Attack Trees for Practical Security Assessment: Ranking of Attack Scenarios with ADTool 2.0

Olga Gadyatskaya, Ravi Jhawar, Piotr Kordy, Karim Lounis, Sjouke Mauw and Rolando Trujillo-Rasua

SnT, University of Luxembourg, Luxembourg
{name.surname}@uni.lu

Abstract. In this tool demonstration paper we present the ADTool2.0: an open-source software for design, manipulation and analysis of attack trees. The tool supports ranking of attack scenarios based on quantitative attributes entered by the user; it is scriptable; and it incorporates attack trees with sequential conjunctive refinement.

1 Introduction

Attack trees are a well-known and established methodology for security assessment that facilitates brainstorming, structures available information, and assists human experts in analysis [12, 11]. An attack tree is a graphical model, and as such it is better comprehensible than pure text-based approaches. However, graphical models require usable and efficient tools with suitable Graphical User Interfaces (GUIs) in order to be practical. Moreover, recent advances in automated risk assessment techniques now call for tool support to handle automatically generated attack trees with many thousands of nodes [3]. Therefore, the need for more comprehensive analysis tools emerged in the community. In this paper we present the ADTool2.0 that provides advanced capabilities for design, visualization, and analysis of attack trees [9], attack-defence trees [6], and attack trees with sequential conjunctive refinement (SAND attack trees for short)[4].

The new version of the tool brings in many new features, including ranking of critical attack scenarios, attack trees with the sequential AND (SAND) operator, and scriptability. It is not a simple extension of the previous tool [5], but a fully revamped, more advanced system. The ADTool2.0 has been reimplemented using the advanced cross-platform Docking Frames library [1]. It is an open source software freely available to the community[1].

2 Main features of the ADTool2.0

Sequential conjunct refinements in attack trees
The ADTool2.0 integrates a crucial modelling aspect: creation of attack trees with SAND refinements (consistent with the graphical language and semantics

[1] https://github.com/tahti/ADTool2

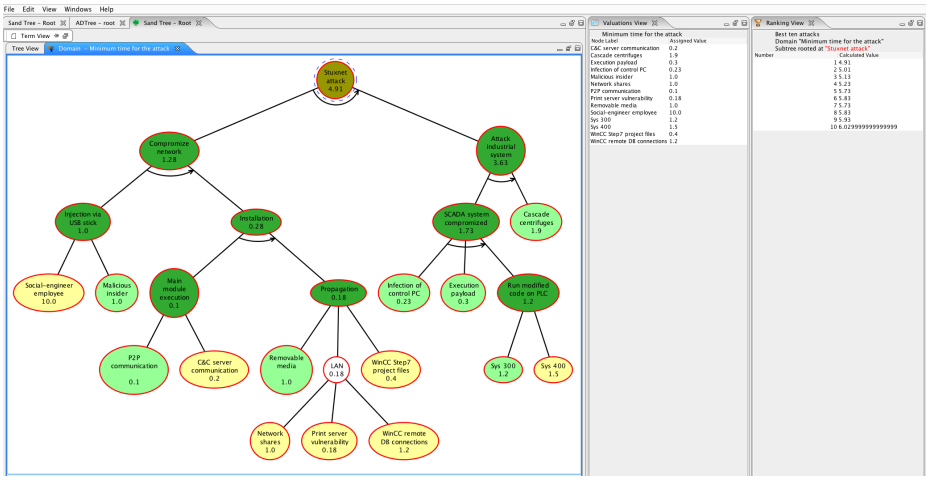

Fig. 1. Screenshot of the `ADTool2.0` with the ranking feature. The `SAND` attack tree used represents the Stuxnet attack, and the ranking is based on the minimal time of attack parameter. The attack scenario (all its attack nodes) with the minimal time of execution is highlighted in green by the tool.

described in [4]) and their quantitative analysis. Usage of the `SAND` refinement allows the analyst to model and analyze attack scenarios involving several attack steps that need to be all executed in a specific order, as opposed to the standard `AND` refinement used to model execution of several attack steps in parallel.

After constructing a `SAND` attack tree, the user can assign an attribute domain (e.g., minimum time for the attack, probability of success) to the tree. Each leaf node is then initialized with a default value representing the worst case scenario (e.g., ∞ as the minimum time for the attack), and all other nodes are automatically assigned using an n-ary function, depending on the type of attribute and refinement operator, in order to evaluate the security scenario. For example, to compute the minimum time needed for the attacker, $min(x, y)$, $max(x, y)$ and $x + y$ functions are used for `OR`, `AND` and `SAND` refinements respectively. The analyst can modify values at the leaf nodes and introduce new computation functions to meet her/his needs. The `ADTool2.0` will automatically compute new attribute values using a bottom-up algorithm.

Ranking attack trees

The human ability to visualize and understand attack trees quickly decreases with the increase in size and complexity of the latter. Identifying important portions of an attack tree is therefore of paramount importance for security analysts; it allows to prioritize and focus on those branches that contribute most to the attacker goal.

A systematic approach to prioritization is *ranking*, whereby a set of elements are sorted with respect to a total preorder. In attack graphs, a modelling language similar to attack trees, several ranking approaches have been defined [10,

8]. In attack trees, however, ranking has been mostly neglected by both quantification methods and tools. Remark that an attack tree may contain an exponential number of attack scenarios in terms of the tree depth, thus ranking cannot be straightforwardly performed by a sorting algorithm.

The ADTool2.0 implements an efficient and formal approach to rank attack scenarios. In particular, we have extended the bottom-up computation approaches proposed for attack trees [9], attack-defence trees [6], and SAND attack trees [4], in order to efficiently rank attack scenarios, where an attack scenario is either a *bundle* as in the formalisms in [9, 6] or an *SP graph* as in [4].

Our approach works intuitively as follows. Given a set of quantitative values V for attack scenarios and a total order $\leq$ on V, we store at every node of the tree n the least attacks with respect to the total order $\leq$, where n is a natural number representing a bound on the number of attack scenarios to be ranked. Soundness of the ranking method is ensured when: i) the function used in the bottom-up computation of the OR gate is an extrema function with respect to $\leq$, and ii) the functions used in the bottom-up computation of the AND and SAND gates are monotonic with respect to $\leq$.

Ranking results in the ADTool2.0 are shown in the *Ranking View* window, which can be opened from the menu Windows $\rightarrow$ Ranking View. As in the Attribute window, the Ranking window gives the option to open or create an attribute domain. By default, the ADTool2.0 uses as a total order the operator assigned to the OR gate in the attribute domain. A screenshot of the ADTool2.0 provided in Fig. 1 shows an example of the ranking feature applied to a SAND attack tree modelling the Stuxnet attack (inspired by [7]).

In order to rank attack scenarios up to a given node in the tree, we ought to click that node in the domain for which we want to see the ranking. Doing so, the Ranking view window will automatically update with a table containing optimal attacks with respect to the chosen attribute domain. The ADTool2.0 also offers the option to highlight those nodes that contribute most to the attack, what can be done by clicking on attack scenarios in the ranking table.

Scripting

Scriptability is an important capability for security assessment tools, as it allows integration into tool chains and *sensitivity analysis*, whereby the ADTool2.0 can be used to automatically assess how changes in some attribute values affect the overall security posture (and what are the changes to the most critical attacks). It is now also possible to experiment with countermeasure selection: we can write scripts that will input several attack-defence trees with different defence scenarios applied to a particular attack, and output the best countermeasure set based on the results of the ranking.

In the scripting mode (e.g., executed from the command line) the ADTool2.0 can accept attack trees in all supported flavors as input, and output, e.g., N the most critical attacks, or an attribute domain with calculated values[2]. Additionally, the ADTool2.0 is integrated into the TREsPASS project tool chain [2],

[2] Basic directions on running ADTool2.0 from the command line can be obtained by executing `java -jar ADTool-2.0.jar --help`.

where it is used to visualise attack-defence scenarios and produce ranked attacks for automatically generated attack trees [3].

Usability Features

The `ADTool2.0` includes many usability features, e.g., copy-paste of subtrees, handling of multiple trees, reorder of children nodes, and extended input format (automatically generated attack trees [3] not conforming to the `ADTool2.0` XML schema). The `ADTool2.0` can handle and analyze *large trees* with several thousand nodes (automatically generated trees are typically of that size).

3 Conclusion

In this tool demonstration paper we presented the main features of the `ADTool2.0` that is an open-source software for displaying, designing and analyzing attack trees in many flavors (`SAND` attack trees [4], attack-defence trees [6], and normal attack trees [9]). The `ADTool2.0` allows to rank attack scenarios based on the quantitative values selected by the end-user (e.g., time of attack, cost, probability, and so on). It can also be scripted for integration in tool chains.

References

1. `http://www.docking-frames.org/`
2. `http://www.trespass-project.eu/`
3. Ivanova, M.G., Probst, C.W., Hansen, R.R., Kammuller, F.: Transforming graphical system models to graphical attack models. In: Proc. of GraMSec 2015. LNCS, vol. 9390. Springer (2016)
4. Jhawar, R., Kordy, B., Mauw, S., Radomirović, S., Trujillo-Rasua, R.: Attack Trees with Sequential Conjunction. In: Proc. of IFIP SEC. pp. 339–353 (2015)
5. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: ADTool: Security analysis with attack-defense trees. In: Proc. of QEST (2013)
6. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Attack–Defense Trees. Journal of Logic and Computation pp. 1–33 (2012)
7. Kriaa, S., Bouissou, M., Pietre-Cambacedes, L.: Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments. In: Proc. of CRiSIS (2012)
8. Lu, L., Safavi-Naini, R., Hagenbuchner, M., Susilo, W., Horton, J., Yong, S.L., Tsoi, A.C.: Ranking attack graphs with graph neural networks. In: Proc. of ISPEC. pp. 345–359 (2009)
9. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Won, D., Kim, S. (eds.) ICISC'05. LNCS, vol. 3935, pp. 186–198. Springer (2006)
10. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Proc. of RAID. pp. 127–144. Springer-Verlag (2006)
11. NATO Research and Technology Organisation: Improving common security risk analysis (2008)
12. OWASP: CISO AppSec Guide: Criteria for managing application security risks (2013)