The Marriage Between Safety and Cybersecurity: Still Practicing

Marielle Stoelinga[1,2(✉)], Christina Kolb[1], Stefano M. Nicoletti[1], Carlos E. Budde[3], and Ernst Moritz Hahn[1]

[1] Formal Methods and Tools, University of Twente, Enschede, The Netherlands
{m.i.a.stoelinga,c.kolb,s.m.nicoletti,c.e.budde,e.m.hahn}@utwente.nl
[2] Department of Software Science, Radboud University, Nijmegen, The Netherlands
[3] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
carlosesteban.budde@unitn.it

Abstract. Emerging technologies, like self-driving cars, drones, and the Internet-of-Things must not impose threats to people, neither due to accidental failures (safety), nor due to malicious attacks (security). As historically separated fields, safety and security are often analyzed in isolation. They are, however, heavily intertwined: measures that increase safety often decrease security and vice versa. Also, security vulnerabilities often cause safety hazards, e.g. in autonomous cars. Therefore, for effective decision-making, safety and security must be considered in combination.

This paper discusses three major challenges that a successful integration of safety and security faces: (1) The complex interaction between safety and security (2) The lack of efficient algorithms to compute system-level risk metrics (3) The lack of proper risk quantification methods. We will point out several research directions to tackle these challenges, exploiting novel combinations of mathematical game theory, stochastic model checking, as well as the Bayesian, fuzzy, and Dempster-Schafer frameworks for uncertainty reasoning. Finally, we report on early results in these directions.

Keywords: Safety · Security · Model-based · Interaction · Fault trees · Attack trees · Fault tree-attack tree integration

1 Introduction

New technology comes with new risks: drones may drop on to people, self-driving cars may get hacked, medical implants may leak in people's body. Such risks concern both accidental failures (safety) and malicious attacks (security). Here, *security* refers to the property that allows the system to perform its mission or critical functions despite risks posed by threats [25]. *Safety*, in contrast, is the absence of risk of harm due to malfunctioning behavior of technological systems [32].

This work was partially funded by ERC Consolidator Grant 864075 (*CAESAR*).

A. Laarman and A. Sokolova (Eds.): SPIN 2021, LNCS 12864, pp. 3–21, 2021.
https://doi.org/10.1007/978-3-030-84629-9_1

Safety and security are heavily intertwined. Measures that increase safety may decrease security and vice versa: the Internet-of-Things offers ample opportunities to monitor the safety of a power plant, but their many access points are notorious for enabling hackers to enter the system. Passwords secure patients' medical data, but are a hindrance during emergencies. It is therefore widely acknowledged, also by international risk standards [21,39], that safety and security must be analyzed in combination [3,36]. The overarching challenge in safety and security risk management is decision making: which risks are most threatening, and which countermeasures are most (cost-)effective? Such decisions are notoriously hard to take: it is well-known (e.g. from Nobel prize winner Daniel Kahneman [24]) that people, have very poor intuitions for risks and probability.

Vision. To make effective decisions, *risk management should be accountable.*
1. *systematic*, so that no risks are overlooked;
2. *transparent*, so that experts can share and discuss their viewpoints;
3. *objective*, i.e. based on recorded facts and figures, rather than on (fallible) intuitions.

Hurdles. Tough hurdles that have hindered the effective integration of safety and security [1,32] are their opposite perspectives on:

H1. *User intention:* safety concerns unintended mishaps, while security is about malicious attacks.
H2. *Dynamics:* Whereas safety analysis is often static, developing design-time solutions; security demands constant defence against new vulnerabilities.
H3. Risk *quantification:* Whereas safety analysis can fruitfully exploit historic failure data, risk quantification for security is a major open problem. With hackers continuously changing their targets and strategies, historic data is of little value. Therefore, security decisions are often based on subjective estimates.

The demanding challenge in safety-security co-analysis is to overarch these diametrical viewpoints.

Challenges. To overcome these hurdles above and make decision making about safety and security less ad hoc, and more systematic, transparent, and quantitative, three challenges have to be solved.

- A systematic way to map safety and security risks, identifying how failures and vulnerabilities propagate through the system and lead to system level disruptions.
- Effective algorithms to compute system level risk safety and security metrics, together with diagnostic algorithms that explain how such metrics arise, and how one could improve these.
- Novel risk quantification methods. Reliable numbers are indispensable in decision making. Since objective data is often not available, we need algorithms that reason under uncertainty.

The ERC project CAESAR. The ERC-funded project CAESAR picks up the challenges above, exploring three novel directions:

1. *Game-theoretic methods* uniting the cooperative versus malicious user intention in safety versus security (H1). Our aim is to model the attacker versus defender as two players in a stochastic game. We will focus on a time dependent game (H2) that faithfully models the complex interaction between safety and security aspects.
2. *Stochastic model checking techniques* to compute safety-security risk metrics. Metrics are pivotal to prioritize risks and select effective countermeasures, as they clarify how failures and attacks affect system-level performance. Since risk is defined as a combination of likelihood and severity, metrics are stochastic by nature. Apart from computing numbers, we will also elucidate how these numbers arise.
3. *Risk quantification methods that handle data uncertainty.* Effective decision making requires insight in the most frequent failures and attacks. Since objective data is scarce, security decisions are often based on subjective estimates. We will combine objective and subjective probabilities, exploiting three prominent frameworks for data uncertainty: Bayesian reasoning, fuzzy logic and Dempster-Schafer theory. These explicate the underlying assumptions and (dis)agreements about risks.

Contributions. This paper outlines the first results and the approach taken in CAESAR: we present findings of a literature survey, where we compare existing formalisms for safety-security co-analysis and identify several gaps. An important outcome of our survey is that most of these formalisms are based on various combinations of the popular formalisms of attack trees for security analysis and fault trees for safety analysis. One important instance of a research gap is that in fault trees and attack trees OR-gates are interpreted in a different manner. This difference in interpretation is not considered in current analysis algorithms. To obtain a unified framework for safety and security building on these mechanisms, we thus have to unify the interpretation of such gates. Afterwards, we outline how CAESAR aims to solve such gaps. We discuss how recent results in attack tree analysis provides results for tree-shaped attack trees and fault trees as well as a formal semantics of DAG-shaped attack trees and fault trees. In these results, we exploit methods based on binary decision diagrams. Throughout the paper, we indicate current results as well a research gaps.

Organization of the Paper. Section 1 has provided an introduction to this paper. Section 2 provides some background on the interaction between safety and security and on the two formalisms attack trees and fault trees. Section 3 presents an overview of formalisms of safety-security co-analysis. Section 4 provides a comparison between attack trees and fault trees, highlighting similarities and differences and defining metrics. Section 5 discusses analysis algorithms for attack and fault trees. Section 6 concludes the paper.

2 Background

Attack Trees and Fault Trees. Attack trees and fault trees are popular models in dependability analysis, representing respectively how low-level attacks and failures propagate through the system and lead to system-level disruptions. As shown in Fig. 1, these are tree-like structures that culminate in a top level event (TLE), which models a system-level failure or attack. The TLE is thus refined via intermediate events, equipped with gates: the AND-gate indicates that all children must fail (be attacked) in order for the gate to fail (be attacked). For the OR-gate to fail (be attacked), at least one of its children need to fail (be attacked). The leaves in a FT are called *basic events (BEs)* and model atomic failures; the leaves in ATs model atomic attack steps, called *basic attack steps (BASs)*. Despite their names, FTs and ATs are directed acyclic graphs, since intermediate events can share children.

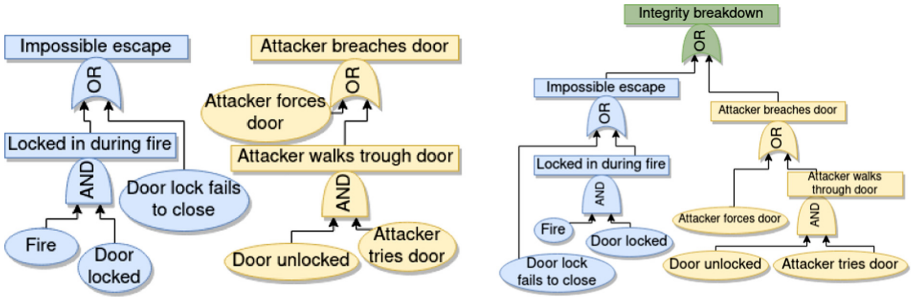

Fig. 1. Fault tree (left), attack tree (center) and their combination (right). These represent respectively safety, security, and combined risks. In the FT, for the intermediate event "locked in during fire" to happen, both a fire and the door being locked have to occur, modelled through an AND-gate. In the AT, for an attacker to breach the front door he/she needs to either walk through an unlocked door or to force a locked door, modeled as an OR. On the right, a possible combination of ATs and FTs in the attack fault trees formalism [46].

FTs and ATs enable numerous analysis methods [28]: *Cut set analysis* indicates which combinations of BEs or BASs lead to the TLE. The set {Fire, Locked} is a cut set in Fig. 1. Quantitative analyses compute *dependability metrics*, such as the system reliability, attack probabilities and costs. For example, by equipping the BEs and BASs with probabilities, one can compute the likelihood of a system level failure or attack to occur.

Both FTs and ATs are part of international standards [14] and have been used to analyse numerous case studies [11,19,48]. FTs and ATs also feature some differences: FTs often focus on probabilities, whereas ATs consider several other attributes, like cost, effort and required skills. Further, FTs have been extended with repairs [44], and dynamic gates [17,22]; ATs with defenses, and sequential AND (SAND) gates [19,27].

Section 4 presents a more formal treatment of fault trees and attack trees, and in particular their different quantitative interpretation of the OR-gate. Their comparison is summarized in Table 3.

Safety-Security Dependencies. One of the key challenges in safety-security co-analysis is to model their interaction. The paper [32] has identified four safety-security dependencies:

- *Conditional dependency* means that fulfillment of safety requirements conditions security or vice-versa.
- *Mutual reinforcement* means that fulfillment of safety requirements or safety measures contributes to security, or vice-versa, thereby enabling resource optimization and cost reduction.
- *Antagonism* arises when safety and security requirements or measures, considered jointly, lead to conflicting situations.
- *Independency* means that there is no interaction between safety and security properties.

Figure 1 shows a classical example of antagonism: the door needs to be locked in order to prevent an attacker from entering the house (security requirement), but it has to be unlocked to allow the owner to escape during a fire (safety requirement). In this scenario, mutual reinforcement can be achieved by introducing a fire door: this contributes to safety by limiting the spread of an eventual fire and to security by increasing the robustness of the door, thus making it harder to breach. Conditional dependency is, in our view, always present: for the lock to ensure security, it must function properly. E.g., it must not break when locking door. Similarly, safety solutions must be secure and not be hacked: it must not be possible to easily force the door open.

3 Formalisms for Safety-Security Co-analysis: An Overview

3.1 The Formalisms

As a first step in the CAESAR project, we carried out a literature survey [26], comparing the most prominent formalisms for safety-security co-analysis. Via a systematic literature search [8], which also considered earlier surveys on this topic [12,32,38], we have identified 10 important formalisms for model-based safety-security co-analysis. These are summarized in Table 1.

A first remarkable result of our survey is that the majority of safety-security formalisms combines attack trees (ATs) and fault trees (FTs). ATs and FTs are well established formalisms, extensively used in industry and academia. As previously mentioned, they are similar nature, and model respectively how failures and attacks propagate through a system. In that sense, combining attack trees and fault trees is a natural step. We further divided these approaches into two categories (plain combinations and extended combinations). A third category comprises architectural formalisms.

Table 1. Overview of safety-security formalisms. Citations from Google Scholar, April 2021.

Formalism	Ref.	Year	#Citations
Fault Tree/Attack Tree Integration (FT/AT)	[18]	2009	170
Component Fault Trees (CFTs)	[46]	2013	52
Attack-Fault Trees (AFTs)	[33]	2017	56
State/Event Fault Trees (SEFTs)	[42]	2013	25
Boolean Driven Markov Processes (BDMPs)	[31]	2014	36
Attack Tree Bow-ties (ATBTs)	[5]	2017	9
STAMP	[20]	2017	120
SysML	[40]	2011	72
Architectural Analysis and Design Language (AADL)	[16]	2020	0
Bayesian Networks (BNs)	[30]	2015	46

1. Plain combinations of attack trees and fault trees. These formalisms combine attack trees and fault trees without adding additional constructs: fault tree/attack trees (FT/AT) [18], which investigate how the a basic event of a FT can be triggered by an attacker, refining these with ATs with the event in question as goal, component fault trees (CFTs) [46] merge attack trees and fault trees without any restrictions, Attack-Fault Trees (AFTs) [33] merge dynamic attack trees and dynamic fault trees.

2. Extensions of attack trees-fault tree combinations. These merge attack trees, fault trees with additional constructs: State/Event Fault Trees (SEFTs) [42] exploit Petri nets to refine the basic attack steps in an attack tree and the basic component failures in a fault tree. The Petri nets can for instance model that the attack and failure behavior is different depending whether a door is open or closed. Boolean Driven Markov Processes (BDMPs) [31] extend attack trees and fault trees with both Petri nets and triggers. The latter model sequential behaviour, where one fault or attack triggers another one. Finally, Attack Tree Bow-ties (ATBTs) [5], extend bowties [37] with attack trees, where bowties themselves combine fault trees with event trees.

3. Architectural formalisms and bayesian networks. Apart from combinations of attack trees and fault trees, we identified a third category, containing formalisms that take as a starting point the system architecture: The Systems-Theoretic Accident Model and Processes (STAMP) [20], is an accident causality model, rooted in the observation that system risks do not come from component failures, but rather from inadequate control or enforcement of safety and security constraints. Systems-Theoretic Process Analysis then systematically identifies the consequences of incorrect control and feedback actions, e.g., when these happen too early, in the wrong order, or were maliciously inserted.

SysML-sec [43] extend the SysML modeling framework with safety and security requirements, which can be checked using model checkers. In particular, SysML-sec supports the modelling of communication channels between processes with the encryption methods and their complexity overhead.

The Architectural Analysis and Design Language (AADL) [16] enables safety analysis, via the AADL error model, and security analysis via the AADL LAMP extension. In this way the same AADL model can be separately analyzed to investigate safety and security properties.

Finally, albeit somewhat artificially, we also put Bayesian Networks (BNs) for safety-security analysis in this category [30]. This model allows to represent probabilistic dependencies between several variables via a directed acyclic graph. BNs are used to model safety and security dependencies. The two root nodes represent system safety and security. BNs can analyze which nodes influence other nodes and how (conditional independence analysis) calculate reliability metrics.

3.2 Findings

The analysis we performed highlighted some notable findings, summarized in Table 2. For each analyzed formalism we highlight the dependencies it captures, its modeling constructs, the analysis types it enables, case studies that were performed deploying this formalism and possible tools.

Table 2. Comparison of safety-security formalisms. A = Antagonism, CD = Conditional Dependency, MR = Mutual reinforcement, I = Independence. $*$ = capable when NOT-gate is supported. $\rightarrow$ = capable but only directional from security to safety.

Formalism	Dependencies				Modelling	Analysis		Application	Tool
	A	CD	MR	I		QL	QT		
FT/AT	$*$	$\rightarrow$		x	ATs refine FT leaves	x	x	Chemical plant	
CFTs	$*$	x	x	x	Merge ATs+FTs	x	x	Cruise control	
AFTs	x	x	x	x	Merge dynamic ATs+FTs	x	x	Pipeline, lock door	UPPAAL
SEFTs	x	$\rightarrow$	x	x	FTs+Petri nets	x	x	Tyre pressure, lock door	ESSaRel
BDMPs	x	x	x	x	Triggers, Petri nets	x	x	Pipeline, lock door	KB3, Figaro
ATBTs	$*$	$\rightarrow$		x	Bowties+FT/AT	x	x	Pipeline, Stuxnet	
STAMP				x	Process controller	x		Synchronous-islanding	
SysML					System components	x		Embedded systems	TTool
AADL				x	System components + ports	x		Lock door	Cheddar, Marzhin
BNs	x	x	x	x	Conditional prob	x	x	Pipeline	MSBNx

Finding #1: The majority of approaches combine attack trees and fault trees. As stated, six out of the ten formalisms combine attack trees and fault trees. This is not surprising, since FTs and ATs are well established model-based formalisms, extensively used both in industry and academia.

Finding #2: No novel modeling constructs are introduced. Despite the shown combinations and extensions, existing safety-security co-analysis do not introduce novel modeling constructs to capture safety-security interactions. They do merge existing safety and security formalisms, however they do not add new operators. As such, they are suited to represent safety and security features in one model, but do not seem to appropriately capture the interaction between them.

Finding #3: Safety-security interactions are still ill-understood. In spite of the definitions provided in [32], we are convinced that safety-security interactions can still be defined more thoroughly by adopting more rigour and by focusing on requirements and events. In particular, it is not so clear to which entities the safety-security interactions refer: do these concern safety-security requirements, measures, or something else? Clarifying their definitions is a prerequisite for properly modeling safety-security interactions in a mathematical formalism.

Finding #4: No novel metrics were proposed. Analyzed formalisms adopt classic metrics, such as mean time to failure and attacker success probabilities. However, none of them introduce new metrics to quantify safety-security dependencies or to analyze trade-offs, e.g., through Pareto analysis. New metrics and trade offs are paramount to understand the interaction between safety and security aspects.

Finding #5: No large case studies were carried out. To the best of our knowledge, no large case studies were carried out. The majority of analyzed papers present small examples used to showcase the formalism in question. Some notable exceptions are [31] and [33]: here, the medium-sized example of a pipeline is presented. However for safety and security, when considered separately, large case studies do exist [7].

Finding #6: Different formalisms model different safety-security interactions. As shown in Table 2, only AFTs, BDMPs and BNs can unconditionally model all four dependencies between safety and security. CFTs and SEFTs can model them provided with extensions/with some limitations.

Research gaps.

- Realistic *large-sized case studies* concerning safety-security interactions are still missing. Performing large-sized case study analysis would contribute to further address additional gaps:
- It would clarify the nature of *safety-security interactions*, that are currently still ill-understood. Furthermore, it would help improve standard definitions for safety-security interdependencies;
- From this understanding, *novel metrics* and *novel modeling constructs* for safety-security co-analysis - that are still missing - could be developed.

4 Attack Trees Versus Fault Trees

We saw that most safety-security formalisms combine attack trees and fault trees. This is a natural step, since attack trees and fault trees bear many similarities. What is less known, is that they also feature a number of remarkable differences, elaborated in [10]. In particular, the interpretation of the OR-gate is crucially different in attack trees than in fault trees, and therefore their analysis should not be mindlessly combined. Below, we present the most remarkable similarities and differences, summarized in Table 3.

Table 3. Differences between attack and fault trees

Syntax	Attack trees	Fault trees
Leaves	Basic attack steps (BAS)	Basic events (BEs)
Non-leaves	Subgoals	Intermediate events (IEs)
Static gates	AND, OR	AND, OR, VOT
Dynamic gates	SAND	SPARE, FDEP, PAND
Other extensions	Defenses	Repairs, maintenance
Analysis		
Qualitative	(Minimal) attack vectors/scenarios	(Minimal) cut sets
Attributes	probability, cost, time, skill, impact	probability
Metrics	Min cost, time, skill	Reliability, availability,
	Max impact, probability	MTTF, MTBF
Semantics		
Qualitative	Structure function	Structure function
Stochastic		
AND(a,b)	$p_a \cdot p_b$	$p_a \cdot p_b$
OR(a,b)	$\min(p_a, p_a)$	$p_a + p_b - p_a \cdot p_b$

4.1 Attack Trees Versus Fault Trees

It is no surprise that ATs and FTs are similar to each other, since ATs were inspired by FTs. FTs were introduced in 1961 at Bell Labs to study a ballistic missile [13,45,47]. Weiss introduced threat logic trees—the origin of ATs—in 1991, and its "similarity... to fault trees suggests that graph-based security modelling has its roots in safety modelling" [28].

Attack trees and fault trees come in various variants and extensions. Following [9], we categorize these along two axes. First, we distinguish between *static* and *dynamic* trees. Static attack and fault trees are equipped with Boolean gates. Dynamic trees come with additional gates to model time-dependent behavior.

Second, we distinguish between *tree-shaped* and *DAG-shaped* trees. Trees are relatively easy to analyse via a bottom up algorithm. This algorithms works for all quantitative attributes (cost, time, probability) as long as they constitute an attribute domain.

4.2 The Static Case

Syntax. The basic variants, called *static* or *standard* fault and attack trees, have the exact same syntax: trees or DAGs equipped with AND and OR gates. Fault trees often contain a (k, m) voting gate, which fails if k out of the n inputs fail; these can however be expressed in terms of AND and OR gates. We use the word *disruption tree* (DT) for either an attack tree or a fault tree.

Formally, DTs are rooted DAGs with typed nodes, for which we consider types $\mathbb{T} = \{\texttt{LEAF}, \texttt{OR}, \texttt{AND}\}$. For Booleans we use $\mathbb{B} = \{1, 0\}$. The edges of a DT are given by a function *ch* that assigns to each node its (possibly empty) sequence of children. We use set notation for sequences, e.g. $e \in (e_1, \ldots, e_m)$ means $\exists i.\, e_i = e$, and we denote the empty sequence by ε.

Definition 1. *A* disruption tree *is a tuple* $T = (N, t, ch)$ *where:*

- N *is a finite set of* nodes*;*
- $t\colon N \to \mathbb{T}$ *gives the* type *of each node;*
- $ch\colon N \to N^*$ *gives the sequence of* children *of a node.*

Moreover, T *satisfies the following constraints:*

- (N, E) *is a connected* DAG*, where* $E = \big\{(v, u) \in N^2 \mid u \in ch(v)\big\}$*;*
- T *has a unique root, denoted* R_T*:* $\exists!\, R_T \in N.\ \forall v \in N.\ R_T \notin ch(v)$*;*
- LEAF_T *nodes are the leaves of* T*:* $\forall v \in N.\ t(v) = \texttt{LEAF} \Leftrightarrow ch(v) = \varepsilon$.

4.3 Semantics

Semantics pin down the mathematical meaning for attack and fault trees. The semantics of both fault trees and attack trees is given in terms of their *structure function*, indicating which sets of leaves cause the top level events to happen.

Thus, the *structure function* of a disruption tree T is a function $f_T\colon \mathbb{B}^n \to \mathbb{B}$. Technically, a status vector $\boldsymbol{v} = \langle v_1, \ldots, v_n \rangle$ indicates for each leaf i whether it was disrupted, i.e., $v_i = 1$ if leaf i has failed or was attacked. Then $f_T(\boldsymbol{v}) \in \{0, 1\}$ indicates whether the whole system was disrupted. This function can be defined recursively in the nodes of T: $f_T(v, A)$ tells whether $A \subseteq \text{LEAF}$ suffices to disrupt node $v \in N$ of T, where A encodes $\boldsymbol{v}$ as usual.

Definition 2. *The* structure function $f_T\colon N \times 2^{\text{LEAF}} \to \mathbb{B}$ *of a disruption tree* T *is given by:*

$$f_T(v, A) = \begin{cases} 1 & \text{if } t(v) = \texttt{OR} \quad \text{and } \exists u \in ch(v).f_T(u, A) = 1, \\ 1 & \text{if } t(v) = \texttt{AND} \quad \text{and } \forall u \in ch(v).f_T(u, A) = 1, \\ 1 & \text{if } t(v) = \texttt{LEAF} \text{ and } v \in A, \\ 0 & \text{otherwise.} \end{cases}$$

The structure function can be used to asses suites: a *disruption suite* $\mathcal{S} \subseteq 2^{\text{LEAF}}$ represents all ways in which the system can be compromised. From those, one is interested in disruptions $A \in \mathcal{S}$ that actually represent a threat. These correspond to (minimal) cut sets in fault trees and attack scenarios in attack trees. To find them, we let $f_T(A) \doteq f_T(R_T, A)$ and call disruption A *successful* if $f_T(A) = 1$, i.e. it makes the top-level of T succeed (resp. be attacked or failed). If, moreover, no proper subset of A is successful then A is a *minimal disruption*.

It is well known that attack trees and fault trees are *coherent* [4], meaning that adding attack steps/basic events preserves success: if A is causes the TLE to happen, then so is $A \cup \{a\}$ for any $a \in \text{LEAF}$. Thus, the suite of successful disruptions of an DT is characterised by its minimal disruptions.

Definition 3. *The* semantics of a static DT T *is its suite of minimal disruptions:* $[\![T]\!] = \left\{ A \subseteq 2^{\text{LEAF}} \mid f_T(A) \wedge A \text{ is minimal} \right\}$.

4.4 Metrics for Attack and Fault Trees

Dependability metrics quantify key performance indicators (KPIs), that quantify several dependability characteristics of a system. Such metrics serve several purposes, e.g. allowing to compare different design alternatives w.r.t. the desired dependability features; computing the effectiveness of measures; verifying whether a solution meets its dependability requirements; etc.

Metrics for attack trees focus on a wide variety of attributes, such as the cost of an attack, its time and, success probability. These can be conveniently summarized via an attribute domain [35]. Metrics for fault trees focus on probabilistic aspects, such as the system reliability (i.e. the probability that a system fails within its mission time T), the availability (i.e., the average percentage of time that the system is up), mean time to failure, etc.

Attribute Metrics. We define dependability metrics for DTs in three steps: first an attribution α enrich the leaves with attributes, assigning a value to each $a \in \text{LEAF}$, then a dependability metric $\widehat{\alpha}$ assigns a value to each disruption scenario A; and finally the metric $\breve{\alpha}$ assigns a value to each disruption suite.

Example 1. Consider the AT in Fig. 1b. The metric we study is the time required to execute a successful attack. Thus, the attribution α equips each AT leaf with its attack time, setting e.g. $\alpha(\text{Attacker forces door}) = 5$, $\alpha(\text{Door unlocked}) = 0$ and $\alpha(\text{Attacker tries door}) = 2$. If all attack steps are executed sequentially, then the time needed to execute an attack $A = \{a_1, \dots a_n\}$ is sum of the attack times of the BASs:

$$\widehat{\alpha}(a_1, \dots a_n) = \sum_{i=1}^{n} \alpha(a_i)$$

Both for attackers and defenders of the system, it is relevant to consider the shortest attack in an attack suite $\mathcal{S}$:

$$\breve{\alpha}(\{A_1, \dots A_n\}) = \min\{\widehat{\alpha}(A_1), \dots \widehat{\alpha}(A_n)\}$$

Other metrics give rise to other attribute definitions. For example, the success probability of an attack is the product of the success probabilities of the BAS. The probability of attack suite $\mathcal{S}$ is the probability of the most successful attack in $\mathcal{S}$. The cost of an attack is the sum of the cost of the BASs, and the cost of an attack suite is the minimum cost of its attacks. A formal definition is as follows.

Definition 4. *Given a DT and a set V of values:*

1. *an* attribution $\alpha\colon \text{LEAF} \to V$ *assigns an* attribute value $\alpha(a)$*, or shortly an* attribute*, to each leaf a;*
2. *a* dependability metric *over disruptions is a function $\widehat{\alpha}\colon \mathscr{A}_T \to V$ that assigns a value c to each disruption A;*
3. *a* dependability metric *over disruption suites is a function and to a function $\breve{\alpha}\colon \mathscr{S}_T \to V$ that assigns a value $\breve{\alpha}(\mathcal{S})$ to each disruption suite $\mathcal{S}$.*

We write $\breve{\alpha}(T)$ for $\breve{\alpha}([\![T]\!])$, setting the metric of a DT to the metric of its minimal disruption suites.

Remark 1. We choose the notation $\widehat{\alpha}$ for metrics over disruptions, since $\widehat{\ }$ resembles $\triangle$, and $\widehat{\alpha}(A)$ corresponds to the interpretation of the AND gate. Similarly, $\breve{\alpha}$ resembles $\triangledown$, and corresponds to the OR gate, since $\breve{\alpha}(\mathcal{S})$ often chooses the best disruption set among all $A \in \mathcal{S}$.

Different Metric Interpretation of the OR-Gate. It is important to realize that the quantitative interpretation of the OR-gate is different in attack trees than in fault trees. Fault trees assume that all components work in parallel. Thus, component i fails with probability p_i, the fault tree $OR(C_1, C_2)$ fails with probability $p_1 + p_2 - p_1 \cdot p_2$. In attack trees, the OR-gate works in parallel. The interpretation of the attack tree $OR(C_1, C_2)$ is that the attacker executes either C_1 or C_2. Since the attacker maximizes their success probability, the probability on a successful attack in the tree $OR(C_1, C_2)$ equals $max(p_1, p_2)$.

This distinction is completely ignored in the analysis methods for all six attack-fault combinations/extensions [5,18,31,33,42,46]. In particular, the analysis methods for computing probabilities may not account for the different interpretation of the OR-gates related to safety or security events. This could further lead to incorrect computations of dependability metrics e.g., probability values.

5 Analysis Algorithms for Attack and Fault Trees

Numerous analysis methods for quantitative analysis of attack trees and fault trees exist [2,6,15,23,29,41,45]. In this section, we give an overview of two common algorithms for fault trees and attack trees.

5.1 Algorithms for Tree-Shaped DTs

We first provide the algorithms for tree-structured DTs, where every node in the graph has a single parent. These can be analyzed via a bottom-up algorithm, propagating the values from the bottom to the root of the tree. In order for this procedure to work for all metrics, we combine the inputs of the AND-gate using an operator $\triangle$, and the inputs of the or gate via $\triangledown$. Then this procedure works whenever the algebraic structure $(V, \triangle, \triangledown)$ constitutes a semiring [35].

Next, we treat the computationally more complex DAG-structured DTs. These can be analyzed by converting the DT to a binary decision diagram (BDD). Again, this works if $(V, \triangle, \triangledown)$ is a semiring [9].

Input: Tree-structured DT T,
node v from T,
attribution $\alpha\colon \mathrm{BAS}_T \to V$,
semiring attribute domain
$D = (V, \triangledown, \triangle)$.
Output: Metric value $\breve{\alpha}(T) \in V$
from node v downwards.

```
if t(v) = OR then
    return ⨄_{u∈ch(v)} BU(T, u, α, D)
else if t(v) = AND then
    return ⨅_{u∈ch(v)} BU(T, u, α, D)
else // t(v) = BAS
    return α(v)
```

Algorithm 1: BU for tree DTs

Input: BDD B_T from static DT T,
node w from B_T,
attribution $\alpha\colon \mathrm{BAS}_T \to V$, semiring
attribute domain
$D_* = (V, \triangledown, \triangle, 1_\triangledown, 1_\triangle)$.
Output: Metric value $\breve{\alpha}(T) \in V$ from node
w downwards.

```
if Lab(w) = 0 then
    return 1_▽
else if Lab(w) = 1 then
    return 1_△
else // non-terminal Lab(w) = v ∈ BAS_T
    return BDD(B_T, Low(w), α, D_*) ▽
        (BDD(B_T, High(w), α, D_*) △ α(v))
```

Algorithm 2: BDD for static DAG DTs

Example 2. We illustrate the (straightforward) bottom up algorithm via the attack tree in Fig. 2. We compute the time required to reach the top event, with the same attribute values as before: Abbreviating $f =$ Attacker forces door, $u =$ Door unlocked and $t =$ Attacker tries door, we have $\alpha(f) = 5$, $\alpha(u) = 0$ and $\alpha(t) = 2$. The bottom up computation first computes the time required to achieve the subgoal "Attacker walks through door", abbreviated as w. Since the attack time metric interprets the AND-gate as the sum, we take $\triangle = +$ and obtain the value for w as the sum of the metric values of its children "Attacker forces door" and "Door unlocked", abbreviated u and f respectively.

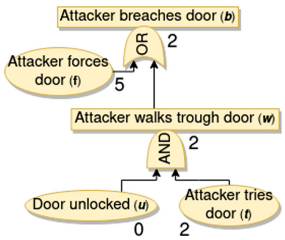

Fig. 2. Algorithm 1 for min. attack time.

$$\breve{\alpha}(w) = \breve{\alpha}(f) \triangle \breve{\alpha}(u) = \alpha(f) \triangle \alpha(u) = 0 + 2 = 2.$$

Similarly, we compute the time required for the TLA "Attacker breaches door", abbreviated as b. Since the attack time metric interprets the OR-gate as

the minimum, we take $\nabla = \min$ and obtain the value for b as the minimum of the metric values of its children f and w:

$$\breve{\alpha}(b) = \breve{\alpha}(f) \nabla \breve{\alpha}(u) = \min(\alpha(f), \alpha(u)) = \min(5, 2) = 2.$$

The above procedure, formalized in Algorithm 1, works whenever the structure $(V, \nabla, \triangle)$ constitute an attribute domain. Note that this algorithm is linear in the number of DT nodes.

Definition 5. *Let V be a set:*

1. *an* attribute domain *over V is a tuple $D = (V, \nabla, \triangle)$, whose* disjunctive operator $\nabla\colon V^2 \to V$, *and* conjunctive operator $\triangle\colon V^2 \to V$, *are associative and commutative;*
2. *the attribute domain is a* semiring[1] *if $\triangle$ distributes over ∇, i.e. $\forall x, y, z \in V.\ \triangle\,(y \nabla z) = (x \triangle y) \nabla (x \triangle z)$;*
3. *let T be a static DT and α an attribution on V. The* metric for T *associated to α and D is given by:*

$$\breve{\alpha}(T) = \underbrace{\bigtriangledown_{A \in [\![T]\!]}}_{\breve{\alpha}} \underbrace{\bigtriangleup_{a \in A}}_{\widehat{\alpha}} \alpha(a).$$

5.2 Algorithms for DAG-Shaped DTs

DTs with shared subtrees cannot be analysed via a bottom-up procedure on their DAG structure. This is a classical result from fault tree analysis [34]. Intuitively, the problem is that a visit to node v in any bottom-up procedure that operates on the DT structure can only aggregate information on its descendants.

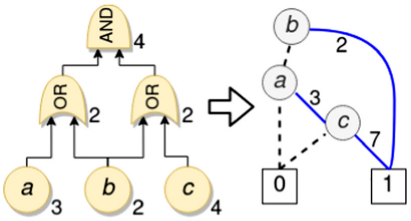

Fig. 3. DAG-shaped AT (left) and its BDD (right).

This is illustrated by the DAG-shaped AT in Fig. 3: We assign attack time to the leaves: $\alpha(a) = 3$, $\alpha(b) = 2$ and $\alpha(c) = 4$. Then the bottom up algorithm yields the following results: for the OR-gates, we take the minimum value between the children, which both equal 2, and for the AND-gate we sum the values of the children, resulting in 4. However, this computation does not take the sharing of b into account. In fact, the shortest attack is to take the BAS b, which takes time 2.

As a matter of fact, computing metrics in a DAG-structured DT T is an NP-complete problem. Various methods to analyse DAG-structured DTs have

[1] Since we require $\triangle$ to be commutative, D is in fact a commutative semiring. Further, rings often include a neutral element for disjunction and an absorbing element for conjunction, but these are not needed in Definition 5.

been proposed: contributions over the last 15 years include [2,6,15,23,29]. We now detail our recent work on binary decision diagrams (BDDs) [9].

BDD Algorithms. BDDs offer a very compact representation of Boolean functions, and can therefore represent the structure function of a DT T: Each BDD node v is labeled with a leaf a of T, and has two children: its left node v_L (reached via a dashed line) represents the structure function of T in case a has the value 0; its right child v_R (reached via a solid line) represents the structure function of T in case a evaluates to 1. The key insight in [9] is that the values of an attribute domain can be computed recursively over this BDD, thereby avoiding duplication of values as in Fig. 3. The idea is as follows. The value for the BDD terminal node labeled with 0 is set to the constant $1_\nabla \in V$; the BDD terminal node labeled with 1 is set to $1_\triangle \in V$. For an internal node v with children v_L and v_R, we proceed as follows: When choosing v_R, i.e. the basic event a occurs, we extend the value computed at v_R with the attribute value of a. We do so via the $\triangle$ operator, since taking the right child corresponds to executing both a and all leaves needed in $\alpha(v_R)$. If a is not executed, then we do not incur the value of $\alpha(a)$, and only take $\breve{\alpha}(v_L)$. Now, the best disruption (i.e. attack or cut set) is obtained by choosing the best option, by deploying the ∇ operator: either one does not execute a, incurring $\breve{\alpha}(v_L)$, or executes a and incurs $\alpha(a) \triangle \breve{\alpha}(v_R)$. This yields the value $\breve{\alpha}(v) = \breve{\alpha}(v_L) \nabla (\alpha(a) \triangle \breve{\alpha}(v_R))$. This is illustrated in (Fig. 3, in blue). As we can see, the TLE can fail either by the failure of b in 2 time units or by the failure of a and c but not of b, in 7 time units. Algorithm 2 shows the pseudocode of this algorithm. The algorithm is linear in the size of the BDD, but that the BDD size can be exponential in the size of the DT. In particular, the BDD size heavily depends on the order for the variables. In practice good heuristics are available, making BDD-computations efficient in practice.

5.3 Research Gaps

Research gaps.

- *An overarching formalism* is still missing. Since attack trees and fault trees interpret the OR-gate in a different manner, proper combinations must feature two variants of the OR-gate: one that coincides with the AT-interpretation and for the FT-variant.
- Another research gap concerns *proper analysis of OR-gates.* Analysis algorithms must handle both the aforementioned variants of OR-gates. Section 5 partially solves this problem for the case of tree-structured attack-fault trees. Efficient analysis of DAG-structured attack-fault tree combinations remains an open problem.

6 Conclusions

Conclusion. Safety and security interactions have been identified as important topics in complex systems, and multiple modeling methods have been developed in an attempt to account for their interactions. Our preliminary results show that most of these methods are based on extending and/or combining existing safety and security modeling methods. No specific metrics or novel modeling constructs are introduced. The majority of considered formalisms combine/extend attack trees and fault trees. As a consequential next step, we performed a thorough analysis of similarities and differences between ATs and FTs. Their static variants - SATs and SFTs - share the same syntax: we group them under the label of disruption trees (DTs), for which we provide shared semantics. Furthermore, we show how to compute dependability metrics on DTs highlighting differences between ATs and FTs when needed, e.g., the different interpretation of the OR-gate. Finally, we propose analysis algorithms for ATs and FTs both for their tree-shaped and DAG-shaped variants.

Future Work. While addressing some of the research gaps, this work also highlights future challenges. With the growing need for safety-security co-analysis, the urge of a better understanding of safety-security interactions arises:

Open problems.

- To foster this understanding, realistic *large-sized case study analysis* concerning safety-security interactions should be performed.
- This would clarify the nature of *safety-security interactions* - that are still ill-understood - and help *improve standard definitions* for safety-security interdependencies.
- As mentioned, *novel metrics* and *novel modeling constructs* for safety-security co-analysis - that are still missing - could then be developed, alongisde an overarching formalism.
- This *overarching formalism* would need to account for two different OR-gates: one that coincides with the AT-variant and one for the FT-variant.
- Furthermore, *proper analysis of OR-gates* has to be performed, as analysis algorithms must handle both the aforementioned variants of OR-gates.

References

1. Amorim, T., Schneider, D., Nguyen, V.Y., Schmittner, C., Schoitsch, E.: Five major reasons why safety and security haven't married (yet). ERCIM News **102**, 16–17 (2015)
2. Arnold, F., Guck, D., Kumar, R., Stoelinga, M.: Sequential and parallel attack tree modelling. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 291–299. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_25

3. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Sec. Comput. **1**, 11–33 (2004)
4. Barlow, R.E., Proschan, F.: Statistical Theory of Reliability and Life Testing: Probability Models. International Series in Decision Processes. Holt, Rinehart and Winston, New York (1975)
5. Bernsmed, K., Frøystad, C., Meland, P.H., Nesheim, D.A., Rødseth, Ø.J.: Visualizing cyber security risks with bow-tie diagrams. In: Liu, P., Mauw, S., Stølen, K. (eds.) GraMSec 2017. LNCS, vol. 10744, pp. 38–56. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74860-3_3
6. Bobbio, A., Egidi, L., Terruggia, R.: A methodology for qualitative/quantitative analysis of weighted attack trees. IFAC **46**(22), 133–138 (2013)
7. Bozzano, M., et al.: A model checker for AADL. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 562–565. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_48
8. Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., Cleven, A.: Reconstructing the giant: on the importance of rigour in documenting the literature search process. In: ECIS (2009)
9. Budde, C.E., Stoelinga, M.: Efficient algorithms for quantitative attack tree analysis. In: CSF, pp. 501–515. IEEE Computer Society (2021). ISSN: 2374-8303. https://doi.org/10.1109/CSF51468.2021.00041
10. Budde, C.E., Kolb, C., Stoelinga, M.: Attack trees vs. fault trees: two sides of the same coin from different currencies. In: QEST (to appear)
11. Byres, E.J., Franz, M., Miller, D.: The use of attack trees in assessing vulnerabilities in SCADA systems. In: Proceedings of the International Infrastructure Survivability Workshop, pp. 3–10. Citeseer (2004)
12. Chockalingam, S., Hadžiosmanović, D., Pieters, W., Teixeira, A., van Gelder, P.: Integrated safety and security risk assessment methods: a survey of key characteristics and applications. In: Havarneanu, G., Setola, R., Nassopoulos, H., Wolthusen, S. (eds.) CRITIS 2016. LNCS, vol. 10242, pp. 50–62. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71368-7_5
13. Clifton, E., et al.: Fault tree analysis-a history. In: Proceedings of the 17th International Systems Safety Conference, pp. 1–9 (1999)
14. Commission, I.E., et al.: IEC 61025: Fault tree analysis. IEC Standards (2006)
15. Dalton, G.C., Mills, R.F., Colombi, Raines, R.A.: Analyzing attack trees using generalized stochastic Petri nets. In: 2006 IEEE Information Assurance Workshop, pp. 116–123 (2006)
16. Dissaux, P., Singhoff, F., Lemarchand, L., Tran, H., Atchadam, I.: Combined real-time, safety and security model analysis. In: ERTSS (2020)
17. Dugan, J.B., Bavuso, S.J., Boyd, M.A.: Dynamic fault-tree models for fault-tolerant computer systems. IEEE Trans. Reliab. **41**(3), 363–377 (1992)
18. Fovino, I.N., Masera, M., De Cian, A.: Integrating cyber attacks within fault trees. Reliab. Eng. Syst. Saf. **94**(9), 1394–1402 (2009)
19. Fraile, M., Ford, M., Gadyatskaya, O., Kumar, R., Stoelinga, M., Trujillo-Rasua, R.: Using attack-defense trees to analyze threats and countermeasures in an ATM: a case study. In: Horkoff, J., Jeusfeld, M.A., Persson, A. (eds.) PoEM 2016. LNBIP, vol. 267, pp. 326–334. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48393-1_24
20. Friedberg, I., McLaughlin, K., Smith, P., Laverty, D., Sezer, S.: STPA-SafeSec: Safety and security analysis for cyber-physical systems. J. Inf. Secur. Appl. **34**, 183–196 (2017)

21. ISO/IEC 25010:2011, S., software engineering: Systems and software quality requirements and evaluation (square). System and software quality models (2011)
22. Junges, S., Guck, D., Katoen, J., Stoelinga, M.: Uncovering dynamic fault trees. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 299–310 (2016)
23. Jürgenson, A., Willemson, J.: Computing exact outcomes of multi-parameter attack trees. In: Meersman, R., Tari, Z. (eds.) OTM 2008. LNCS, vol. 5332, pp. 1036–1051. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88873-4_8
24. Kahneman, D.: A perspective on judgment and choice: mapping bounded rationality. Am. Psychol. **58**(9), 697 (2003)
25. Kimelman, D., Kimelman, M., Mandelin, D., Yellin, D.M.: Bayesian approaches to matching architectural diagrams. Trans. Software Eng. **36**(2), 248–274 (2010)
26. Kolb, C., Nicoletti, S.M., Peppelman, M., Stoelinga, M.: Model-based safety and security co-analysis: a survey. In: arXiv (2021)
27. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Attack-defense trees. J. Logic Comput. **24**(1), 55–87 (2012)
28. Kordy, B., Pièvre-Cambacédès, L., Schweitzer, P.: DAG-based attack and defense modeling: don't miss the forest for the attack trees. Comput. Sci. Rev. **13–14**, 1–38 (2014)
29. Kordy, B., Wideł, W.: On quantitative analysis of attack–defense trees with repeated labels. In: Bauer, L., Küsters, R. (eds.) POST 2018. LNCS, vol. 10804, pp. 325–346. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89722-6_14
30. Kornecki, A.J., Subramanian, N., Zalewski, J.: Studying interrelationships of safety and security for software assurance in cyber-physical systems: approach based on Bayesian belief networks. In: 2013 FedCSIS, pp. 1393–1399. IEEE (2013)
31. Kriaa, S., Bouissou, M., Colin, F., Halgand, Y., Pietre-Cambacedes, L.: Safety and security interactions modeling using the BDMP Formalism: case study of a pipeline. In: Bondavalli, A., Di Giandomenico, F. (eds.) SAFECOMP 2014. LNCS, vol. 8666, pp. 326–341. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10506-2_22
32. Kriaa, S., Pietre-Cambacedes, L., Bouissou, M., Halgand, Y.: A survey of approaches combining safety and security for industrial control systems. Reliab. Eng. Syst. Saf. **139**, 156–178 (2015)
33. Kumar, R., Stoelinga, M.: Quantitative security and safety analysis with attack-fault trees. In: 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), pp. 25–32 (2017)
34. Lee, W., Grosh, D., Tillman, F., Lie, C.: Fault tree analysis, methods, and applications: a review. IEEE Trans. Reliab. **R-34**(3), 194–203 (1985)
35. 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
36. Nicol, D.M., H.Sanders, W., Trivedi, K.S.: Model-based evaluation: From dependability to security. IEEE Trans. Dep. Sec. Comput. **1**(1), 48–65 (2004)
37. Nielsen, D.S.: The Cause/Consequence Diagram Method as a Basis for Quantitative Accident Analysis. Risø National Laboratory (1971)
38. Nigam, V., Pretschner, A., Ruess, H.: Model-based safety and security engineering (2019)
39. Organization, I.S.: ISO/dis 26262: Road vehicles, functional safety. Technical report (2009)

40. Pedroza, G., Apvrille, L., Knorreck, D.: AVATAR: A SysML environment for the formal verification of safety and security properties. In: 2011 NOTERE, pp. 1–10. IEEE (2011)
41. Rauzy, A.: New algorithms for fault trees analysis. Reliab. Eng. Syst. Saf. **40**(3), 203–211 (1993)
42. Roth, M., Liggesmeyer, P.: Modeling and analysis of safety-critical cyber physical systems using state/event fault trees. In: SAFECOMP 2013 (2013)
43. Roudier, Y., Apvrille, L.: SysML-Sec: A model driven approach for designing safe and secure systems. In: MODELSWARD
44. Ruijters, E., Guck, D., Drolenga, P., Stoelinga, M.: Fault maintenance trees: reliability centered maintenance via statistical model checking. In: 2016 Annual Reliability and Maintainability Symposium (RAMS), pp. 1–6 (2016)
45. 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)
46. Steiner, M., Liggesmeyer, P.: Combination of safety and security analysis - finding security problems that threaten the safety of a system (2016)
47. Watson, H.: Launch control safety study. Technical Report, Section VII, Vol. 1. Bell Labs (1961)
48. Zampino, E.J.: Application of fault-tree analysis to troubleshooting the NASA GRC icing research tunnel. In: Annual Reliability and Maintainability Symposium, 2001 Proceedings, pp. 16–22 (2001)