security analysis in general. For instance, Coras, Magerit, Mehari, these methods similar to EBIOS, propose a model based analysis of the system. However, with these tools we can only model security concepts and Attack scenarios, but not the design of the system. For Attack trees, we can find several graphical tools, but the analysis is still independent of the system architecture and of other security analysis methods.

The use of UML Profiles becomes very widespread in several domains: In (Panesar-Walawege et al., 2013), the authors provide a generalizable and tool-supported solution to support the verification of compliance to safety standards IEC 61508. The Object Management Group (OMG) has standardized the UML Profile for Modelling and Analysis of Real-time and Embedded Systems (MARTE) (Bran and Gérard, 2014). In (OMG, 2003), authors present a UML Profile for modelling QoS and Fault Tolerance Characteristics and Mechanisms (QFTP), etc.

An existing framework named Sophia (Yakymets et al., 2013) based on UML Profiles, presents a similar approach but for safety. Sophia framework extends generic Papyrus environment to safety domain. Sophia includes facilities (i) to automatically perform various Safety Analysis methods (SA), (ii) to make semantic connections with formal SA tools, (iii) to represent SA results in the system modelling environment. Our approach is a security approach for Sophia framework.

Our framework considers a part of the RMF (Risk Management Framework) proposed by NIST SP 800-160 (National Institute of Standards and Technology, 2014). RFM provides a process that integrates information security and risk management activities into the system development life cycle. RFM considers 6 steps: (1) Categorize information system, (2) Select security controls, (3) Implement security controls, (4) Assess security controls, (5) authorize information system, (6) Monitor security controls. In our framework, we consider steps (1) and (2). Phases 1 to 5 in our Process are included in step (1): we describe the system, and the existing security controls (or countermeasures) of the system (Figure 4 - Phase 2: we consider these existing controls to compute the residual risk in the EBIOS analysis), and we compute the risk level. Phase 6 is included in step (2) where based on the previous step results we select the controls to apply. Documents related to the security plan considered by RFM can be partially generated as we don't have for the moment the implementation related to process management.

5 CONCLUSION, PERSPECTIVES

In this work, we propose a model-driven based framework for security analysis. We implement several security analysis methods into uniform Papyrus modelling environment as UML profiles. We choose to implement two methods: The first is a qualitative method named EBIOS which is widely used and that supports entirely ISO 27005 standard. This method allows identifying the most critical Feared Events for the system. Then we apply the second method we have implemented, the quantitative Attack trees method, which gives more accurate results

Our approach presents many advantages: from one side, it keeps the traceability with the design architecture. From the other side, it keeps the traceability in overall the security as the Attack trees are partially automatically generated from the EBIOS analysis phase. These generated Attack trees can then be completed to fit our needs from this study (adding parameters, formulas, etc.).

This work presents several perspectives: One perspective is to merge this framework with Sophia to get a safety and security. Another perspective is to

improve the qualitative analysis phase to be able to generate more complete Attack trees. Besides, many other methods can still be added to our process.

ACKNOWLEDGEMENTS

The work in this paper is funded by SesamGrids project (The consortium Sesam-Grids, 2012) and Risc project (The consortium RISC, 2013).

We want to thank the Phd student Anas Motii for his important participation to a part of this work.

REFERENCES

Bernardi, S., Merseguer, J., & Petriu, D. C. (2013). *Model-Driven Dependability Assessment of Software Systems*. Springer.

Bran, S., Gérard, S. (2014): Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE. Elsevier.

Stoneburner, G., Goguen, A., & Feringa, A. (2002). Risk management guide for information technology systems. *Nist special publication*, 800(30), 800-30.

Alberts, C., Dorofee, A., Stevens, J., & Woody, C. (2003). Introduction to the OCTAVE Approach. Pittsburgh, PA, Carnegie Mellon University.

Secrétariat Général de la Défense Nationale (2004). EBIOS- Expression des Besoins et Identification des Objectifs de Sécurité.

Gérard, S., Dumoulin, C., Tessier, P., & Selic, B. (2011). 19 Papyrus: A UML2 tool for domain-specific language modeling. In *Model-Based Engineering of Embedded Real-Time Systems* (pp. 361-368). Springer Berlin Heidelberg.

Mcdonald, J., Decroix, H., Caire, R., Sanchez, J., Chollet, S., Oualha, N., Puccetti, A., Hecker, A., Chaudet, C., Piat, H., others (2013): The SINARI project: security analysis and risk assessment applied to the electrical distribution network.

Basin, D., Clavel, M., & Egea, M. (2011, June). A decade of model-driven security. In *Proceedings of the 16th ACM symposium on Access control models and technologies* (pp. 1-10). ACM.

Panesar-Walawege, R. K., Sabetzadeh, M., & Briand, L. (2013). Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation. *Information and Software Technology*, 55(5), 836-864.

OMG, U. (2003). Profile for modeling quality of service and fault tolerance characteristics and mechanisms. Revised submission, *Object Management Group*.

den Braber, F., Hogganvik, I., Lund, M. S., Stølen, K., & Vraalsen, F. (2007). Model-based security analysis in seven steps—a guided tour to the CORAS method. *BT Technology Journal*, 25(1), 101-117.

Behnia, A., Rashid, R. A., & Chaudhry, J. A. (2012). A Survey of Information Security Risk Analysis Methods. *Smart CR*, 2(1), 79-94.

Gudemann, M., & Ortmeier, F. (2011, June). Towards model-driven safety analysis. In *Dependable Control of Discrete Systems (DCDS), 2011 3rd International Workshop on* (pp. 53-58). IEEE.

Schneier, B. (1999). Attack trees: Modeling security threats. *Dr. Dobb's Journal*, vol. 12, no 24, p. 21–29.

International Security Technology (IST), (2002). A brief history of CORA.

http://www.ist-usa.com Accessed 16-6-2013.

Karabacaka B, Songukpinar I., (2005), ISRAM: Information security risk analysis method, *Computer & Security*, March, pp. 147-169.

Ten, C. W., Liu, C. C., & Manimaran, G. (2008). Vulnerability assessment of cybersecurity for SCADA systems. *Power Systems, IEEE Transactions on,*23(4), 1836-1846.

Saini, V., Duan, Q. & Paruchuri, V., (2008). Threat modeling using Attack trees. *J. Comput. Small Coll*., 23(4), 124-131.

Ministerio de Administraciones Publicas (2006). Magerit - version 2 - Methodology for Information Systems Risk Analysis and Management - *Book I - The Method*, Madrid, 20 June.

Dhouib, S., Kchir, S., Stinckwich, S., Ziadi, T., & Ziane, M. (2012). Robotml, a domain-specific language to design, simulate and deploy robotic applications. In *Simulation, Modeling, and Programming for Autonomous Robots* (pp. 149-160). Springer Berlin Heidelberg.

Yakymets, N., Dhouib, S., Jaber, H., Lanusse, A. (2013). Model-driven safety assessment of robotic systems. In: *Intelligent Robots and Systems (IROS)*, 2013 *IEEE/RSJ International Conference on*, pp.1137-1142.

Secrétariat Général de la Défense Nationale (2010). EBIOS- Expression des Besoins et Identification des Objectifs de Sécurité, Méthode de Gestion des risques. http://www.ssi.gouv.fr/IMG/pdf/EBIOS-1-GuideMethodologique-2010-01-25.pdf.

The consortium Sesam-Grids (2012), The Sesam-Grids Project, In http://www.sesam-grids.org/.

The consortium RISC (2013), *The RISC Project*, http://risc.sec4scada.com/

National Institute of Standards and Technology (2014). Systems Security Engineering, An Integral Approach to Building Trustworthy Resilient Systems. *NIST Special Publication* 800-160.