Reactive Defences in Attack Defence Trees

Pratik Rajesh Borikar, Naman Chokhani
Birla Institute of Technology and Science Pilani, Pilani Campus
Email: f20170550@pilani.bits-pilani.ac.in, f20170726@pilani.bits-pilani.ac.in

***Abstract*—We introduce and give formal representation and semantics of a new component, the Reactive Defence box (RD box). This is a relatively simple, yet power modification to the traditional Attack Defence Trees as it incorporates reactive defences into the pre-existing model. We present the usage of this new component with an example along with the semantics for the same.**

I. Introduction

Security of any system is in this rapidly progressing world is of utmost concern. It is a well-established fact that security parameters and measures of a sufficiently valuable system is not static. In order to keep a system secure, it has to be defended against a growing number of attacks [1]. But, as defence measures get employed , more complicated attacks develop to counter these defences leading to increasingly complex systems.

Attack trees along with its variants provide quick and effective ways to express attack scenarios in a much formal and easy to understand ways. Whether it is for representing vulnerabilities of various voting schemes, analyzing security of critical infrastructures in the electric sector , classifying ATM-related frauds , or quantifying cost, difficulty, and time of attacks against an RFID-based goods management system , attack trees and their derivatives have been successfully adopted by the industrial sector as a means of modelling and evaluation of security [2]. Attack Defence Trees(ADTrees) have been successful in incorporating defences in an attack tree which prevent an attacker from attacking a particular node. The major limitation of defences in ADTrees is that . They do not provide any path to be followed if an attack has been successful by disabling the defence.

This paper attempts to introduce a new component in ADTrees called a Reactive Defence box which provides reactive counter-measures in response to an attack. The semantics and the representation of this component along with an example is provided in the paper.

II. Related Work

Attack defence trees were introduced by Weiss [3] and by Amoroso [4]. The dynamic nature of attack defence tree is broadly discussed in [5] wherein semantics using statistical markov chains is also added.The formal methods in the paper are also inspired by those mentioned in [2]. The sequential time dependent gates used in the fault trees are also widely mentioned in the [6]. Several papers in the domain of fault tree analysis discuss the reactive nature in terms of repairing a component. Maintenance and repairability of a tree in a dynamic sense is broadly discussed in a thesis by Ruitjers [7] . The Reactive Defence box is inspired by repair boxes in Fault Maintenance trees mentioned above.The basics of countermeasures in an attack tree are discussed in a paper by Kumar, R. in [8].Attack countermeasures trees are presented in the paper by A. Roy in the following paper [9]

III. Basic Definition of Attack(-defence) trees

Attack trees provide an opportunity to represent an attack scenario hierarchically. Formally, they are rooted trees with labelled nodes [2]. The goals of the attacker are represented by the labels of the node, whereas label of the root node is the final/main goal of the attacker. The nodes may also be called as events in some cases. The root node could also be referred to as asset or a top-event. If achieving a particular goal requires from the attacker to achieve some other sub-goals, then the node labelled with that goal is called refined [2]. The rudimentary model of attack trees has two types of refinements: OR and AND. The goal of an OR refinement is achieved when any of its sub-goals/children is achieved, whereas the goal of an AND refinement is achieved when sub-goals of all its children are achieved.

An ADTree(Attack Defence Tree) is a more expressive and enhanced version of attack trees defined above. An ADTree is a node-labelled rooted tree describing the measures an attacker might take in order to attack a system and the defences that a defender can employ to protect the system [1]. Consequently, there are two types of node in an ADTree: attack nodes and defence nodes. The two key features of ADTrees are the representations of refinements and counter-measures. A goal having one or more children of the same type is called a refined node. If the node does not have any children of the same type it is called non-refined node or a basic attack step (BAS).

Every node can have one child of the opposite type, representing a counter-measure. Thus, a defence node may be refined into may sub-nodes which refine the defence and may have one attack node as a countermeasure and vice-versa for an attack node.

Example of an attack tree and ADTree is given in figure 1. In order to achieve the main goal in Fig. 1a, the attacker needs to perform actions a and b, and anyone of the actions c, d or e. But, the same actions won't work with the ADTree in Fig. 1b because execution of c may be stopped by defender by performing actions i, j and k. Furthermore, the attacker will have to counter defence f, by performing either g or h. We will further continue our discussion by taking the example of

the attack-tree shown in Fig 2, which represent the attack tree of a house break-in.

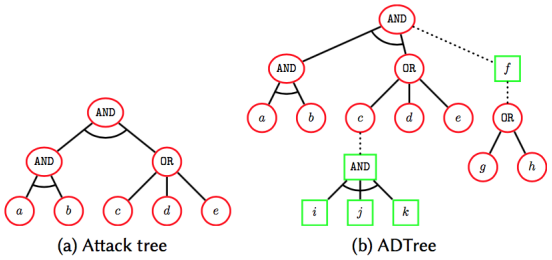


Fig. 1. An attack tree and an Attack Defence Tree

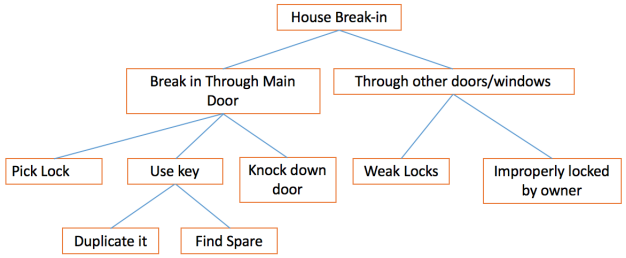


Fig. 2. Attack tree for house break-in

IV. Gates in an ADTree

Gates in an ADTree Basic attack step are the leaves of the AD trees on which attack can occur spontaneously. Nodes of the trees could also be called as an event, as mentioned in the previous section, an attack or defence node can be refined into sub-nodes using an AND or OR refinement. Gates can be used to represent these refinements and even more complicated ones. Gates represent how attacks propagate through the system and how successful attacks on leaves or basic-events can combine to lead to an intermediate event and eventually to the top event. Gates take as an input either leaves or outputs from gates in their sub-trees. Some of the gates are:
1. AND gate: Output event occurs if attack is successful on all of its children.
2. OR gate: Output event occurs if attack is successful on any one of its children.
3. SAND gate: Children of a SAND gate are activated sequentially from left to right.
After the success (attack) of the leftmost child, the second left most child is activated, and so on until the disruption of rightmost child. If all children are successfully attacked, the SAND gate is disrupted. However, if any child fails (to be disrupted), the SAND gate directly fails.
4. PAND gate: PAND gate is similar to SAND gate but the only difference lies in the fact that all children of PAND gate are activated initially.
The representation of these gates are given in Fig 3.

Fig. 3. From left to right: AND, SAND, PAND, OR gates

Figure 4 shows us the semantics of AND, OR and PAND gates.

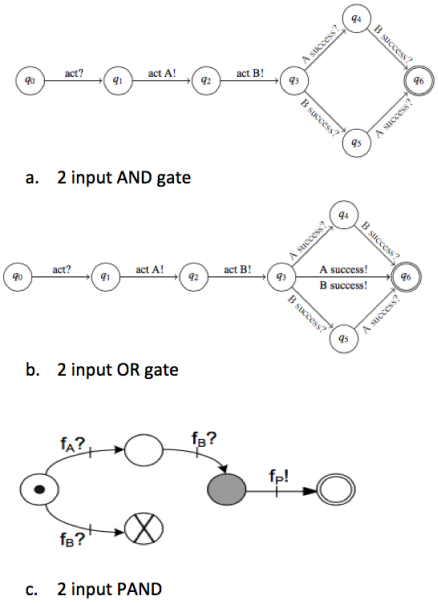


Fig. 4. Semantics of AND,OR and PAND gates

V. Proposed Work

We plan to introduce a new component in attack-defence trees which would make defences capable of reacting to a situation/attack more dynamically rather than the traditional static defences.
Traditionally, defences in attack-defence trees where implemented as a preventive measure whose main objective was to make the attack on the node they were defending harder. So in order for an attack to be successful, the attacker had to disable the defence node. However, these defences do not provide any way to react in situations after the attack has occurred. Our main aim is to introduce a component which takes action after a successful attack and employs a suitable defence.

VI. The Reactive Defence(RD) Box

The reactive defence box is a new component introduced in attack-defence trees with the aim of making defences dynamic in nature. Fig 5 shows the basic structure of the RD box. The listening lines are connected to basic attack steps or to intermediate nodes which constantly determine the the condition of nodes according to the policy. The policy is

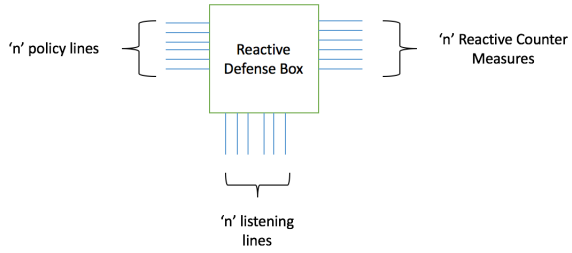


Fig. 5. RD box

basically a condition which is evaluated to determine where the attack on node in an ADTree is successful or not. Thus, the policy and listening lines work together to determine whether to invoke the reactive countermeasure or not. The reactive counter measure can be either used to alert the defender of the attack or may be of corrective nature. When the reactive counter-measure is of corrective type, it returns the event associated with it to its initial state of not being attacked. When the nature of the reactive counter-measure is that of alerting, it notifies the defender that a successful attack has taken place.

For each listening line (right to left) there is a corresponding policy (top to bottom) which is being evaluated. If the policy fails at any point of time, the respective counter-measure is invoked (top to bottom). Each listening line has 1 policy and 1 reactive counter-measure. The value 'n' starts from 1 to the maximum number of nodes in an attack tree. The attack-tree with house break-in example is now illustrated with a RD box in Fig 6.

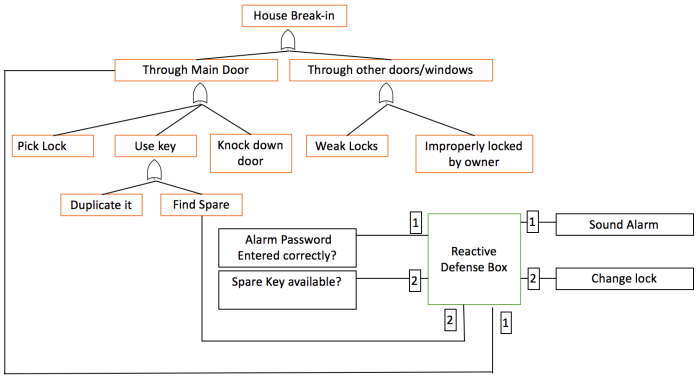


Fig. 6. Attack tree with RD box

From the figure 6, it can be seen that the RD box has the value of n = 2. The first listening line is connected to an intermediate node, when an intruder breaks-in through the main door. The door is equipped with an alarm system that would be disabled if a password is entered correctly. The policy checks whether the password is entered correctly when the main door opens. If the policy fails, that is, if the wrong or no password is entered the reactive counter measure corresponding to this failure of this policy, which is the sounding of the alarm. This type of counter-measure alerts the concerned user about the attack.

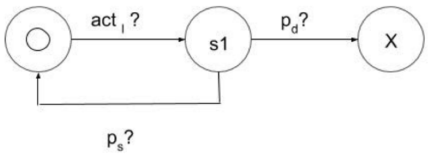
The second listening line has been connected with a BAS. The attacker can find a spare key to enter the house. The policy associated with this listening line checks whether the spare key is available with the home-owner or not. If the spare key is not available, it is considered to be stolen by the attacker. In this case, the reactive counter-measure is activated and the lock of the door is changed. Such type of counter-measure is corrective in nature.

VII. Semantics of RD box

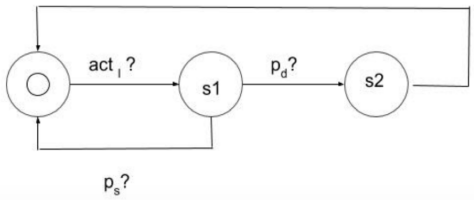
Figure 7 shows the semantics for RD box. It is to be noted that the semantics shown here are for the case of n = 1. For values of n greater than 1, we simply define semantics for each line individually, that is, we view n-line RD box as a combination of n 1-line RD boxes.

From the example given in the text, there are two types of counter measures given. One of them is alerting in nature, and the second one is corrective in nature. The main difference is that a corrective counter-measure would return to the initial stage after employing the reactive defence whereas an alerting countermeasure would just perform the job of alerting/notifying a system or a user and needs to be reset manually.

The reactive counter measure waits for an activation signal



a. Semantics for reactive counter-measure of alerting type



b. Semantics for reactive counter-measure of correcting type

Fig. 7. Semantics of the RD box

(act_i) from the listening node. As soon as the node is successfully activated (attacked as per the given example), an activation signal is sent to the defence box, taking it to the state s1.

Now, at state s1, a predefined policy is checked for. Here, p_s signifies that the policy is satisfied whereas p_d stands true if the policy is dissatisfied. If the policy is satisfied, no action is taken by the reactive counter measure box and it returns to the initial state.

For an alerting type countermeasure as shown in the fig 7a, if the policy is not satisfied, a predefined countermeasure is activated while it stops at a new final state. On the other hand,

for a corrective countermeasure (as shown in fig 7b), the initial state is restored after the defence has been employed.

VIII. Conclusion and future proposals

The RD box certainly paves way to a much more flexible and dynamic Attack Defence tree formalization. This however, has a wide scope of improvement and enhancement in the near future. For example, in a corrective type reactive defence, a timed parameter can be introduced which specifies the time gap between transitioning from state s2 to its' initial state. The reactive defences could also be coupled with AND, OR or other gates to enhance the dynamic nature of the proposed RD box.

References

[1] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer, "Foundations of attack–defense trees," in *International Workshop on Formal Aspects in Security and Trust*. Springer, 2010, pp. 80–95.

[2] W. Wideł, M. Audinot, B. Fila, and S. Pinchinat, "Beyond 2014: Formal methods for attack tree–based security modeling," *ACM Computing Surveys*, vol. 52, pp. 1–36, 08 2019.

[3] J. D. Weiss, "A system security engineering process," in *Proceedings of the 14th National Computer Security Conference*, vol. 249, 1991, pp. 572–581.

[4] E. G. Amoroso, *Fundamentals of computer security technology*. Prentice-Hall, Inc., 1994.

[5] H. Boudali, P. Crouzen, and M. Stoelinga, "Dynamic fault tree analysis using input/output interactive markov chains," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. IEEE, 2007, pp. 708–717.

[6] S. Junges, D. Guck, J.-P. Katoen, and M. Stoelinga, "Uncovering dynamic fault trees," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 299–310.

[7] E. J. J. Ruijters, "Zen and the art of railway maintenance: Analysis and optimization of maintenance via fault trees and statistical model checking," 2018.

[8] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, and R. 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*. Springer, 2016, pp. 326–334.

[9] A. Roy, D. S. Kim, and K. S. Trivedi, "Attack countermeasure trees (act): towards unifying the constructs of attack and defense trees," *Security and Communication Networks*, vol. 5, no. 8, pp. 929–943, 2012.