Attack Protection Trees [*]

Aliyu Tanko Ali and Damas Gruska

Department of applied informatics, Comenius University, Slovakia.
{aliyu.ali,gruska}@fmph.uniba.sk

Abstract. In this paper, an extended model for attack tree, called attack protection tree, is presented. The traditional attack trees threat model is extended with protection actions on the leaf nodes to protect the intermediate nodes from malicious attack. The proposed formalism allows the protection actions to be defined at the leaf nodes, by doing so, eliminating the chances of attack(s) being successful through OR-refinement. The concepts are illustrated through examples and we use a model checker for attack protection tree analyses.

Keywords: Attack trees, attack-defence trees, attack protection trees

1 Introduction

Security is a difficult topic to discuss and it remains a challenge to both users and developers of modern systems. More than ever before, systems are experiencing threats from both competitors (intellectuals theft), motivated hackers, unintended mistakes done by users and governmental agencies as part of cyber warfare between competing nations. Most of the system attacks are done through existing vulnerabilities. Often, system developers relay on an informal testing process to identify system vulnerabilities. However, a testing technique can only certify that the system meets the business (user) requirements or can only detect the presence of errors, and not their absence. As a result, it is difficult to capture a potential threat to a system using a testing process.

In recent times, the application of formal methods for the development and verification of systems is gaining momentum. Formal methods are a particular kind of mathematically based technique that is used for the specification, development, and verification of complex systems. The systems are model as mathematical entities such that; their properties can be proven and verified. Using formal methods, it is possible to verify the system's properties in a more thorough fashion than empirical testing. There are many diverse formal methods and verification techniques for modeling and checking of security properties, one of which is attack trees (ATs) [1].

Attack trees are formal methods used in modeling varying ways in which a system can be attacked, in a tree structure. They are one of the popular formal

[*] Work supported by the grant VEGA 1/0778/18. Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

security engineering formalism used to reason about multi-step attack scenarios. AT is first introduced by Schneier [1] to provide a formal way of describing the security of a system, they are best known for their graphical representation of attack in a tree structure where the root node of the tree is the target goal of an attacker. Intermediate nodes (sub-trees) are refinements of this goal, and leaf nodes representing the attacker's actions (also called basic actions). An attacker interacts with the system (tree) through the leaf nodes. Schneier defined the refinement of nodes in attack tree to be of two types: AND (conjunctive) and OR (disjunctive). In order to reach the root node, all of its AND children (aggregation) or at least one of its OR children (choice) must be accomplished.

In the literature, different symbols are used to represents attack tree diagrams. For instance, logic gate symbols can be used to represents nodes (goal, sub-goal) instead of the used of conventional circle or rectangle, while in some paper, a rectangle with the gate symbol just below it is used.

ATs are straightforward, and easy to work with, these results to them being adopted and used by non-security professionals in both academia and industry to model many practical case studies such as automated teller machine (ATMs) [2], SCADA communication systems [3], BGP routing protocols [4], to name a few. One of the most important features of attack trees is to provide a simple and non-technical way of understanding security problems in IT systems. However, the overwhelming growth of vulnerabilities in modern systems is increasing spontaneously. As well as ATs lack the expressing power to represents the order in which attacks are executed (sequential, parallel), attack deference strategy, etc. Furthermore, there is little work done in both the industrial and academic literature on extending the use of attack trees beyond depicting ways in which threats can be identified.

Related work. There are several discussions about general issues and challenges in the use of attack trees for threat modeling. The basic formalism of ATs does not include any defence mechanisms. It can only statically shows varying ways in which a system can be attacked. Ever since the introduction of ATs, a wide range of research areas has emerged from them, this includes; threat modeling, extending refinement with introduced new gates to cover different attack scenarios, using attack trees to analyze probabilistic attack to systems, to name a few. We will look at some of the work done in these areas.

Threat modeling. One of the common practices for assessing risk in businesses or systems is through the use of risk management techniques, and there exist several processes for identifying and prioritizing risks. The concept of threat modeling is one of those processes. Traditionally, it involved the use of theoretical and mathematical concepts. Several research and projects adopted the use of attack trees to this end. For example, the work in [2] used attack tree to identify possible attacks to automated teller machine (ATMs), SCADA communication systems attacks using attack trees was presented in [3], BGP routing protocols [4] and medical implant in [5].

Extending attack trees. Several attack tree extensions have been proposed, amongst which are attack-defence trees (ADT) [2], attack response tree (ART) and attack countermeasures tree (ACT) [4]. ADT is an extension of attack trees that include, not only the attackers' actions but also the defenders' actions and model their interplay in the form of game theory portraying the attacker-defenders repelling each other move [6, 2, 7]. ACT, on the other hand, extends the defence tree with countermeasures to mitigate attackers' actions. Attack response tree (ART) formalism extends the attacker-defender game to find an optimal policy from a list of countermeasures. ART is not popular as they suffer a state-space explosion problem. The work in [9] used defence tree for the evaluation of Information Technology (IT) security investments, sequential conjunction SAND gates were introduced in [10] to model attacks that are executed in an orderly manner.

Quantitative and qualitative analysis. An excellent survey on attack trees and related formalism was presented by Kordy et al. [8]. The work in [11] model the attackers' behaviour by introducing temporal order to the attackers' decision-making process. In another effort, attack net for penetration texting was introduced in [12]. A time-dependent technique to analyze attack time was presented in [13]. A large number of AT analysis and verification frameworks has been developed, model-driven engineering approach [14],timed automata [15], Petri nets [17] e.t.c. There have been quite a several tools developed for verifying different attack scenarios. ADTool is free, open-source software assisting graphical modeling and quantitative analysis of security properties using attackdefence trees, others include SeaMonster[18] and, ATSyRa [19]. The work in [8] introduced SPTool, an equivalence model checker for SAND ATs and attack trees in Isabel is presented in [16].

Aim. This paper aims to go beyond attack trees, identify varying ways in which a system can be attacked and also defining a set of protection actions at the point at which the attacker interacts with the system. We plan to achieve this in two ways, firstly we will adopt the concept of attack trees and introduce protection actions to counter basic attack steps. Secondly, we will use a model checker to model our approach, verifying whether the designed protection actions are satisfied in the model and also if the attackers' actions can lead to an undesirable state. The results obtained in this work will serve as preliminary work that will further investigate towards dynamic attackers' actions. We also plan to investigate the notion of time, the flow of information within the system. This is particularly interesting because attack trees only represent static varying ways in which a system can be attacked while the attackers' actions are dynamic. Also, there are different ways in which information systems can be attacked which is almost impossible to be captured using attack trees; for example side-channel attacks.

This paper is organized as follows, section 1 provides an introduction and related work on the use of attack trees for threat modeling. Section 2 provides a review of the concepts of attack tree, we defined attack protection tree in section 3. Section 4 analyses attack protection tree using UPPAAL model checker and end the section with the conclusion and future work direction.

2 Attack Trees

In this section, we provide a gentle introduction to attack trees and define the working formalism of attack protection tree. Attack Tree (AT) is a methodological way of describing the security of a system on different varying attacks with a clear path description of possible ways to refute them. As the name implies, it is represented in the form of a tree (read bottom-up), with the root node as the attack target, children nodes (sub-goals) and leaf. The goal of an attacker is to reach to the root node, children nodes are the refinement of the root node. For any successful attack, the conditions (OR, AND) on the children nodes have to be satisfied.

The underlying concept of attack tree threat modeling is similar to that of popular divide and conquer technique [20] where a problem is recursively broken down into smaller problems that are theoretically simpler to reason about and solve. Although, unlike divide and conquer technique, AT consist of two types of nodes:

- **Event nodes:** This is made up of basic attack steps (BASs), intermediate events and top event. The BASs model malicious actions carried out by the attacker and they are modelled as the leaves of AT. The intermediate events are caused as the results of the achievement of BASs and are referred to as internal nodes, sub-goals or child nodes and finally, the top event is modelled as the root of the tree.
- **Gate nodes:** The gates combined the BASs, representing how and in which temporal order the successful execution of BASs results in the compromise of node above (parent node). The OR and AND are two of the most popular gates used to modelled simple attacks. OR gates represent different ways and the AND to represent necessary steps toward achieving (sub)goal. For a successful attack to be carried out through an AND-gate, the attacker has to succeed in all of its child nodes, while through the OR-gate requires the attacker to execute at least one child node successfully. Other gate extensions such as SOR and SAND were introduced to model the sequential attack process.

Besides presenting the visual attack scenarios to a system, attack trees can also be used as formal models with assigned semantics that can be used to answer several quantitative and qualitative questions such as; "What is the probability of reaching to the top node? [21], What is the cheapest attack cost? [11], Which attacks do not require special equipment to execute an attack? [22, 21] e.t.c.

The work in [2] introduced attack-defence trees to analyze threats and provide countermeasures to attacks on ATMs, the work extended attack trees by adding countermeasure nodes and termed it as defence trees. The defence methods only have OR relationships with the leaf(s) it is said to defend, meaning once the countermeasures failed as a result of introducing another attack vector, the defence mechanism will fail as well. For example, in the black box attack, a sensor to detect ATM modules deactivation can be spoofed and that can result in a failed defence mechanism.

3 Attack Protection Tree (APT)

In this section, we define the working formalism of attack protection trees. Firstly, we introduce some notations that will be used to formalise the attack protection tree definition.

3.1 APT formalism

APT is an extension of attack trees with protection actions. As stated in the previous section, the root node is the attackers' ultimate goal, intermediate nodes represent sub-goals necessary to achieve the root node. We exclusively define leaf nodes as the point of interaction between an attacker and the tree. An attacker can only influence the tree by interacting with the leaves. All other nodes are derived from attacker actions, meaning, only by successfully achieving the leaf node(s) does the attacker move to the next node (intermediate). Hence, an attacker cannot penetrate the attack tree at intermediate nodes. We formalise APT as follows.

An attacker denoted by A, is an outsider that interact with the system to per-take in malicious activities. A defender denoted by $\mathtt{D}$, could be either human or automated designed to counter the attackers' actions. We represent the set of actions which can be performed by an attacker as $\mathcal{B}$. While the set of defenders (protection) actions as $\mathcal{P}$. We denoted by $L = \{s_1, ..., s_n\}$ the set of leaves in the tree. The attackers' actions are defined on the leaf nodes as a function $\mathrm{A} : L \to 2^{\mathcal{B}}$, and protection actions are associated with the leaves to block the attackers' action $\mathtt{D} : L \to 2^{\mathcal{P}}$. This interprets that an attacker try to perform on a given leaf node action $b \in \mathrm{A}(s)$, and that there is a protection action assigned by the defender $\alpha \in \mathtt{D}(s)$ that will block the malicious activity. For simplicity, we denoted this as $s^{\alpha/b}$. Whereas for a leaf s, if each $b \in \mathrm{A}(s)$ is associated with protection action $\alpha \in \mathtt{D}(s)$ then the leaf is said to be protected, denoted as $s \vdash (\mathrm{A}, \mathtt{D})$. For instance, protection action $\alpha \in \mathcal{P}$ to `password attack` (attackers' action b) could be `multi-factor authentication`. The general syntax of APT is generated by the following.

$$\mathcal{T} = op(S_\circ(t(S_1), ..., t(S_N))) \tag{1}$$

Where T is a tree, $op = \{AND, OR\}$, is the gates refinement, each node which is not a leaf is either AND or OR node. $S_\circ$ is the root of the tree (goal) and $t(S_i)$ is a subtree/ intermediate (I) with the root S_i. Based on this syntax, we define attack protection trees as follows;

Definition 1. *An attack protection tree is a tuple $\mathcal{T} = (S^*, E, \mathcal{B}, \mathcal{P}, \mathrm{A}, \mathtt{D})$. Where $S^* = \{S_\circ\} \cup \{\mathrm{I}\} \cup \{L\}$ is the set of nodes, E is the set of edges that link two nodes in the tree, $\mathcal{B}, \mathcal{P}$ are the disjoint actions of the attacker and the defender. $\mathrm{A}, \mathtt{D}$ representing the attacker $\mathrm{A} : L \rightarrow 2^{\mathcal{B}}$ and defender $\mathtt{D} : L \rightarrow 2^{\mathcal{P}}$ respectively.*

Lets assume $S\{s_1, ...s_n\}$ is an intermediate node with set of leaves. The intermediate node S is said to be secure if $\forall s_i, \forall b \in \mathrm{A}(s_i).\{\exists \alpha \in \mathtt{D}(s_i) : s_i^{\alpha/b}\}$ (i.e. for each s_i we have $s_i \vdash (\mathrm{A}, \mathtt{D})$). This is being denoted as $S \vdash (\mathrm{A}, \mathtt{D})$.

Another instance of malicious activity `password attack` can be viewed as the goal of an attacker A. To achieve this, some basic actions $b \in \mathcal{B}$ has to be carried out such as `brute force`. We can define more protection action to defend against this attack. Example, the protection actions $\alpha \in \mathcal{P}$ for `brute force` could be restricting the number of times the attacker can make an attempt or using a delay in between the actions and additional `password masking` as a protection action to the password attack.

Example 1. Consider Fig 1 were a tree with two child nodes and three leaf nodes is depicted. we presented in equation (2) the corresponding attack protection tree construction that can captured both the attacker actions and the protection actions.

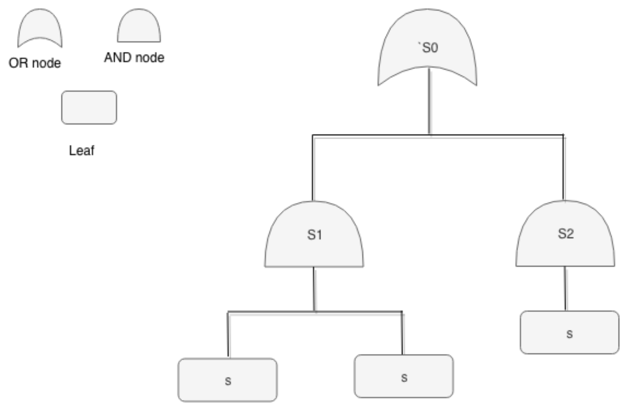

Fig. 1. APT example.

$$S_\circ(S_1(s^{\alpha/b}, s^{\alpha/b}), (S_2(s^{\alpha/b})). \qquad (2)$$

Assuming that gates OR, AND (denoted by $\vee, \wedge$) construction are included in the nodes, the semantics representation of the tree in equation (2) is as follows;

$$\begin{aligned}
S_{\circ\vee}(S_1(s^{\alpha/b}, s^{\alpha/b})), (S_2(s^{\alpha/b})) &= S_\circ\{(S_1)/\mathcal{P} \vee (S_2)/\mathcal{P}\},\\
S_{\circ\wedge}(S_1(s^{\alpha/b}, s^{\alpha/b})), (S_2(s^{\alpha/b})) &= S_\circ\{(S_1)/\mathcal{P} \wedge (S_2)/\mathcal{P}\}.
\end{aligned} \tag{3}$$

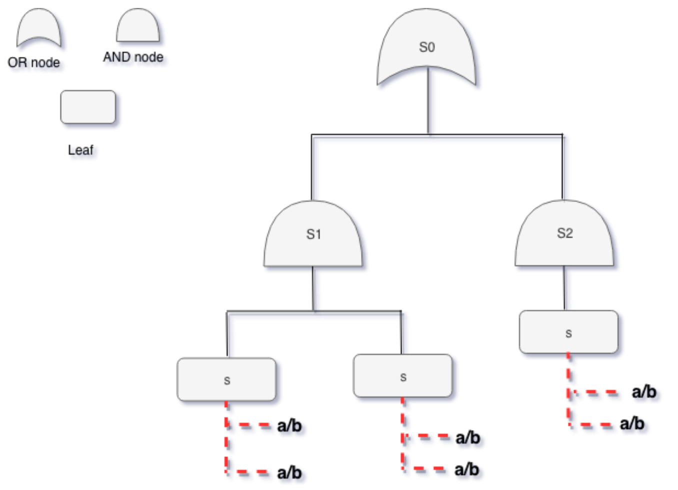

Fig. 2. APT with actions.

Note that, this is different from inserting a defence node on an attack path as each intermediate (sub-tree) node S_1, S_2 in the tree will be protected by protection action(s) at the leaf nodes, as shown in Fig 2.

Example 2. To demonstrate further example using this approach, we look at an IoT device compromise (see Fig. 3), whose attack tree was presented in [14].

We work with the same set(s) of goal, sub-goals and leafs presented in the paper and, introduced sets of basic attackers actions and protection actions which can be seen in Table 1.

The goal ($S_\circ$) of the attack is to compromise an IoT device, the intermediate nodes (attack refinement) includes; access home network as (S_1), gain access to a private network (S_2), access LAN (S_3) and access WAN (S_4). The leaf nodes are; exploit software vulnerabilities (s_1), run malicious scripts (s_2), get credentials (s_3), find LAN access point (s_4), spoof MAC address (s_5), find WAN (s_6), and break WPA keys (s_7). Attackers' point of interaction with the tree is at the leaf(s) nodes and therefore, protection at the attackers' point of interaction with the tree (leaves) will be reduced drastically the possibility of the system being

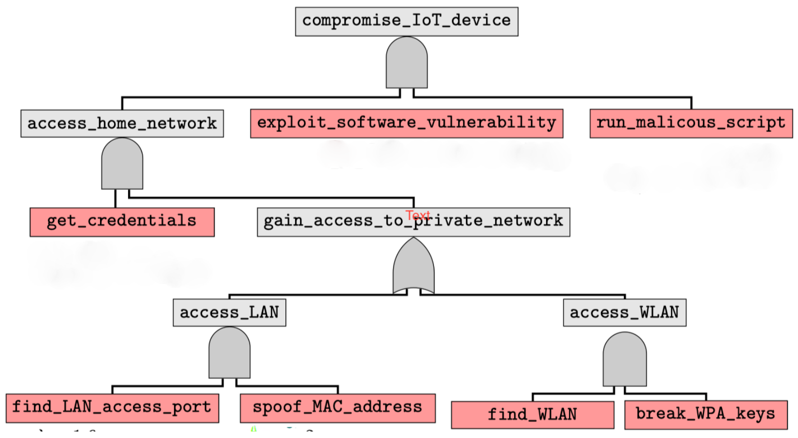

Fig. 3. IOT attack tree

basic action (b)	**protection action** (α)
b_1. use tools and techniques weakness	α_1. Access restriction, IPSs to block exploit behaviors
b_2. email attachments	α_2. Watch out for malicious or compromised websites
b_3. crack password	α_3. avoid simple passwords that are easy to guess
b_4. install rogue access point	α_4. wireless intrusion detection systems (IDS)
b_5. explore weak authentication	α_5. Use encrypted protocols
b_6.rogue access point	α_6. wireless intrusion detection systems (IDS)
b_7. deauth attack	α_7. use a non-standard channel and hid it

Table 1. List of basic, protection actions

compromised. As it is often the case in security analysis, it is difficult to obtain precise data (all basic action and protection action), and our guesses are not intended to reflect reality, but rather to illustrate our formalism.

$$S_{\circ\wedge}(S_{1\wedge}(s^{\alpha_1/b_1}, S_{2\vee}(S_{3\wedge}(s^{\alpha_2/b_2}, s^{\alpha_3/b_3}), S_{4\wedge}(s^{\alpha_4/b_4}, s^{\alpha_5/b_5}))), s^{\alpha_6/b_6}, s^{\alpha_7/b_7}). \tag{4}$$

Securing the leaf node (in our belief) provide better protection rather than a single defence node to protects the intermediate node, by applying protection actions at each leaf node, the risk of an attack occurring through the OR refinement will be eliminated.

4 Analysing Attack Protection Trees via Automata

Formal methods play an essential role in developing safety-critical systems. At the heart of formal methods, (formal) specification and verification are used to model and analyse security, safety, and system behaviour. Model-checking has proven to be a successful technique for requirements verification, design, and analysis for various real-time systems [23]. Although at this point, we did not consider attributes for attack protection tree, we make use of UPPAAL [24], a model checker, to analyze attack protection trees. One of the strengths of UPPAAL is; the ability to model systems with quantitative attributes (time, cost. e.t.c).

We provide a compositional semantics of APT in terms of finite-state automata (FSM). We translate the entire APT element into an FSM and then analyse the APTs by formulating security and safety properties of interest as queries in CTL logic [25].

As an example, consider a modern critical infrastructure where different independent systems made up of a larger critical system that is essential for the functioning of a society and economy [26]. Sometimes, these systems can be composed of different inter-dependent components (systems) to provide better performance. For example, agriculture and water supply; modern irrigation systems (agriculture) heavily depend on the use of water (water supply) from sources such as a local reservoir or dams. However, cyber-attacks are considered one of the major threats and most challenging problems to such critical infrastructures. To model and analyze this system using attack protection tree, we partition it into 3 three; the infrastructure model, the attacker model, and the protection model.

4.1 Infrastructure model

The infrastructure model provides an abstract interaction view of the behaviour of different components in the system. Using UPPAAL, we model these components as locations, create channels that synchronise $\mathcal{B} \in \mathrm{A}$ to indicates the desire of the attacker to influence the locations. In Fig 4, we presented the infrastructure model in UPPAAL, we created 4 channels (`attck, succ, unsucc, attack_failed`) and use them to trace the attackers actions. The attck channel is for initiating the action, succ channel synchronises successful actions of the attacker while the failed actions are passes through unsucc channel. The `attack_failed` channel returns the attacker to the initial location. The locations in Fig. 4 for the infrastructure model are; $S0, S1, S2, S3$. $S0$ indicates the initial state where the attacker execute actions, when the actions are successful, In the case of protection action $\alpha \in \mathtt{D}(s)$ is true on $b \in \mathrm{A}(s)$, the attacker will be left with no option since the transition will be to failed location. $S1$ `and` $S2$ are intermediate locations which could be reach depending the actions of the attacker, while $S3$ is represent the target goal.

4.2 Attacker model

Since the attacker has no strong influence on the system without $b \in \mathrm{A}(s)$ being successful, successful execution of the action will results in a transition to the next location while unsuccessful will force the system to failed location. The

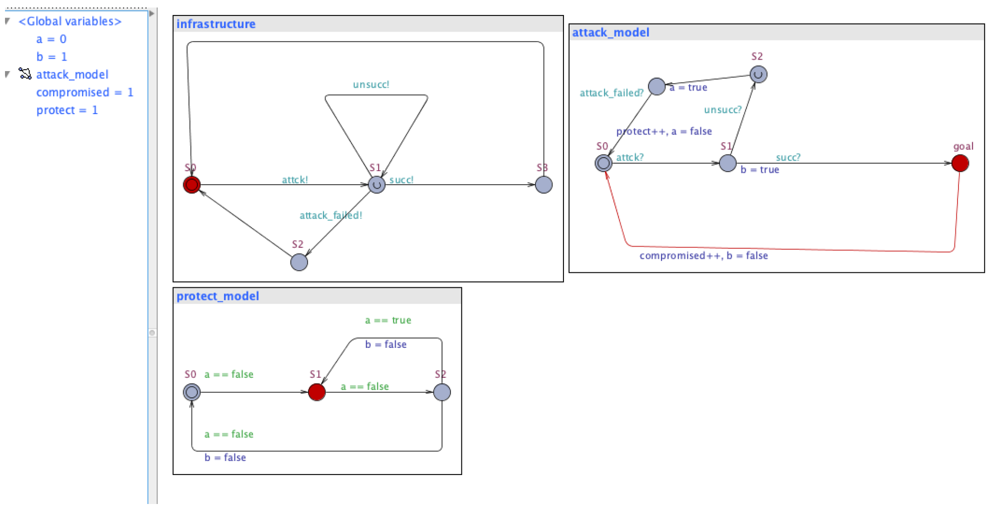

Fig. 4. UPPAAL analysis of APT model

attacker model in Fig. 4 models the behaviour of the received broadcast from the infrastructure model.

4.3 Protection model

As for the protection model, the main aim is to have a protection action that will counter the basic action of the attacker. We set this as a guard condition, depicted as `protect_model` in one of the sections in Fig 4. In the Figure, the expression $a == false$ at location $S0 \rightarrow S1$ restrict the transition of $attck$ while be possible only then the variable a is false while the assignment $b = false$, is an update whenever the basic action is failed in the `attack_failed`.

5 Conclusion and future work

This paper provides a gentle introduction to attack protection tree, we formalise the definition and apply the concept in some examples. As for this work, we did not consider scenarios concerning a given parameter (quantification) to analyze APT. As earlier mentioned, this work serves as preliminary for the future work ahead. We plan to extend APT to analyze case studies incorporating realistic values, dynamic attackers' actions. Furthermore, we plan to introduce time information in the future.

This is particularly interesting because attack trees only represent static varying ways in which a system can be attacked. Applying realistic values will allow us to analyze properties such as time, cost, difficulty, etc. At the same time, since the attackers' actions are dynamic, we will investigate the timing of actions in the system. This will lead our work to study information flow properties as a new way in which a system can be attacked, we will model the system with scenarios whereby some attacks/ protection could be available only in some given time. Hence we plan to associate with functions $\mathtt{A}, \mathtt{D}$ other parameters such as time, cost, e.t.c. For example, by $\mathtt{A}(s, t)$ and $\mathtt{D}(s, t)$ we will express attacker's and defender's actions at time t, respectively.

References

1. Bruce Schneier. Attack trees. *Dr. Dobbs journal*, 24(12):21–29, 1999.
2. Marlon Fraile, Margaret Ford, Olga Gadyatskaya, Rajesh Kumar, Mariëlle Stoelinga, and Rolando Trujillo-Rasua. Using attack-defense trees to analyze threats and countermeasures in an atm: a case study. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 326–334. Springer, 2016.
3. Chee-Wooi Ten, Chen-Ching Liu, and Manimaran Govindarasu. Vulnerability assessment of cybersecurity for scada systems using attack trees. In *2007 IEEE Power Engineering Society General Meeting*, pages 1–8. IEEE, 2007.
4. Arpan Roy, Dong Seong Kim, and Kishor S Trivedi. Attack countermeasure trees (act): towards unifying the constructs of attack and fense trees. *Security and Communication Networks*, 5(8):929–943, 2012.
5. Muhammad Ali Siddiqi, Robert M Seepers, Mohammad Hamad, Vassilis Prevelakis, and Christos Strydis. Attack-tree-based threat modeling of medical implants. In *PROOFS@ CHES*, pages 32–49, 2018.
6. Zaruhi Aslanyan and Flemming Nielson. Pareto efficient solutions of attack-defence trees. In *International Conference on Principles of Security and Trust*, pages 95–114. Springer, 2015.
7. Stefano Bistarelli, Marco DallAglio, and Pamela Peretti. Strategic games on defense trees. In *International Workshop on Formal Aspects in Security and Trust*, pages 1–15. Springer, 2006.
8. Barbara Kordy, Piotr Kordy, and Yoann van den Boom. Sptool–equivalence checker for SAND attack trees. In *International Conference on Risks and Security of Internet and Systems*, pages 105–113. Springer, 2016.
9. Stefano Bistarelli, Fabio Fioravanti, and Pamela Peretti. Defense trees for economic evaluation of security investments. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8–pp. IEEE, 2006.
10. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In *IFIP International Information Security and Privacy Conference*, pages 339–353. Springer, 2015.
11. Aivo Jürgenson and Jan Willemson. Serial model for attack tree computations. In *International Conference on Information Security and Cryptology*, pages 118–128. Springer, 2009.
12. James P McDermott. Attack net penetration testing. In *NSPW*, pages 15–21, 2000.
13. Florian Arnold, Holger Hermanns, Reza Pulungan, and Mariëlle Stoelinga. Time-dependent analysis of attacks. In *International Conference on Principles of Security and Trust*, pages 285–305. Springer, 2014.
14. Rajesh Kumar, Stefano Schivo, Enno Ruijters, Bura Mehmet Yildiz, David Huistra, Jacco Brandt, Arend Rensink, and Mariëlle Stoelinga. Effective analysis of attack trees: A model-driven approach. In *International Conference on Fundamental Approaches to Software Engineering*, pages 56–73. Springer, Cham, 2018.

15. Olga Gadyatskaya, René Rydhof Hansen, Kim Guldstrand Larsen, Axel Legay, Mads Chr Olesen, and Danny Bøgsted Poulsen. Modelling attack-defense trees using timed automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 35–50. Springer, 2016.
16. Florian Kammüller. Attack trees in isabelle. In *International Conference on Information and Communications Security*, pages 611–628. Springer, 2018.
17. GC Dalton, Robert F Mills, John M Colombi, Richard A Raines, et al. Analyzing attack trees using generalized stochastic petri nets. In *Information Assurance Workshop*, pages 116–123, 2006.
18. Per Håkon Meland, Daniele G Spampinato, Eilev Hagen, Egil Trygve Baadshaug, Krister-Mikael Krister, and Ketil Sandanger Velle. Seamonster: Providing tool support for security modeling. *Norsk informasjonssikkerhetskonferanse, NISK*, 2008.
19. Sophie Pinchinat, Mathieu Acher, and Didier Vojtisek. Atsyra: an integrated environment for synthesizing attack trees. In *International Workshop on Graphical Models for Security*, pages 97–101. Springer, 2015.
20. Xiaochun Wang, Xiali Wang, and D Mitchell Wilkes. A divide-and-conquer approach for minimum spanning tree-based clustering. *IEEE Transactions on Knowledge and Data Engineering*, 21(7):945–958, 2009.
21. Vineet Saini, Qiang Duan, and Vamsi Paruchuri. Threat modeling using attack trees. *Journal of Computing Sciences in Colleges*, 23(4):124–131, 2008.
22. Arpan Roy, Dong Seong Kim, and Kishor S Trivedi. Cyber security analysis using attack countermeasure trees. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, page 28. ACM, 2010.
23. Anders Hessel, Kim G Larsen, Marius Mikucionis, Brian Nielsen, Paul Pettersson, and Arne Skou. Testing real-time systems using uppaal. In *Formal methods and testing*, pages 77–117. Springer, 2008.
24. Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. 2006.
25. François Laroussinie and Ph Schnoebelen. Specification in ctl+ past for verification in ctl. *Information and Computation*, 156(1-2):236–263, 2000.
26. John Moteff, Claudia Copeland, and John Fischer. Critical infrastructures: What makes an infrastructure critical? Library of Congress Washington DC Congressional Research Service, 2003.