ATSyRa: An Integrated Environment for Synthesizing Attack Trees

(Tool paper)

Sophie Pinchinat, Mathieu Acher, and Didier Vojtisek

IRISA/Inria
Campus de Beaulieu, 35042 Rennes Cedex, France
firstname.lastname@irisa.fr

Abstract. Attack trees are widely considered in the fields of security for the analysis of risks (or threats) against electronics, computer control, or physical systems. A major barrier is that attack trees can become largely complex and thus hard to specify. This paper presents ATSyRA, a tooling environment to automatically synthesize attack trees of a system under study. ATSyRA provides advanced editors to specify high-level descriptions of a system, high-level actions to structure the tree, and ways to interactively refine the synthesis. We illustrate how users can specify a military building, abstract and organize attacks, and eventually obtain a readable attack tree.

1 Introduction

Attack trees [8] provide a systematic way of describing the vulnerability of a system, taking various types of attacks into account. Strengths of attack trees rely on two aspects: they combine an intuitive representation of possible attacks with formal mathematical ways of analyzing them in a qualitative and quantitative way [6, 4]. Kordy et al. showed that attack trees have been extensively studied by the scientific community and are widely considered within the industry [5].

Up to now, analysts and technicians usually construct attack trees manually, based on their knowledge and experience. A large number of tools for editing and analyzing attack trees exist (see, e.g., [3, 4]). Unfortunately, the manual design of attack trees is time-consuming and error-prone, especially if the size of the attack tree becomes substantial. Moreover, a manual design is likely to be incomplete and unsound w.r.t. the security issues of a system under consideration. Supported by automation, practitioners can obtain large attack trees that are correct by construction and in line with the properties of the system. Moreover the generation process can also be reiterated in case new kinds of attacks emerge or the system evolves. As a consequence, automated generation of attack trees recently attracts the attention of researchers and industry practitioners [12, 9, 2, 11].

Specifically, our long-term objective is to develop a (semi-)automated process, applicable to a large panel of risk analysis domains (physical security, communication security and dependability, business, management, engineering, etc.), that will assist

practitioners in fulfilling the security modeling task. This paper presents ATSyRA[1] a tool for synthesizing attack trees. ATSyRA is built upon the mathematical foundations presented in [7]. Compared to [12, 9, 2, 11], ATSyRA aims to provide an interactive and user-guided synthesis; an integrated environment with domain-specific languages (DSLs) and advanced editors. We also aim to augment the level of abstraction and consider as input high-level description of a system for generating attack trees.

Remainder. Section 2 presents the underlying methodology. Section 3 illustrates the main features of ATSyRA. Section 4 identifies future work.

2 Towards Synthesis of Attack Trees

At the algorithmic level, we experienced that a naive fully automated generation is likely to produce unexploitable trees (because they are flat), as also noticed by [2]. Mauw and Oostdijk [6] and Kordy et al. [4] showed that numerous structurally different attacks trees can capture the same information, out of which a few are readable and meaningful for an expert. An original and crucial feature of our methodology is the support of *high-level actions (HLA)* [7] to specify how sequences of actions can be abstracted and structured – a high-level action can be seen as a sub-goal of the attacker.

The typical workflow is depicted in Figure 1: inputs, either given by the practitioners or generated by the tool, are depicted in round-corner boxes (1)-(4), and intermediate tools/transformations are depicted in rectangle boxes (a)-(b). Dashed arrows suggest partial automation and an involvement of users to generate the results.

3 ATSyRa: Tooling the Approach

We implement an environment, called ATSyRA, for realizing the methodology previously introduced. Our experience for assessing the physical security of military buildings[2] motivated its design. The tool assists practitioners in synthesizing attack trees from the high-level description of the system. In our case, we develop a *domain-specific language (DSL)* for expressing military buildings. Other DSLs can be considered as well. ATSyRA[3] is implemented on top of Eclipse and offers to experts different facilities (DSLs' services like editors and automated reasoning support). Box (0) in Figure 1 is a screen-shot of the ATSyRA environment, with windows ①-④, which we now detail.

① Experts define the system in a dedicated, textual or graphical language, called a Building specification, which is composed of three main parts: the *building description*, the attacker's *strength level*, and her *attack objective*.
  – The building description is entirely determined by a finite set of elements of four types: *zones* (rooms, garden, etc.), *accesses* (doors, windows, etc.), *items* (keys) and *alarms*. Each type of elements is equipped with an attribute, called its *defense level*, which determines the minimum strength attacker must possess in order to act on this very element.

[1] for "Attack-Tree Sythesis for Risk Analysis".
[2] in the context of a collaboration between IRISA and Defense Ministry in France (DGA).
[3] http://tinyurl.com/ATSyRA.

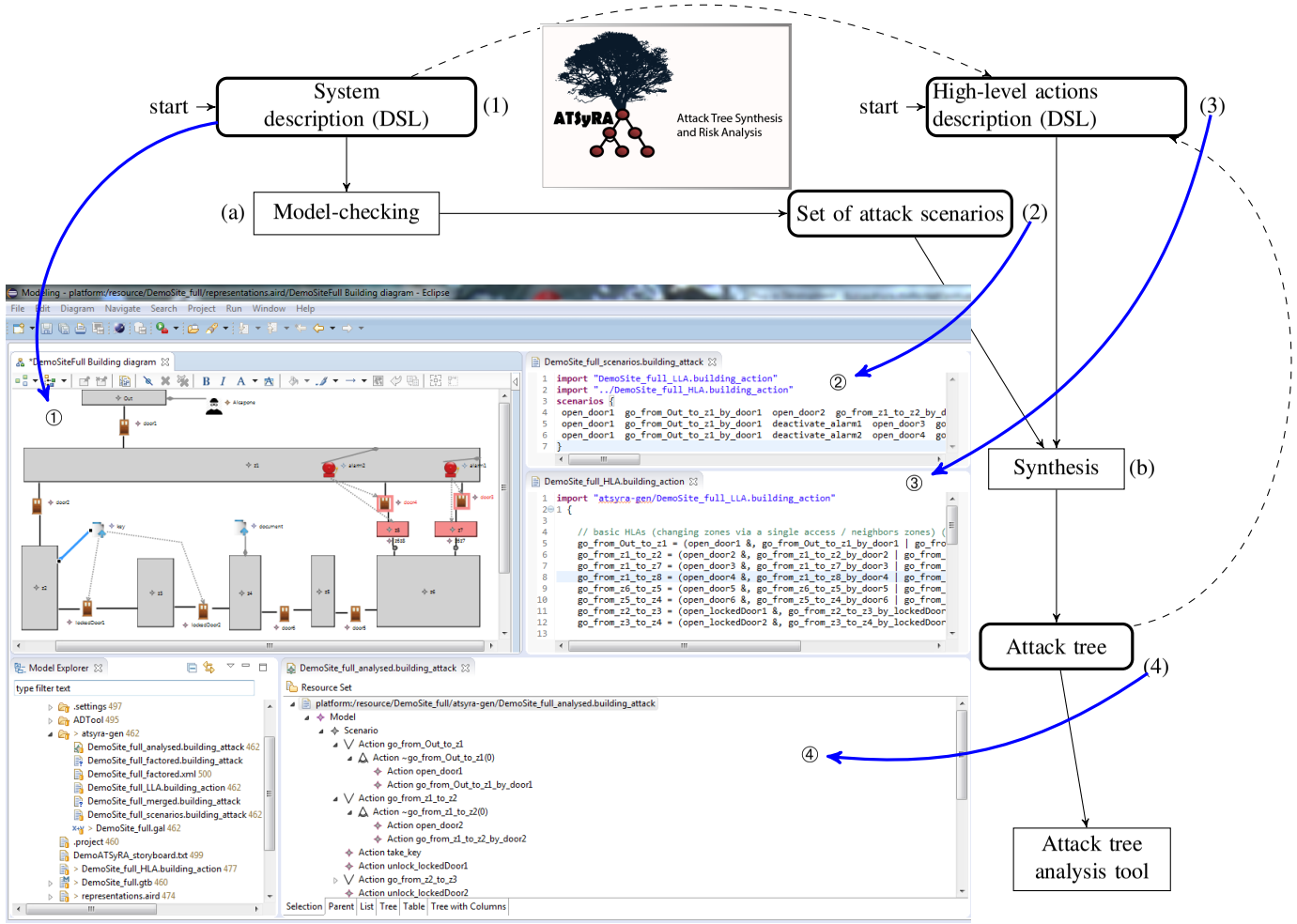

Fig. 1. The ATSyRA workflow

(0)

- The attacker's strength level is modelled by an integer value, that denotes her knowledge and skills necessary to execute a given action on a given element (such as opening a door, or using a key). This choice is by no mean a definitive one, but it is acceptable for the first version of the tool.
- The attack objective consists of a final zone to reach, with some items collected, and determines whether the scenario may be subject to detection by alarms.

② Experts then run the generation of the set of attack scenarios. The underlying process is the compilation of the Building specification into an attack graph. The transitions of this attack graph are labelled by *(atomic) actions* inferred from the building's elements description, and which are executable by the attacker (according to her strength level). The compilation process is highly compositional, allowing for the generation of a symbolic (hence very succinct) attack graph. The target language is GAL (for "Guarded Action Language") [1], a simple yet expressive formalism to model concurrent systems which is supported by a very efficient decision diagram library for model-checking [10]. ATSyRA notably exploits a tuned reachability analysis procedure. The objective is to produce the sequences of atomic actions that yield paths in the graph and that correspond to winning attack scenarios.

③ Experts specify a set of high-level actions (HLAs) with a dedicated, textual language. An HLA is described in terms of how it can be refined into less abstract actions. The formalism is inspired from context-free grammars [7]: HLAs are the non-terminal symbols of the grammar, atomic actions are terminal symbols, and refinements are derivation rules.

④ Experts eventually run the attack tree synthesis: this "final" step exploits both HLAs specifications ③ and generated attack scenarios ②. It relies on bottom-up syntactic analysis techniques for the context-free derivation rules given by the HLAs and input words given by the attack scenarios. Then, an algorithm (see details in [7]) merges the syntactic trees into the attack tree, the nodes of which have type ranging over disjunction, conjunction and sequential conjunction.

ATSyRA is developed using model-driven principles technologies (e.g., Xtext, Sirius). We can deliver almost for free advanced editors, being textual or graphical, with auto-completion, syntax highlighting, location of errors, etc. Experts that specify military buildings or HLA thus benefit from advanced and dedicated editing support. Another benefit is that our model-based tool is extensible. Other inputs for the high-level description of a system can be seamlessly integrated and come with advanced editors as well. For instance we are investigating the use of system description languages (e.g., SysML) as part of ATSyRA.

4 Conclusion

We presented ATSyRA, an environment built on top of Eclipse, to support a methodology for synthesizing attack trees. Starting from a military building, we illustrated how security experts can specify high-level actions and eventually generate readable and well-structured attack trees.

As future work, we plan to consider other inputs – beyond military building specification – in other fields (e.g., computer networks). As the synthesis process is likely to

be interactive and incremental, we plan to integrate as part of ATSyRA some visualisations and suggestions that can help an expert. We hope ATSyRA can be of interest for practitioners and researchers in charge of analyzing security risks with attack trees.

Acknowledgements. This work is funded by the Direction Générale de l'Armement (DGA) - Ministère de la Défense, France. We thank Salomé Coavoux and Maël Guilleme for their insightful comments and development around ATSyRA.

References

1. Colange, M., Baarir, S., Kordon, F., Thierry-Mieg, Y.: Towards Distributed Software Model-Checking Using Decision Diagrams. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification - 25th International Conference, CAV 2013. Lecture Notes in Computer Science, vol. 8044, pp. 830–845. Springer (2013)
2. Hong, J.B., Kim, D.S., Takaoka, T.: Scalable Attack Representation Model Using Logic Reduction Techniques. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications pp. 404–411 (2013)
3. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: ADTool: Security Analysis with Attack-Defense Trees. In: Joshi, K.R., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST. Lecture Notes in Computer Science, vol. 8054, pp. 173–176. Springer (2013)
4. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Attack–Defense Trees. Journal of Logic and Computation 24(1), 55–87 (2014)
5. Kordy, B., Pièvre-Cambacédès, L., Schweitzer, P.: DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. Computer Science Review 13–14(0), 1–38 (2014)
6. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Won, D., Kim, S. (eds.) International Conference on Information Security and Cryptology - ICISC 2005. LNCS 3935. Lecture Notes in Computer Science, vol. 3935, pp. 186–198. Springer (2006)
7. Pinchinat, S., Acher, M., Vojtisek, D.: Towards Synthesis of Attack Trees for Supporting Computer-Aided Risk Analysis. In: Canal, C., Idani, A. (eds.) Software Engineering and Formal Methods - SEFM 2014 Collocated Workshops: HOFM, SAFOME, OpenCert, MoKMaSD, WS-FMDS, Grenoble, France, September 1-2, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8938, pp. 363–375. Springer (2014)
8. Schneier, B.: Attack Trees : Modeling Security Threats. Dr. Dobb's Journal (1999)
9. Stéphane Paul: Towards Automating the Construction & Maintenance of Attack Trees: a Feasibility Study. In: Kordy, B., Mauw, S., Pieters, W. (eds.) GraMSec. EPTCS, vol. 148, pp. 31–46 (2014)
10. Thierry-Mieg, Y.: Symbolic Model-Checking Using ITS-Tools. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9035, pp. 231–237. Springer (2015)
11. TREsPASS: Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security, FP7 project, grant agreement 318003 (2012–2016), http://www.trespass-project.eu/
12. Vigo, R., Nielson, F., Nielson, H.R.: Automated Generation of Attack Trees. In: 2014 IEEE 27th Computer Security Foundations Symposium (CSF). pp. 337–350. IEEE (2014)