








On Validating Attack Trees with Attack Effects

Hideaki Nishihara¹, Yasuyuki Kawanishi^{1,2}, Daisuke Souma^{1,2},
and Hirotaka Yoshida¹

¹ SEI-AIST Cyber Security Cooperative Research Laboratory, Cyber Physical Security Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Osaka, Japan

{h.nishihara,hirotaka.yoshida}@aist.go.jp

² Cyber-Security R&D Office, Sumitomo Electric Industries, Ltd., Osaka, Japan
{kawanishi-yasuyuki,souma-daisuke}@sei.co.jp

Abstract. Threats or attacks can be decomposed into more primitive attacks/events by attack trees. These trees can show possible scenarios of threats. In addition, the quantitative properties of attacks, called attributes, can be integrated along with the tree structures. This paper introduces a formal system for attack trees focusing on refinement scenarios, and enriches attack trees with effects of attacks, which allows the evaluation of the validity of attack decomposition systematically. The property that sub-attacks refine an attack is described by the relationship among their effects, that is defined as consistency of a branch. Consistent attack trees support a systematic approach for the entire attack tree process. Furthermore the effects of attacks in consistent attack trees are well-behaved as an attribute. These ideas are applied to the case study of a vehicular network system. As an application, possible degrees of mitigation for attacks in attack trees are discussed.

1 Introduction

Progress in information technology has contributed to the evolution of various systems worldwide. In particular, cyber-physical systems now have more flexible and finer functionalities, and cooperate with other systems via networks. However, security threats on these systems have also increased. Protecting a system from security threats has become an important issue recently.

Attack trees are major tools in analyzing the security of a system. These trees represent the decomposition of threats in the form of **AND/OR**-trees (Examples are presented graphically in Fig. 1, 4), as well as fault trees represent structures of faults in a system in safety domain. We can analyze every scenario for a threat in a sub-tree of the attack tree [20]. Moreover we can evaluate the quantitative properties of the threat along with the corresponding attack tree.

Simple and intuitive descriptions of attack trees allow various extensions of the concepts. Examples include adding other types of nodes, connecting trees expressing defenses, and specifying the maximum number of children of

a node [23]. Specifically, to distinguish causal relationships among sub-attacks, attack trees are extended to ones with *sequential conjunctions* [1, 8, 13].

Fortunately attack trees are well formulated in some research [1, 8, 13, 15]; formal syntax and semantics are provided, the quantitative attributes are related to attacks formally, or attacks are linked to state transitions of the target system. However, all the aspects of attack trees are not supported by formal approach.

In particular, relations among attacks around a branch have not been discussed thoroughly. In an attack tree, a branch represents a concretization of the parent, another branch shows preceding events for the parent, or those are mingled together at the other branch. Resulting attack trees tend to be diverse, and it is a problem from the practical viewpoint. Although there are several studies [1, 2, 7] dedicated to this issue, the frameworks proposed in these research works are deemed rather indirect. Methodologies for considering the consistency of attack trees simply are required.

This paper tackles this problem by introducing the effects of attacks. An effect, considered as the post-condition of an event, is tightly related to an attack, and therefore we can discuss the abstract-refinement relation around a branch with the help of effects rather than attacks alone. Furthermore, when an attack depends on another one, they are related with effects, that is, the effect of the preceding attack works as the pre-condition of the subsequent attack. We define the consistency of a branch by logical conditions in terms of effects. This approach allows checking the validity of each branch in an attack tree with sequential conjunctions. As a result, attack trees and analyses for them can be described in more rigorous way.

In order to discuss intermediate nodes in attack trees with effects, we define a novel semantics of attack trees. Horne et al. [8] provided a semantics of attack tree with sequential conjunctions, which took values in the set of directed graphs whose nodes were labeled with primitive attacks. Here, a primitive attack corresponds to a leaf node in an attack tree. It meant that attack trees were interpreted as combinations of only primitive attacks. However, the focus on this paper is to investigate relationships among an attack and its sub-attacks, especially at intermediate nodes. Here we consider that an attack tree expresses a collection of inseparable refinement scenarios. The semantics proposed in this paper takes value in the powerset of sub-trees without **OR** branches. These sub-trees can derive directed graphs labeled by the leaf nodes in the attack tree, and therefore, the semantics can be related to the semantics proposed by Horne et al.

As an application of attack trees with effects, we evaluate countermeasures and possible mitigation for attacks. A countermeasure or mitigation eliminate some part of the consequences (i.e. effects) of attacks. Hence, the mitigated effects and the residuals can be described as fragments of effects. It enables to link the mitigation to the consistency of attack decomposition. To date, obtained results are rather rough; the cancellations of effects of the sub-attacks tend to be stronger than that of the parent. With a vehicular network system and its threats, this paper analyzes possible countermeasures in detail from this viewpoint.

Generally, an attack tree process for a threat consists of the following three steps: identification of the target, tree construction, and analysis. The first step is commonly conducted in system engineering, such as system development or risk management. To conduct the step in a systematic way, it is possible to follow the established methodologies like with SysML [6]. Moreover, formal analyses with attack trees have been developed for the third step. With the use of attributes, logical or quantitative properties of a threat are integrated according to the tree structure, and in this way we can evaluate the threat rigorously. However, systematic approach to the tree construction, the second step of the process, seems to be overlooked, as we pointed ambiguous relationships among attacks. Our results support tree constructions by observing consistency in a more direct, simpler and formal way. As a result, all steps in the attack tree process can be approached systematically, contributing into an improvement of attack tree analysis.

Organization. Section 2 deals with theoretical aspects of attack trees. Attack trees with sequential conjunctions and attributes are defined formally, after reviewing related works. Next, we introduce the concept of effects of attacks in Sect. 3. Consistent attack decompositions are also discussed with the use of effects. Based on this discussion, Sect. 4 illustrates a case study on threats on a vehicular network system. Moreover we attempt to estimate which grades are required for mitigation against attacks in attack trees. Finally, conclusions and future research directions are outlined in Sect. 5.

2 Attack Trees with Sequential Conjunctions

2.1 Overview of Attack Trees

Here, we review attack trees, particularly, about formal descriptions related to this paper and practical applications of them. Comparison with fault trees used in the safety domain are discussed in the last part of this section.

The concept of attack trees was firstly introduced by Schneier [20]. He expressed the decomposition of an attack as an **AND/OR** tree, and demonstrated several examples of evaluating of attacks using the tree. That is, an attack was evaluated by integrating the evaluations of sub-attacks along with the tree structure. Subsequently, this idea was formalized by Mauw and Oostdijk [15]. They specified a formal syntax of attack trees and defined the corresponding semantics as a set of multisets consisting of primitive attacks. Moreover they discussed the equivalence and transformations of attack trees compatible with the semantics. They also introduced evaluations of attack trees as *attributes*. An attribute was defined as a function from the nodes of an attack tree to a set, where the function values did not conflict with **AND** and **OR** decomposition. Indeed, the attribute of an attack in an attack tree was calculated with attribute values of the attack's children.

Attack trees have been applied in various domains for the purpose of attack modelling, although in many cases, the concept was not defined formally.

Security analyses together with attack trees were conducted for cyber-physical systems recently. The paper [21] showed an attack tree for an implantable medical device, and checked whether communication protocols for the device had vulnerabilities or not. The paper [3] analyzed security about a railway system. Attack tree was applied to identify detailed attack scenarios, while effect identification and risk evaluation was done with failure modes, vulnerabilities, effects analysis (FMVEA). In EVITA [18] for the automotive domain, attack trees were considered as major tools to identify attack scenarios and to estimate attack potentials. However approaches to build trees were not discussed there, other than considering abstract structures of trees. JASO TP 15002 [12] for automobiles suggested tree decompositions of selected threats to analyze how these threats could be realized. In DO-356 [17] for the aviation domain, tree diagrams were introduced to analyze security aspects. These diagrams were referred to as threat trees, as they were focused on the threat condition events and vulnerability events as well as attacks.

The idea of attack trees is rather simple, which allows various extensions. Wang et al. [23] classified many variants of attack trees. Fovino and Masera [4] enriched nodes of attack trees with related information such as assertions, vulnerabilities, and operations. With the enrichment, attacks or threats can be analyzed from several viewpoints in the research. The simplicity of attack trees also allows wider interpretations, and it means engineers may experience difficulties in building attack trees. To the best of our knowledge, most related studies have neither explicitly discussed the guidance for attack decompositions nor the validity of decompositions in detail. Although [1, 2, 7] discussed this issue, their frameworks dealt with attacks indirectly.

One of the major extensions of attack trees was to add a new type of branches, that is, sequential conjunction. In several cases, sub-attacks of an attack have causal dependency, and therefore it is natural to consider an attack tree together with the order of attack executions. Attack trees with sequential conjunction were discussed by Jhawar et al. [13] and by Horne et al. [8]. Their studies incorporated Mauw and Oostdijk's formalization [15], and the multiset semantics was extended to sets of graphs representing possible sequences of primitive attacks. Audinot et al. [1] also focused on attack trees with sequential conjunctions. They used pre-conditions and post-conditions of attacks for labels of nodes instead of attacks themselves. The semantics was given as the sets of the target's behaviors that satisfied the corresponding pre-conditions and post-conditions. Their framework shows the relationship among the parent node and its children of a branch clearly, but actual events by attacks are not presented explicitly. Furthermore, a transition model expressing possible behaviors in the target system should be prepared in advance. They discussed the consistency¹ of decomposition by comparing the semantics of nodes around a branch.

In safety domain, fault trees have been used for reliability analysis of systems since the 1960s. Fault trees were presented as **AND/OR**-trees as same as attack trees, but showed causal decompositions [9, 19]. In the literature, attack trees and

¹ It is called *correctness* properties in their paper.

fault trees were often dealt with similarly. For example interpretations were given as sets of labels on the leaf nodes in the tree, or properties of the uppermost event of the tree were quantitatively calculated with the properties of leaf nodes [15, 19]. Moreover, with recent observation that security threats caused harms in safety-critical systems, the integration of attack trees and fault trees were proposed [5] to connect security analysis and safety analysis.

2.2 Formulation

We provide a formal definition of attack trees. Its syntax is defined inductively, and the semantics represents possible scenarios of attack refinements. In the sequel, we denote attack trees, including sequential conjunctions.

Definition 1. *An attack tree is a labeled tree with three types of branches:*

$$\begin{aligned}
 t &::= Lf(n) \mid Nd(n, op, \langle t, t, \dots, t \rangle), \\
 op &::= \mathbf{AND} \mid \mathbf{OR} \mid \mathbf{SAND},
 \end{aligned}$$

where $\langle - \rangle$ means a non-empty finite sequence of its arguments, and where the symbol n is a label for the node, which expresses an action or an event.

Intuitively, $Lf(n)$ corresponds to a primitive attack n , which is no longer decomposed (a leaf node in a tree), and $Nd(n', op, \langle t_1, \dots, t_k \rangle)$ corresponds to an attack n' , which has sub-attacks t_1, \dots, t_k with type op as its decomposition (an intermediate node in a tree). Attack trees can be diagrammatically represented, as illustrated in Fig. 1. The type of a branch is expressed by a gate symbol under nodes. Around a **SAND** branch (“S”-marked **AND** gate), actions in the child nodes are executed from left to right. The uppermost node of an attack tree is called the root node.

We do not consider the order of children for **AND** or **OR** branches, as well as the type of a branch having only one child. Hence the following equations are assumed for arbitrary subtrees $\{t_j\}_{1 \leq j \leq k}$ and a subtree t :

$$\begin{aligned}
 &Nd(n, op, \langle t_1, \dots, t_i, t_{i+1}, \dots, t_k \rangle) \\
 &= Nd(n, op, \langle t_1, \dots, t_{i+1}, t_i, \dots, t_k \rangle) \quad (op \in \{\mathbf{AND}, \mathbf{OR}\}) \\
 &Nd(n, op, \langle t \rangle) = Nd(n, op', \langle t \rangle) \quad (op, op' \in \{\mathbf{AND}, \mathbf{SAND}, \mathbf{OR}\}).
 \end{aligned}$$

We denote an attack tree without **OR** branches as an *R-tree*. Intuitively, an R-tree expresses an individual refinement scenario regarding the attack of the root node.

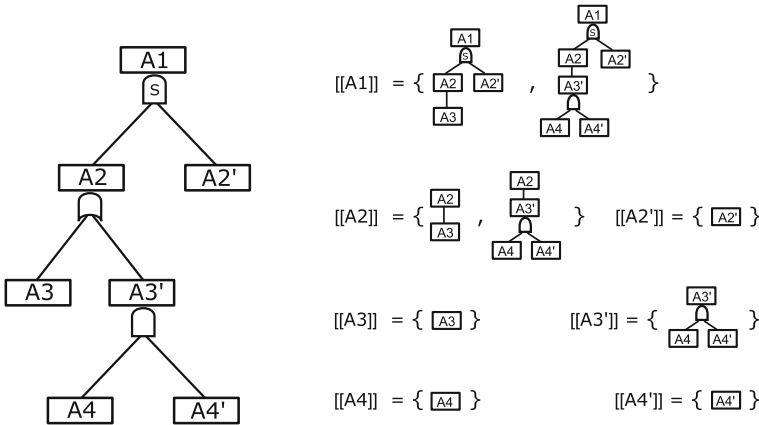
A semantics $\llbracket \cdot \rrbracket$ of attack trees is the function which maps an attack tree to a multiset of R-trees.

Definition 2. The function $\llbracket \cdot \rrbracket$ on the set of attack trees is defined by the following rules, where $\bar{t} = \langle t_1, \dots, t_m \rangle$ and $\llbracket \bar{t} \rrbracket = (\llbracket t_1 \rrbracket, \dots, \llbracket t_m \rrbracket)$:

$$\begin{aligned} \llbracket Lf(n) \rrbracket &= \{Lf(n)\}, \\ \llbracket Nd(n, \mathbf{AND}, \bar{t}) \rrbracket &= \{Nd(n, \mathbf{AND}, \langle \tau_1, \dots, \tau_m \rangle) \mid (\tau_1, \dots, \tau_m) \in \llbracket \bar{t} \rrbracket\}, \\ \llbracket Nd(n, \mathbf{SAND}, \bar{t}) \rrbracket &= \{Nd(n, \mathbf{SAND}, \langle \tau_1, \dots, \tau_m \rangle) \mid (\tau_1, \dots, \tau_m) \in \llbracket \bar{t} \rrbracket\}, \\ \llbracket Nd(n, \mathbf{OR}, \bar{t}) \rrbracket &= \bigsqcup_{1 \leq i \leq n} \{Nd(n, \mathbf{AND}, \langle \tau \rangle) \mid \tau \in \llbracket t_i \rrbracket\}. \end{aligned}$$

The semantics is based on the idea that decompositions appearing in attack trees are logical refinement. An **OR** branch is interpreted as a multiset union, indicating this branch corresponds to a case division. An aspect of the attack is refined, and possible detailed attacks are listed as sub-attacks. On the other hand an **AND/SAND** branch is interpreted as a factorization of an attack to sub-attacks. The collection of sub-attacks around the branch is inseparable, as a single sub-attack in it does not invoke the original attack. The causal dependency of attacks exists only between children of each **SAND** branch, and does not exist elsewhere, especially between an attack and its sub-attacks.

Comparing our formulation (Definition 1, 2) with those in [8], syntaxes are very similar - the difference is whether a branch is limited to binary or not. However, our semantics keeps intermediate nodes and analyses of them are available, whereas the semantics in [8] only considers leaf nodes.



The gates under the node A1, A2, and A3' present **SAND**, **OR**, and **AND** branches respectively. As examples, nodes labeled A4 and A3' are expressed as $Lf(A4)$ and $Nd(A3', \mathbf{AND}, \langle Lf(A4), Lf(A4') \rangle)$.

Fig. 1. Attack trees and their interpretation.

2.3 Attributes

An attribute of an attack tree is defined as a function f from the set of nodes. The codomain D of f is a set with the three operations μ_O , μ_A , and μ_S , corresponding to **OR**, **AND**, and **SAND**, respectively. Around a branch in an attack tree, attribute values of the child nodes are summarized with these operations. Therefore the following equalities are required to hold ($\varphi \in \{\mu_O, \mu_A\}$):

$$\begin{aligned} \varphi(\langle f(x_1), \dots, f(x_i), f(x_{i+1}), \dots, f(x_k) \rangle) \\ &= \varphi(\langle f(x_1), \dots, f(x_{i+1}), f(x_i), \dots, f(x_k) \rangle), \\ \mu_A(\langle f(x) \rangle) &= \mu_S(\langle f(x) \rangle) = \mu_O(\langle f(x) \rangle). \end{aligned}$$

One example of attributes is the minimum number of experts required to perform an attack, discussed in [8]. When we denote the defining function of the attribute by ν , its codomain is defined as the set of natural numbers \mathbb{N} and $(\mu_O, \mu_A, \mu_S) = (\min, \text{Sum}, \max)$. Remark that this attribute is assumed to be determined by the values of lower nodes rigorously. Namely, $\nu(Nd(n, \mathbf{OR}, \bar{t})) = \min\{\nu(t_1), \dots, \nu(t_k)\}$ where $\bar{t} = \langle t_1, \dots, t_k \rangle$, and similar equations hold for **AND/SAND** branches. When there is another attribute for which we cannot have the assumption, the equalities may not be expected and we must consider contributions for the attribute by intermediate nodes themselves. Such an attribute will be compatible with the semantics in Definition 2, but not with that in [8] using only leaf nodes.

3 Validating Decompositions with Effects

3.1 Effects of Attacks

An effect is one of the major properties of an attack. It is a situation or a property of a specific entity related to the target system, and is caused by a specific action. Let us consider the attack “Message receive function is interfered” as an example. After the attack, messages may be lost, the function may be unavailable, or some irregular behaviors occur. These situations have not occurred before the attack, and therefore it can be considered that the attack causes these situations. As a result, we can identify the summarized situation “messages are not processed correctly” as the effect of the attack.

Effects are also significant concepts in the areas adjacent to security. In the safety standard ISO/IEC Guide51 [11], negative effects² on people, property or the environment, caused by some event are the primary issues to be avoided. In the standard ISO 31000 [10] focused on risk management, a risk is defined as an effect of uncertainty on objectives.

Owing to the above observation for effects, it seems reasonable that *an effect of an attack* meets the following conditions:

² Those effects are referred to as harms.

- The effect must be directly caused by the corresponding attack; it shows a change of the entity that the attack affects.
- Properties that hold before the attack must not be selected as effects.
- The effect occurs immediately after the corresponding attack; no other properties invoked by the attack occur before the effect occurs.

The effect of an attack can be described by propositional formulas. Logical expressions serve as standard tools to consider relationships and operations on properties. In particular, they allow considering refinement of properties easily. When a property P is refined as P' , the inference $P' \Rightarrow P$ holds.

For an attack tree, we assign a formula expressing an effect to a node. In an attack tree diagram, we put the round node labeled by the effect and connect it to the corresponding attack with a blue edge (see Fig. 2).

The effect is considered as an attribute of attack trees. Let us denote the mapping from nodes to effects by ε . The codomain of ε is the set of propositional formulas \mathcal{P} , and the three operations are $\mu_O = \vee$ (disjunction), $\mu_A = \wedge$ (conjunction), and $\mu_S = \mu_p = \lambda\langle e_1, \dots, e_r \rangle.e_r$ (the projection to the last element). However the attribute value of the upper node is not always determined by the attribute values of the lower nodes. Their relationship are discussed in the next section.

As effects are strongly connected to attacks, there can be analyses of attack trees with use of effects. One possibility is the treatment of threats. The treatment cancels some part of the effects of an attack, and hence the effectiveness of treatment can be evaluated. In Sect. 4.3, some properties about treatments are observed and applied in the case study.

3.2 Consistent Branches

As a branch in an adequately constructed attack tree represents a decomposition of the attack corresponding to the parent node, a similar structure can be expected with regard to effects around the branch. For instance, the effect of the parent node will include the conjunction of all effects of the child nodes for an **AND** branch, as all of the attacks corresponding to the child nodes are conducted. By contrast, when there are several conflicts among the effects around a branch, the decomposition of the attack will have inconsistencies, such as misunderstanding of the situation, or inadequate refinement. To analyze whether an attack decomposition is valid or not, we introduce the consistency of a branch with the assigned effects as follows.

Definition 3. *A branch in an attack tree is defined as consistent, if the condition below holds regarding its type, where e_P is the effect assigned to the parent node and e_1, e_2, \dots, e_r are the effects assigned to the children:*

- **OR** branch: *the effect of the parent is inferred from the effect of each child node, i.e. $e_i \Rightarrow e_P$ for each i .*
- **AND** branch: *the effect of the parent is inferred from the conjunction of the effects of all children, i.e., $\bigwedge_i e_i \Rightarrow e_P$.*

- **SAND** branch: the effect e_k is obtained by the k -th attack from the left with assumption e_1, \dots, e_{k-1} , and e_P is inferred from the rightmost effect e_r .

These conditions are depicted in Fig. 2.

If all branches in an attack tree are consistent, then the entire attack tree is consistent.

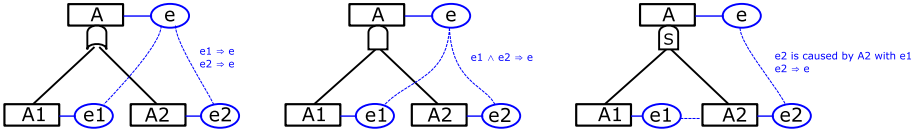


Fig. 2. Consistency of attacks with effects.

Actual examples of attack trees with effects are provided in Sect. 4.2.

As an attribute, effects behave appropriately for a consistent attack tree. The following property is induced straightforwardly.

Proposition 1. Consider the attribute ε as defined above, and take a branch in a consistent attack tree where e_P , [resp. e_1, e_2, \dots, e_r] is the value of ε for the parent [resp. children]. Then the inference $\nu(\langle e_1, \dots, e_r \rangle) \Rightarrow e_P$ holds for $\nu = \vee$ [resp. \wedge, μ_p] if the branch is an **OR** [resp. **AND**, **SAND**].

Remark that it is evident that the effect of the root node can be inferred with the effects of the leaf nodes.

4 Case Study

4.1 Process Overview

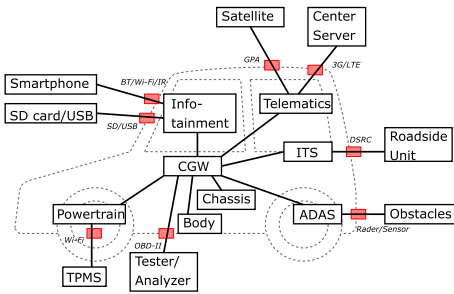
The security analysis process specified in JASO TP15002 [12] consists of the following five phases: ToE (Target of Evaluation) Definition, Threat analysis, Risk assessment, Define security objectives, and Security requirement selection. In [14], the authors studied concepts and their relationship appearing in the process in detail. It allows refactoring activities and the data dealt with in the security analysis process.

A deliverable of the Risk assessment phase is a list of identified threats on ToE. Threats in this list are prioritized, and significant ones are analyzed in depth using tree diagrams. Countermeasure goals for the threats are discussed in the Define security objective phase.

4.2 Verifying Decompositions

Based on the model in [16], a vehicular network system can be specified as ToE (see Fig. 3). Each module inside the vehicle has its own assets. For example, the assets of the Powertrain module are Control function, Authentication function, Authentication information, and Sensor information.

Here, we consider the identified threat “*Authentication function in Powertrain is interfered via TPMS*”³. It is estimated significant because the target (Powertrain) is closer to the entry point (TPMS), and the potential damage to the vehicle system is more severe.



Red boxes indicate entry points. Modules inside the vehicle may be accessed through them.

Fig. 3. The network architecture of ToE.

Figure 4(a) shows an early version of the attack tree (only the upper part is outlined). It should be noted that attacks on node labels include the events that contribute to invoking the parent but are not performed by the attacker. Policies and methods used to construct the tree are rather abstract. Therefore sub-attacks of an attack are intuitively selected; some of sub-attacks do not refine the parent but are expected to occur preceding to the parent. Here, decomposition is interpreted as causal ones implicitly. Moreover only **OR** and **AND** branches are considered. As a result, the following two inconsistencies are found:

1. A temporal-gap among attacks around a branch. The attack A1 has to occur before its parent A0, but the attacks A2 and its parent A0 occur simultaneously.
2. A violated refinement order. The Msg. identification function in A1 refines the Authentication function in A0, whereas Powertrain Software mentioned in A1.1 is a wider entity than Msg. identification function in A1.

These un-structural situations are made explicit by considering the effects of attacks. As mentioned in Sect. 3.1, effects are changes in ToE and its environment caused by specific actions. Here, we identify the entities affected by attacks, and decide effects on them derived from the attacks. The result is illustrated in

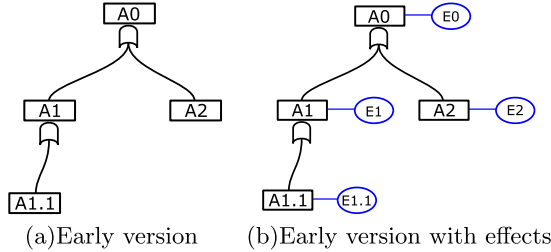
³ Tire Pressure Monitoring System.

Node labels:

- A0: Authentication function in Powertrain is interfered with via TPMS.
- A1: TPMS msg identification function in Powertrain is tampered with.
- A1.1: Powertrain Software is tampered with.
- A2: TPMS msg identification function in Powertrain is interfered with by DoS.

Effects(Underlined parts are the targets of attacks):

- E0: Unintended behaviors in Authentication function.
- E1: Installed TPMS msg identification function is invalid.
- E1.1: Installed Powertrain Software is invalid.
- E2: TPMS msg identification function is not available.



Revised node labels:

- A1: Unauthorized TPMS msg identification function in Powertrain is invoked.
- A1.1.1: TPMS msg identification function in Powertrain is tampered with by replacing Powertrain software.
- A1.2: (Unauthorized) TPMS msg identification function in Powertrain is invoked.

Revised effects:

- E1: Unintended behaviors in TPMS msg identification function.
- E1.1.1: Installed TPMS msg identification function is invalid.
- E1.2: Unintended behaviors in TPMS msg identification function.

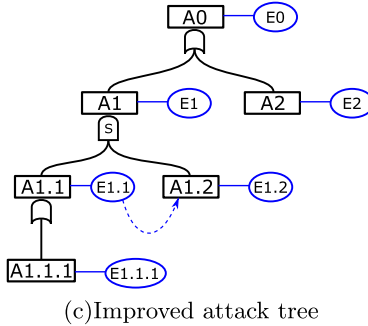


Fig. 4. Improvement of an attack tree.

Fig. 4(b). As the non-availability of a sub-function (the effect of A2) is an instance of unintended behaviors (the effect of A0), their occurrences are not separate, i.e. $E2 \Rightarrow E0$ holds. On one hand, the existence of adverse software (the effect of A1) does not mean the appearance of unintended behaviors immediately, i.e. $E1 \Rightarrow E0$ does not hold. It shows that A1 breaks the consistency of the **OR** branch under A0. Moreover, the branch under A1 is inconsistent, as the fact Powertrain Software is invalid does not imply that Msg. identification function is invalid.

The revised version of the attack tree is outlined in Fig.4(c). The second inconsistency mentioned above is simply resolved by emphasizing the target(A1.1.1). We modify the node A1 and add a **SAND** branch under it to ensure consistency. At last all branches are consistent in this tree, and the entire attack tree is consistent, as well.

4.3 The Degrees of Possible Mitigation

Attacks can be treated by countermeasures. A countermeasure prevents an attack or modifies its results, and therefore it mitigates the effect of the attack. When the original effect e is weakened to e' , e can be expressed as $e' \wedge e''$. Here e'' corresponds to the partial effect the countermeasure cancels, and e' and e'' are independent. In risk management, the four types of treatments are introduced (according to Chapter 6 in [22]): avoidance, limitation/reduction, transference, and acceptance. If we consider them for the purpose of our discussion, then avoidance corresponds to the case when e'' is e itself, and acceptance corresponds to the case when e' coincides e in terms of the same symbols mentioned above. This paper does not consider transference, since it may replace e by another effect and make difficult to estimate the effectiveness of the countermeasure.

Inferences of the properties and corresponding treatments can be related. However the relation is rather rough, that is, two reductions for the effect of an attack cannot be compared by the relations.

Lemma 1. *Let e_p and e_c be properties satisfying $e_c \Rightarrow e_p$. If the mitigated e_p is still inferred from the mitigated e_c , then the mitigation of e_p is weaker than that of e_c . Rigorously, if e_p is weakened by avoidance, then so must be e_c , and if e_p is weakened by reduction, then e_c must be weakened by avoidance or reduction as well.*

The lemma can be proven by a simple observation. Let us split e_p and e_c to $e'_p \wedge e''_p$ and $e'_c \wedge e''_c$ respectively, where e''_p and e''_c are removed by mitigation. Remember that the inference $e_c \Rightarrow e_p$ implies $e'_c \wedge e''_c \Rightarrow e'_p$ and $e'_c \wedge e''_c \Rightarrow e''_p$. If e_p is mitigated by avoidance, that is, $e_p = e''_p$ and e'_p does not actually exist, then the inference $e'_c \Rightarrow e'_p$ is contradictory, unless e'_c does not exist. Similarly, if e_p is mitigated by reduction but e_c is mitigated by acceptance, then e''_p still holds even after mitigation by $e_c \Rightarrow e''_p$.

On the basis of the above lemma, we can restrict mitigation of children in attack trees. However, before stating that, several conditions are assumed for simplicity:

- Children’s effects around an **AND** branch are independent. None of them is inferred by other effects, and the countermeasure for an effect is not overridden by countermeasures for other effects.
- Each child’s effect around a **SAND** branch is not modified by subsequent attacks. Hence the entire effect by the children around a **SAND** branch is the conjunction of effects by every child.

Proposition 2. *Consider the effect e_p of the parent and e_1, \dots, e_k of the children around a branch of an attack tree. Then, the mitigation for e_p must be weaker than or equal to that of all children, if the branch is **OR** and is still consistent after the mitigation. In the case of the **AND/SAND** branch, the mitigation for e_p must be weaker than or equal to the mitigation of at least one child's.*

Let us first consider that the type of the branch is **OR**. According to the semantics, it is sufficient to check the relationship between each child and the parent. The above lemma is applied to an individual child directly, and it is observed that the mitigation of the parent is weaker than that of each child.

Remember that the consistency means $(\wedge_i e_i) \Rightarrow e_p$ when the type of the branch is **AND/SAND**. We can apply the lemma corresponding to e_c as $\wedge_i e_i$, that is, the mitigation for e_p must be weaker than or equal to that of $(\wedge_i e_i)$. It indicates that the mitigation for every e_i must be avoidance if e_p is mitigated by avoidance. Moreover it indicates that the mitigation for all children must not be acceptance if e_p is mitigated by reduction, that is, at least one e_i must be mitigated by reduction or avoidance. Finally, the proposition is obtained.

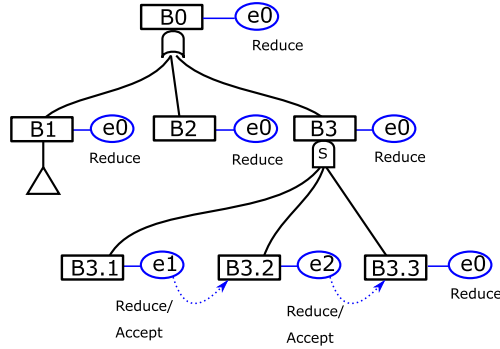
Remark that we can observe stronger results for **SAND** branches. A weakened effect for a child by avoidance or reduction may become an insufficient precondition of the subsequent attacks. Consequently, the attack is not invoked, and its effect does not occur as well. Therefore, mitigating the leftmost sub-attack by avoidance/reduction is sufficient to mitigate the entire $\wedge_i e_i$ by avoidance.

Considering the case study regarding the vehicular network system. After analyzing significant threats using attack trees, countermeasures are considered for primitive attacks of the threats that serve as labels on leaf nodes. The selected countermeasures are those that will negate the corresponding primitive attacks, but some of them may be overreactions. Proposition 2 supports the determination of whether or not the selections are adequate.

Considering the identified threat “*Authentication information in Infotainment is stolen via BT/WiFi/IR*” (B0) and the attack tree for it (Fig. 5), we note that the threat is less critical, as its effect, *the disclosure of the information*, does not affect the drive control of vehicles immediately. The treatment for the threat can be reduction; for example updating the information after a fixed period of time. By referring to Proposition 2, each sub-attack of B0 can be treated by reduction or avoidance. Similarly, all sub-attacks of B3 must not be mitigated by acceptance. Therefore we have several options: to accept obtaining device (B3.1) and eavesdropping (B3.2) but make it difficult to extract authentication information (B3.3), or to restrict obtaining device (B3.1) and simultaneously restrict opportunities to eavesdrop (B3.2) and to extract the information (B3.3).

5 Conclusion

We define a new formal system of attack trees with sequential conjunction, and the consistency around a branch in terms of effects of attacks. They match the



- B0: Auth. info. in Infotainment is stolen via BT/Wifi/IR.
- B1: Auth. info. in Infotainment is obtained by accessing Infotainment.
- B2: Auth. info. in Infotainment is obtained by fake communication request.
- B3: Auth. info. in Infotainment is obtained by eavesdropping BT/Wifi/IR.
- B3.1: Device to eavesdrop is prepared.
- B3.2: Msg.s for authentication are collected.
- B3.3: Auth. info. is extracted from eavesdropped Msg.s
- e0: Auth. info. is disclosed.
- e1: Ready for eavesdropping.
- e2: Analyzing Msg.s are available.

Fig. 5. Attack tree with effects and treatment types. The triangle connected to B1 indicates the node is not decomposed here.

interpretation of attack trees as representations of refinement scenarios, and therefore we can discuss the validity of attack trees formally. These ideas lead to fine and accountable attack models. Consequently, the construction of attack trees is improved, and it becomes possible to analyze attack trees rigorously.

As an application, degrees of mitigation for attacks are measured. Although it is a slightly rough, we show examples and observation for that, with use of a vehicular network system.

There are several issues left to be considered in the future research:

- Our formulation of attack trees does not cover complicated phenomena including the evolution of effects by subsequent attacks, or non-linear relations among effects and attacks. Developing more expressive attack trees will give finer models of attacks appearing in the real world. In addition, semantics should be rigorously related to ones in existing formalizations.
- Research on the derivation of the effects from attacks are required. Additional information, for example a structure of ToE, may allow determining effects in systematic way that reduces informal discussion in attack tree processes, and that validates attack trees entirely at a higher level.
- The evaluation of mitigation can be enhanced. From the practical point of view, it is beneficial if we can evaluate how effective a mitigation is for a specific attack. Further research may refine Proposition 2 and we can choose the most appropriate mitigations for attacks.

References

1. Audinot, M., Pinchinat, S., Kordy, B.: Is my attack tree correct? (Extended version). [arXiv:1706.08507](https://arxiv.org/abs/1706.08507) (2017)
2. Audinot, M., Pinchinat, S., Kordy, B.: Guided design of attack trees: a system-based approach. In: Computer Security Foundations Symposium, CSF 2018, pp. 61–75 (2018)
3. Chen, B., et al.: Security analysis of urban railway systems: the need for a cyber-physical perspective. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 277–290. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_24
4. Fovino, I.N., Masera, M.: Through the description of attacks: a multidimensional view. In: Górski, J. (ed.) SAFECOMP 2006. LNCS, vol. 4166, pp. 15–28. Springer, Heidelberg (2006). https://doi.org/10.1007/11875567_2
5. Fovino, I.N., Masera, M., Cian, A.D.: Integrating cyber attacks within fault trees. *Reliab. Eng. Syst. Saf.* **94**(9), 1394–1402 (2009)
6. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language, 3rd edn. Morgan Kaufmann Publishers Inc., Burlington (2014)
7. Gadyatskaya, O., Trujillo-Rasua, R.: New directions in attack tree research: catching up with industrial needs. In: Liu, P., Mauw, S., Stølen, K. (eds.) GramSec 2017. LNCS, vol. 10744, pp. 115–126. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74860-3_9
8. Horne, R., Mauw, S., Tiu, A.: Semantics for specialising attack trees based on linear logic. *Fundam. Inform.* **153**(1–2), 57–86 (2017)
9. IEC 61025: Fault tree analysis (FTA) (2006)
10. ISO 31000: Risk management - Guidelines (2018)
11. ISO/IEC Guide 51: Safety aspects - Guidelines for their inclusion in standards (2014)
12. JASO TP15002: Guideline for Automotive Information Security Analysis (2015)
13. Jhavar, R., Kordy, B., Mauw, S., Radomirović, S., Trujillo-Rasua, R.: Attack trees with sequential conjunction. In: Federrath, H., Gollmann, D. (eds.) SEC 2015. IAICT, vol. 455, pp. 339–353. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18467-8_23
14. Kawanishi, Y., Nishihara, H., Souma, D., Yoshida, H., Hata, Y.: A comparative study of JASO TP15002-based security risk assessment methods for connected vehicle system design. *Secur. Commun. Netw.* (2019). <https://doi.org/10.1155/2019/4614721>
15. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006). https://doi.org/10.1007/11734727_17
16. Miyashita, Y., et al.: On-vehicle compact and lightweight multi-channel central gateway unit. *SEI Tech. Rev.* **83**, 5–9 (2016)
17. RTCA DO-356: Airworthiness Security Methods and Considerations (2014)
18. Ruddle, A., et al.: Security requirements for automotive on-board networks based on dark-side scenarios. *EVITA Deliverable D2.3* (2009)
19. Ruijters, E., Stoelinga, M.: Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. *Comput. Sci. Rev.* **15–16**, 29–62 (2015)
20. Schneier, B.: Attack trees. *Dr. Dobb's J.* **24**(12), 21–29 (1999)

21. Siddiqi, M.A., Seepers, R.M., Hamad, M., Prevelakis, V., Strydis, C.: Attack-tree-based threat modeling of medical implants. In: PROOFS 2018, 7th International Workshop on Security Proofs for Embedded Systems, vol. 7, pp. 32–49 (2018)
22. Susan Snedaker, C.R.: Business Continuity and Disaster Recovery Planning for IT Professionals, 2nd edn. Elsevier, Amsterdam (2014)
23. Wang, J., Whitley, J.N., Phan, R.C.W., Parish, D.J.: Unified parametrizable attack tree. *IJISR* **1**(1/2), 20–26 (2011)