Attacker profiling in quantitative security assessment based on attack trees

Aleksandr Lenin[1,2], Jan Willemson[1] and Dyan Permata Sari[2] *

[1] Cybernetica AS, Mäealuse 2/1, Tallinn, Estonia
[2] Tallinn University of Technology, Ehitajate tee 5, Tallinn, Estonia

Abstract. Providing meaningful estimations for the quantitative annotations on the steps of complex multi-step attacks is hard, as they are jointly influenced by the infrastructure and attacker properties. The paper introduces *attacker profiling* as the concept of separation of the infrastructure properties from the properties of malicious agents undertaking strategic decisions in the considered environment. We show that attacker profiling may be integrated into existing quantitative security assessment tools without any significant performance penalty. As an example of such integration we introduce the new analysis tool named ApproxTree+ which is an extension of the existing ApproxTree tool, enhancing it by incorporating attacker profiling capabilities into it.

1 Introduction

Targeted malicious attacks are intentional by their nature and may be interpreted as sequences of actions (attack steps) performed by malicious agents undertaking informed strategic decisions in the target infrastructure. This way we can distinguish between the two landscapes – the one which we call the *threat landscape* and the *vulnerability landscape*. The threat landscape is formed by various kinds of malicious agents – they have different sets of properties, available resources, varying intentions, motivations, views, and expectations of the target infrastructure. These properties determine strategic preferences of the agents, and eventually their behavior. The vulnerability landscape is formed by the infrastructure of the organization, its employees, assets, policies, processes, etc. Both landscapes are dynamic by their nature and are constantly changing. The threat landscape may change due to the agent behavior (e.g.

* This research was supported by the European Regional Development Fund through Centre of Excellence in Computer Science (EXCS), the Estonian Research Council under Institutional Research Grant IUT27-1 and European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement ICT-318003 (TREsPASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.

increase in resources available to the agent) as well as external events, while the vulnerability landscape may change due to the infrastructure updates (e.g. patching, component replacement, awareness training, deployment of defensive measures, etc.) as well as unintentional events.

We propose the separation between the infrastructure properties (the vulnerability landscape) and the adversarial properties (the threat landscape), represented by an *attacker profile*. This separation adds flexibility to the quantitative security analysis enabling the assessment of operational security risks using different combinations of attacker profiles and infrastructure properties providing much deeper insight on the surrounding risk landscape. Besides, attacker profiling increases the reliability of the analysis results as the separation of infrastructure properties and attacker properties allows to update these values in a timely manner independently from each other and reflect the ever changing risk landscape in a more reliable way.

The paper aims at introducing attacker profiling in the context of quantitative security analysis based on attack trees and demonstrates integration of attacker profiling into existing security assessment tools introducing the new tool named ApproxTree+. In the introduced ApproxTree+ model the considered infrastructure properties (cost, difficulty, minimal required attack time) are quantitative annotations on the attack tree leaves, while the adversarial properties (budget, skill, available time) are described by attacker profiles. Additionally we compare the performance of the profiling computations to the ApproxTree approach [1] and reassess if the genetic algorithm parameters, used by ApproxTree for fast approximations, are optimal for the profiling computations.

The outline of the paper is the following: Section 2 outlines the state of the art in quantitative security assessment, attack trees, and attacker profiling. Section 3 describes motivation for the attacker profiling in security risk assessment. Section 4 introduces the ApproxTree+ tool, while Section 5 outlines the tool performance analysis results. Section 6 briefly lists the achievements made so far and outlines areas for future research.

2 Related Work

2.1 Attack trees

Attack trees as one of the ways of quantitative security assessment, evolved from fault trees [2] and were popularized by Schneier [3] who suggested to use them as a way to model security threats and to perform quantitative security assessment using this convenient hierarchical representation by means of bottom-up single parameter propagation. Quantitative security assessment has been studied by various researchers [4–8] and different variations of techniques and methodologies were suggested.

Buldas *et al.* [9] suggested to use multi-parameter approach instead of the historical single-parameter one and applied economic reasoning by propagating adversarial utility. This kind of analysis allowed to assess whether the analyzed system is secure against targeted rational profit-oriented attacks.

Jürgenson and Willemson improved the model of Buldas *et al.*, making their parallel [10] and serial [11] models consistent with Mauw and Oostijk foundations [12] and introducing genetic approach to speed up computations. The parallel model assumed that the attacker launches attack steps, required to fulfil the attack scenario, simultaneously, while the serial model assumed that an attacker launches the attack steps in a predefined order.

Later, Buldas and Stepanenko introduced the failure-free model [13] suggesting not to limit the adversary in any way and thus analyzing fully adaptive adversarial utility upper bounds. This approach was later improved by Buldas and Lenin [14]. Their model better conforms to the upper bounds ideology and is computationally less complex.

For a more thorough overview of the quantitative security analysis using attack trees we refer the reader to [15].

2.2 Attacker profiling

Back in 1998, Philips *et al.* [16] outlined the importance of the attacker feature in attack graphs for network vulnerability analysis. Several research projects have focused on attacker profiling using honeypot in "Know Your Enemies" series [17–19] which outlined the range of techniques and tools that were used by attackers for reconnaissance and also motives of the blackhat community. Several researchers proposed the concept of attacker personas, which was related to goal, motivations, attitudes, and skills [20–23]. Faily *et al.* highlighted insider threat motivations and characteristics, as well as the use of attacker personas for threat identification.

Pardue *et al.* [24] mentioned the importance of attacker characteristics and also the complexity of the attacks assessing risks of an e-voting system. The authors argue that the likelihood of attacks can be referred to as *cost* of an attacker, which can be estimated on various scales and measured in various units, such as dollars, number of attackers, time invested into attacking, and effort. In addition, Sallhammar *et al.* [25]

demonstrate the process of deriving the probability of the expected attacker behavior in assumption that the attacker has complete information about the vulnerabilities of the targeted systems. Tipton *et al.* [26] argue that risk aversion, degree of difficulty, discoverability, ease of access, effectiveness of controls, effort, incentive, interest, skill level, motivation, resources required, risk of detection, and special equipment needed are the factors that can be included in attacker profiling. There are some common parameters that are most often used in research projects to define an attacker profile – these values are more feasible for quantitative analysis and give clear understanding of attacker properties.

2.3 Parallel model

The parallel model [10] by Jürgenson *et al.* allows to assess whether the analyzed system is secure against targeted rational profit-oriented attacks by assessing adversarial utility. In case the utility is positive, the system is considered to be insecure, as profitable attack vectors which may result in positive outcome for an attacker are likely to exist. Otherwise the analysis assumed that the system is reasonably secure to withstand emerging attacks.

An attack scenario, represented by an attack tree, is treated as a monotone Boolean function, each variable of which corresponds to a leaf node in the attack tree, and logic operators correspond to the refined nodes in the attack tree. The successful outcome of an elementary attack is modelled by assigning value 1 to the corresponding variable in the Boolean function. If the Boolean function is satisfied, the attacker has succeeded in the security scenario. More complex multi-step attacks are modelled as *attack suites.*

The computational method maximizes the adversarial utility over the entire set of satisfying attack suites. The complexity of the approach arises from the need to process the entire set of 2^n attack suites, which introduces unnecessary overhead. Even with the optimizations proposed [10] this approach was able to analyze attack trees of at most 20 leaves in reasonable time which has made this method inapplicable for the practical case analysis.

To overcome limitations of the parallel model [10], a set of further optimizations was proposed by Jürgenson *et al.* [1] and implemented in the tool later called ApproxTree.

More significant contribution of the paper is the development of genetic algorithm for fast approximations, which increased performance compared to [10]. The implementation of the approach described in the paper reached 89% confidence[1] level within 2 seconds of computation for the tree having up to 29 leaves. As the genetic algorithm is very scalable it has potential to be used for the analysis of practical attack trees containing more than 100 leaves. The computational complexity of the suggested approximation algorithm in the worst case was estimated to be $\mathcal{O}(n^4)$. The authors have performed benchmarking tests and experimentally derived the optimal set of values for genetic algorithm parameters.

3 Motivation for the attacker profiling

An attack tree is a hierarchical description of possible attacks against the target infrastructure. Constructing an attack tree, analysts include all possible attack scenarios in the tree. Some of them are more realistic, some are less, considering the environment in which such a system is deployed. This way, attack tree analysis assumes an overpowered attacker who is capable of launching every possible attack, included in the attack tree, against the system. However, real life attacks are, as a rule, not so powerful and thus analysis assuming the almighty adversary concept does not provide deep insight on the security risks taking into account the surrounding risk landscape. Applying attacker profile to the attack tree invalidates certain nodes and eventually entire subtrees in the initial attack tree, thus enabling the independent analysis of the derived attack scenarios, containing attacks feasible for the considered class of malicious agents. Depending on the severity of adversarial limitations used in the profile, the derived attack scenario may be much smaller and thus much easier to analyze.

Quantitative security analysis relies on quantitative annotations (e.g. likelihood of success in an attack step, time required to launch an attack step, etc.) assigned to single attack steps in complex multi-step attacks. We believe that the quantitative metrics of these annotations is jointly influenced by various sets of underlying components in threat- as well as vulnerability landscapes. Thus it is rather difficult to provide a trustworthy and reliable quantitative estimation for such parameters as it is practically impossible to estimate the cumulative effect of several underlying factors altogether. Such kind of joint estimations are, as a rule, imprecise and contain reasonable degree of uncertainty.

For example, it is almost impossible to provide a meaningful estimation for the time parameter, as the time, required for an attack step,

[1] By confidence authors mean the ratio of the trees actually computed correctly by the suggested approximation technique, compared to the precise outcome.

depends on the attacker skills, capabilities, available resources, previous experience, etc. (agent properties), as well as on the difficulty of the attack step itself (infrastructure property). Similarly, the likelihood of success depends on attacker skill, difficulty of the attack step, and time invested into attacking. The more skilful and experienced the attacker is, the more likely he is to succeed in an attack step. The more resources are available to the attacker, the more likely will he be successful in an attack step. Similar reasoning may be applied to the skill parameter – the more experienced the attacker is, the less difficult is the process for him, the less time it will take to succeed in an attack step. Less skilled attacker, given sufficient time, may be as efficient (in terms of likelihood of success) as a more skilled attacker who has less time for attacking. Similar logic may be applied to other parameters as well.

Despite that, the analysis has to deal somehow with the ever changing nature of each of the landscapes mentioned above and update (or reassess) the estimations of the corresponding quantitative annotations in a timely manner. It is unclear how to update such joint estimations in case some of its components change while the others remain unchanged, or, on the contrary, when all its components change.

In order to tackle the difficulties outlined above the propose attacker profiling as a step forward in dealing with the challenges of security metrics.

4 The ApproxTree+ model

We introduce the ApproxTree+ model – the new model for quantitative assessment of operational security risks. The computational method is built on the logic of the parallel attack tree model [10] and fast approximations of ApproxTree [1], improved by adding attacker profiling considerations into the method.

4.1 Definitions

We will use the same notation as in [10]. Let us have a set of all possible elementary attacks $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$, and a Boolean function $\mathcal{F}$ corresponding to the attack tree.

Definition 1 (Attack Suite). *Attack suite $\sigma \subseteq \mathcal{X}$ is a set of elementary attacks which have been chosen by the attacker to be launched and used to try to achieve the attacker goal.*

Definition 2 (Satisfying attack suite). *A satisfying attack suite σ evaluates $\mathcal{F}$ to* true *when all the elementary attacks from the attack suite σ have been evaluated to* true*.*

Definition 3 (Attacker profile). *An attacker profile is a pair $(t, \mathcal{P}_t)$ where t is an n-tuple of attacker properties $(p_1, p_2, \ldots, p_n)$ and a function $\mathcal{P}_t(\sigma)$ defined by t which takes an attack suite σ as input and returns* true*, iff the attacker with the considered properties t is capable of launching all the attacks in σ, and* false *otherwise.*

Each of the elements p_k in t belongs to a certain domain P_k which provides quantitative metrics to the parameter. Some of the domains may represent continuous values, e.g. money, so we can take $P_k = \mathbb{R}$. Others parameters may be measured on an ordinal scale to reflect the magnitude and measured in levels e.g. High, Medium, and Low, in which case $P_k = \{H, M, L\}$.

In our research we use the following attacker properties:

1. *Budget* $t_b \in \mathbb{R}$ – the monetary resource of the attacker, measured in currency units.
2. *Skill* $t_s \in \{L, M, H\}$ – the skill level of the attacker, measured on an ordinal scale (Low/Medium/High).
3. *Time* $t_a \in \{S, MT, HR, D\}$ – the available time resource of the attacker, measured on an ordinal scale (Seconds/Minutes/Hours/Days).

The attacker properties outlined above define function $\mathcal{P}_f(\sigma)$, which returns *true* iff:

1. $t_b \geqslant \sum_{i=1}^{n} \mathrm{Cost}(\mathcal{X}_i)$,
2. $\forall \mathcal{X}_i \in \sigma : \ t_s \geqslant \mathrm{Difficulty}(\mathcal{X}_i)$, and
3. $\forall \mathcal{X}_i \in \sigma : \ t_a \geqslant \mathrm{Time}(\mathcal{X}_i)$.

Definition 4 (Profile satisfying attack suite). *A profile satisfying attack suite σ is a satisfying attack suite which satisfies all the constraints of the chosen attacker profile $(t, \mathcal{P}_t)$.*

4.2 Description of the approach

The analysis method can be described by the following rules [10]:

1. The attacker constructs the attack tree and evaluates the parameters of each of the elementary attacks following these considerations:

  - The attacker has to spend $Cost_i$ resources to prepare and launch an attack $\mathcal{X}_i$.
  - The attack $\mathcal{X}_i$ succeeds with probability p_i and fails with probability $1 - p_i$.
  - Depending on the detective security measures, the attacker sometimes has to carry additional costs after failing or succeeding with the attack. The sum of preparation and additional costs is denoted as $Expenses_i$ parameter.
  - Additionally, there is global parameter $Profit$ for the whole attack scenario, which describes the benefit of the attacker, in case the root node is achieved.
2. The attacker considers all potential attack suites – subsets $\sigma \subseteq \mathcal{X}$, where $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ is the set of all elementary attacks considered in the attack scenario. Some of the attack suites satisfy the Boolean function $\mathcal{F}$, some do not. For the satisfying attack suites the attacker computes the outcome value $Outcome_\sigma$.
3. Finally, the attacker chooses the most profitable attack suite and launches the corresponding elementary attacks simultaneously.

The computational method presented in [10] aims at maximizing the expression

$$Outcome_\sigma = p_\sigma \cdot Profit - \sum_{\mathcal{X}_i \in \sigma} Expenses_i$$

over all the assignments $\sigma \subseteq \mathcal{X}$ that turn the monotone Boolean function $\mathcal{F}$ to true. The success probability of the primary threat p_σ can be computed in time linear in the size of elementary attacks n:

$$p_\sigma = \sum_{\substack{\mathcal{R} \subseteq \sigma \\ \mathcal{F}(\mathcal{R}:=true)=true}} \prod_{\mathcal{X}_i \in \mathcal{R}} p_i \prod_{\mathcal{X}_j \in \sigma \setminus \mathcal{R}} (1 - p_j) \ . \tag{1}$$

In order to tackle the potential exponential amount of computations in (1), a genetic algorithm was proposed and benchmarked by Jürgenson *et al.* [1].

4.3 Approximation

The ApproxTree+ method uses the genetic algorithm to facilitate the usage of the computational method for large attack trees:

1. Create the first generation of n individuals (profile satisfying attack suites, not all of them are necessarily distinct).

2. All the individuals in the initial population are crossed with everybody else producing $\binom{n}{2}$ new individuals.
3. Each individual is mutated with probability p.
4. The mutated population is joined with the initial population.
5. Finally, n fittest profile satisfying individuals out of the $\binom{n}{2} + n$ individuals are selected and form the next generation.

The reproduction phase terminates when k last generations do not increase outcome. The complexity of the suggested approach was measured to be approximately $\mathcal{O}(0.85^n)$ using exponential regression.

5 Performance analysis

In order to assess the performance of the introduced computational method we have randomly generated a set of attack trees. The attack tree generation procedure was a two-step process. First, the random Boolean function with the predefined number of variables (leaves in the attack tree) was generated. It contained from 2 to 5 operands per operator – the values of operands in each case were chosen randomly. The next step was to provide quantitative annotations on the leaves of the attack tree. These values were chosen randomly from the predefined intervals: the *cost* parameter was estimated in the interval $[100, 1000]$, the *success probability* parameter was estimated in the interval $(0, 1)$. The value for the *difficulty* parameter was chosen from uniformly distributed values *low*, *medium*, *high*, and *very high*. The value for the *time* parameter was chosen from uniformly distributed values *seconds*, *minutes*, *hours*, and *days*[2].

One of the questions that needs to be answered is if attacker profiling adds extra computational overhead. It can be seen on the cumulative time distribution diagram (see Fig. 1) that attacker profiling does not add any significant computational overhead (in the case of a single attacker profile being analyzed) compared to the ApproxTree approach (see Fig. 2). In both methods the initial population generation phase is almost immediate, as well as the mutation phase. The main workload is performed by the crossover phase and consumes approximately 85-99% of the cumulative time distribution among all the phases. The last phase, the best individuals selection phase, does not introduce any significant workload and consumes approximately 1 - 15%. The crossover phase is the most time consuming as each individual is crossed with every other individual in the population producing $\mathcal{N} \times \mathcal{N}$ cross operations, where

[2] assuming uniform distribution of the PRNG output

$\mathcal{N}$ is the amount of individuals in the initial population. Fig. 3 shows that the execution time of the ApproxTree+ approach is proportional to the ApproxTree approach. The increased execution time arises from the fact that, as a rule, one doesn't assess risks using just a single adversarial profile, as it is reasonable to assess risks using the entire set of possible adversarial profiles so that the results would produce meaningful insight on the risk landscape – thus the overall execution time is proportional to the number of the attacker profiles under consideration.

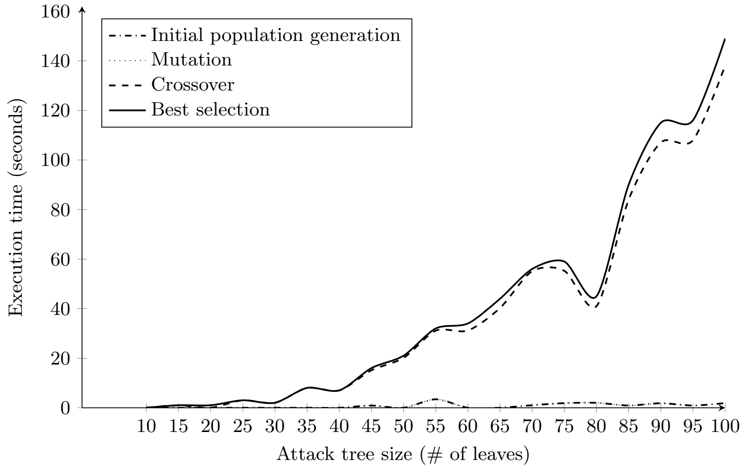

Fig. 1. Cumulative time distribution of ApptoxTree+ phases.

The analysis of the speed of convergence shows that the convergence speed of ApproxTree does not exceed the convergence speed of ApproxTree+. Additionally, it does not depend on the size of the attack tree – independently of the size of the tree, the convergence speed stays approximately at the same level.

Additionally, we have analyzed the effect of the genetic algorithm parameters such as mutation rate and initial population size on the convergence speed to assess whether the parameters of the genetic algorithm used by ApproxTree [1] are optimal for the ApproxTree+ approach.

The convergence speed decreases with the increase in the percentage of mutations from approximately 2 generations in the case when the mutation rate is 10% up to 6 generations in the case when mutation rate is 90% (see Fig. 5). Independently of the mutation rate, the speed of con-

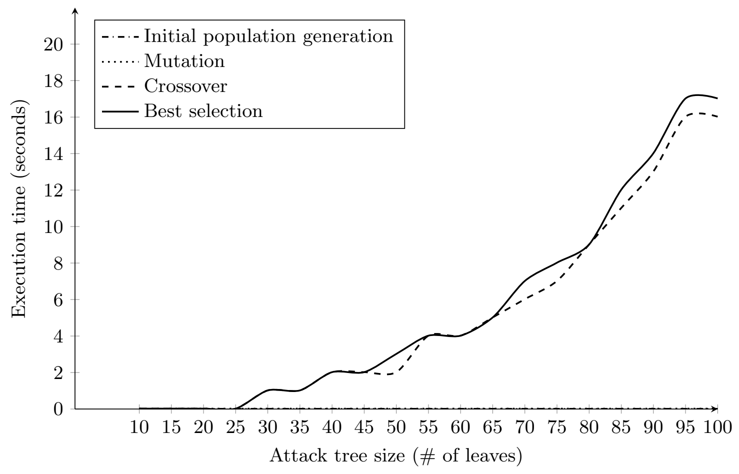

Fig. 2. Cumulative time distribution of ApproxTree phases.

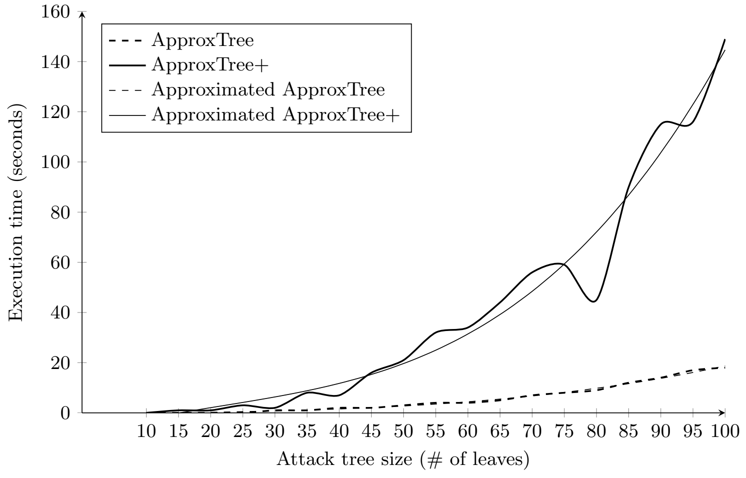

Fig. 3. Execution time.

vergence of ApproxTree+ does not exceed the speed of convergence of ApproxTree.

Benchmarking results have shown that the mutation step has no significant effect on the convergence speed at all. We were unable to find any case where the method would get stuck in the local optimum. Even when the mutation step was excluded entirely (as a phase of the genetic algorithm) – the global optimum was always reached. This may happen because of "good" initial population generation – if the size of the

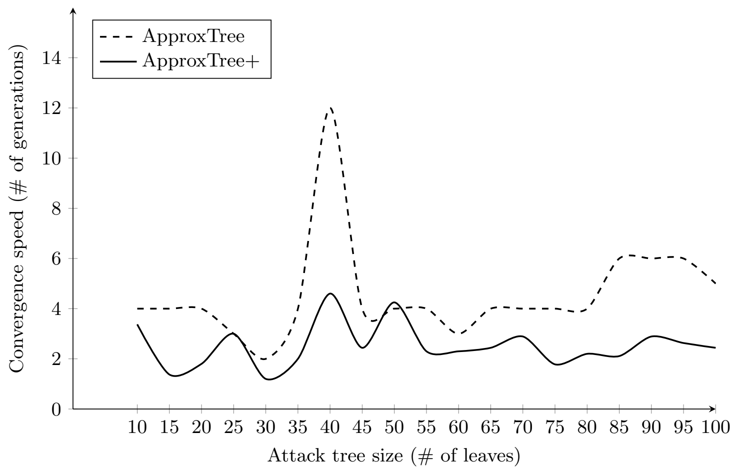

Fig. 4. Convergence speed.

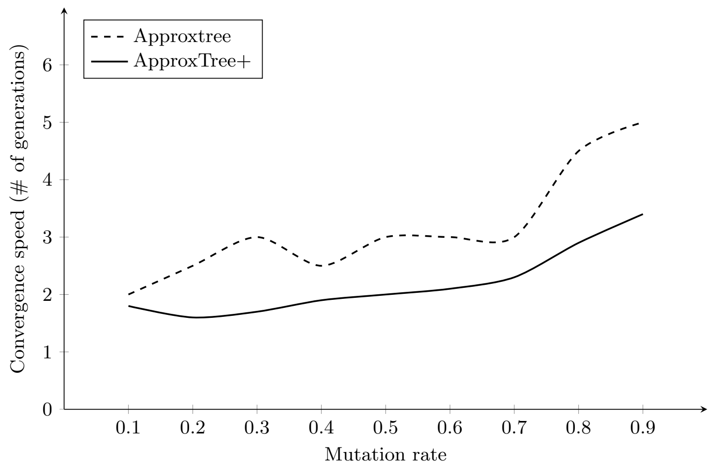

Fig. 5. Convergence speed as a function of the mutation rate.

initial population is rather big compared to the number of satisfying solutions, it is highly likely that the initial population will contain all the solutions (profile satisfying attack suites). In this case the convergence is immediate, which was observed in some cases during benchmarking. If the initial population does not contain all the solutions, still it may be "good enough" so that the crossover step produces the entire domain of solutions.

With the increase in the initial population size (see Fig. 6) the convergence speed increases, stabilising at a value of approximately 1.6 genera-

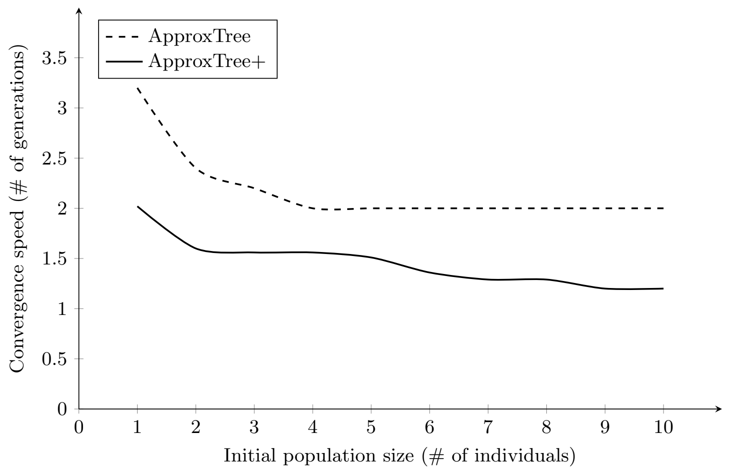

Fig. 6. Convergence speed as a function the size of the initial population.

tions for the initial population size greater than $4n$ (n being the number of leaves in the attack tree) in the case of the ApproxTree approach. In the case of ApproxTree+ we can see slight, but firm decrease in the convergence speed. In some cases when initial population size was less than n the computational method was unable to reach global optimum, which may happen when rather small initial population limits the amount of possible solutions that may be reached and the mutation rate is small enough and does not improve the situation.

The precision assessment shows that in ApproxTree+, as well as in ApproxTree, either the result converges to the global optimum (most profitable attack suite) or the computational method fails to generate the initial population of individuals. In case of profiling, the attacker profile may contain so strict constraints that not a single profile satisfying attack suite may exist. The more strict constraints are used in the considered attacker profiles the higher is the probability that no profile satisfying assignments will be generated. However we are unable to state that the profile satisfying solutions definitely do not exist in this case, as the state when ApproxTree+ is unable to generate the initial population means 2 possible conditions – either no profile satisfying attack suites exist (and thus the considered attack scenario has no profitable solutions), or such attack suites exist, however the attack suite generation procedure failed to generate profile satisfying solutions due to the stochastic nature of the process.

6 Conclusions and Future Research

Attacker profiling is a way to separate infrastructure properties and the properties of the malicious agents who are undertaking strategic decisions in the target infrastructure. This kind of separation allows to estimate and assess these properties independently from one another. This allows to derive meaningful values for the quantitative annotations on the attack steps in complex multi-step attacks from the underlying properties instead of providing joint estimations to these values directly. One can more precisely estimate how would a complex value change in case when some of its underlying components change. In example, how would the likelihood of success in an attack step change if instead of profit-oriented malicious individuals we face organized groups of attackers or a national security agency and the target infrastructure was patched meanwhile and the employees have received an awareness training? Thus, attacker profiling enables more detailed assessment of the impact of the fluctuations in threat and vulnerability landscapes on the values of the quantitative annotations on the attack steps.

Additionally, it adds flexibility to the analysis in general, enabling analysis using different combinations of attacker profiles and infrastructure properties, making comprehensive risk assessment possible. It provides broader and more detailed overview of the risk landscape in a timely manner, following constant changes in the risk environment. It allows to make informed decisions in assessing the cost-effectiveness of the defensive measures and enabling the prediction, prioritization and prevention of emerging attacks in nearly semi-automated way.

We introduced the attacker profiling and demonstrated the application of profiling in the framework of attack tree analysis by introducing the new analysis tool named ApproxTree+ and demonstrating that integrating attacker profiling into an existing analysis method does not introduce any significant performance penalty.

The constraint based approach, outlined in the paper, is only one possible interpretation of attacker profiling. Another possibility is to apply Item Response Theory to represent the relation between various underlying components in the threat and vulnerability landscapes. Such a relation may be represented, in example, in the form of a logistic function in its simplest form indicating that the likelihood of success will be assigned value 0.5 when the skill (β) and difficulty (γ) are equal: $p = \left(e^{\beta-\delta}\right)/(1+e^{\beta-\delta})$. In more complex scenarios the function may be extended to take 3 arguments, as the likelihood of success depends on the invested time parameter as well, and in this case it will take the form of: $p = f(\beta, \delta, \gamma)$ where γ is the time invested into attacking.

We see the way forward in implementing the above mentioned interpretation of profiling, integrating ApproxTree+ in the existing risk assessment frameworks and tools, and validating the approach in real-case risk analysis.

References

1. Jürgenson, A., Willemson, J.: On Fast and Approximate Attack Tree Computations. In Kwak, J., Deng, R.H., Won, Y., Wang, G., eds.: ISPEC. Volume 6047 of Lecture Notes in Computer Science., Springer (2010) 56–66
2. Vesely, W.E., Goldberg, F.F., Roberts, N.H., Haasl, D.F.: Fault Tree Handbook. U.S. Nuclear Regulatory Commission, Washington, DC (1981)
3. Schneier, B.: Attack trees. Dr. Dobb's Journal of Software Tools **24**(12) (December 1999) 21–22, 24, 26, 28–29
4. Schumacher, M.: Security Engineering with Patterns - Origins, Theoretical Models, and New Applications. Volume 2754 of Lecture Notes in Computer Science. Springer (2003)
5. Miede, A., Nedyalkov, N., Gottron, C., König, A., Repp, N., Steinmetz, R.: A Generic Metamodel for IT Security. In: ARES, IEEE Computer Society (2010) 430–437
6. Kishor S. Trivedi, Dong Seong Kim, A.R., Medhi, D.: Dependability and Security Models. In: Proceedings of the 7th IEEE International Workshop on the Design of Reliable Communication Networks (DRCN), Washington, DC (October 2009) 11–20
7. Schneier, B.: Secrets & Lies: Digital Security in a Networked World. 1st edn. John Wiley & Sons, Inc., New York, NY, USA (2000)
8. Barbara Kordy and Sjouke Mauw and Saša Radomirović and Patrick Schweitzer: Attack–Defense Trees. Journal of Logic and Computation **24**(1) (2014) 55–87
9. Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J.: Rational Choice of Security Measures Via Multi-parameter Attack Trees. In López, J., ed.: CRITIS. Volume 4347 of Lecture Notes in Computer Science., Springer (2006) 235–248
10. Jürgenson, A., Willemson, J.: Computing Exact Outcomes of Multi-parameter Attack Trees. In Meersman, R., Tari, Z., eds.: OTM Conferences (2). Volume 5332 of Lecture Notes in Computer Science., Springer (2008) 1036–1051
11. Jürgenson, A., Willemson, J.: Serial Model for Attack Tree Computations. In Lee, D., Hong, S., eds.: ICISC. Volume 5984 of Lecture Notes in Computer Science., Springer (2009) 118–128
12. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In Won, D., Kim, S., eds.: ICISC. Volume 3935 of Lecture Notes in Computer Science., Springer (2005) 186–198
13. Buldas, A., Stepanenko, R.: Upper Bounds for Adversaries' Utility in Attack Trees. In Grossklags, J., Walrand, J.C., eds.: GameSec. Volume 7638 of Lecture Notes in Computer Science., Springer (2012) 98–117
14. Buldas, A., Lenin, A.: New Efficient Utility Upper Bounds for the Fully Adaptive Model of Attack Trees. In Das, S.K., Nita-Rotaru, C., Kantarcioglu, M., eds.: GameSec. Volume 8252 of Lecture Notes in Computer Science., Springer (2013) 192–205

15. Kordy, B., Pietre-Cambacedes, L., Schweitzer, P.: DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. CoRR **abs/1303.7397** (2013)
16. Phillips, C., Swiler, L.P.: A Graph-based System for Network-vulnerability Analysis. In: Proceedings of the 1998 Workshop on New Security Paradigms. NSPW '98, New York, NY, USA, ACM (1998) 71–79
17. series, K.Y.E.: Honeynet Project. Know Your Enemy The Tools and Methodologies of the Script Kiddie. http://project.honeynet.org (jul 2000)
18. series, K.Y.E.: Honeynet Project. Know Your Enemy II: Tracking the blackhat's moves. http://project.honeynet.org (jun 2001)
19. series, K.Y.E.: Honeynet Project. Know Your Enemy III: They Gain Root. http://project.honeynet.org (mar 2000)
20. Blomquist, A., Arvola, M.: Personas in action: ethnography in an interaction design team. In: Proceedings of the second Nordic conference on Human-computer interaction. NordiCHI '02, New York, NY, USA, ACM (2002) 197–200
21. Castro, J.W., Acuña, S.T., Juzgado, N.J.: Integrating the Personas Technique into the Requirements Analysis Activity. In Gelbukh, A.F., Adiba, M.E., eds.: ENC, IEEE Computer Society (2008) 104–112
22. Faily, S., Flechais, I.: Barry is not the weakest link: eliciting secure system requirements with personas. In McEwan, T., McKinnon, L., eds.: BCS HCI, ACM (2010) 124–132
23. Faily, S., Flechais, I.: Persona cases: a technique for grounding personas. In Tan, D.S., Amershi, S., Begole, B., Kellogg, W.A., Tungare, M., eds.: CHI, ACM (2011) 2267–2270
24. Pardue, H., Landry, J., Yasinsac, A.: A Risk Assessment Model for Voting Systems using Threat Trees and Monte Carlo Simulation. In: Requirements Engineering for e-Voting Systems (RE-VOTE), 2009 First International Workshop on. (2009) 55–60
25. Sallhammar, K., Knapskog, S.J., Helvik, B.E.: Building a Stochastic Model for Security and Trust Assessment Evaluation. http://q2s.ntnu.no/publications/open/2005/Mass_media/2005_sallhammar_BSM.pdf (oct 2005)
26. Tipton, H., Baker, P.: Official (ISC)2 guide to the CISSP CBK. In: Official (ISC)2 guide to the CISSP CBK. (2010)