

Causal Model Extraction from Attack Trees to Attribute Malicious Insiders Attacks

Amjad Ibrahim¹, Simon Rehwald¹, Antoine Scemama², Florian Andres¹, and Alexander Pretschner¹

¹ Department of Informatics, Technical University of Munich, Garching b. Munich, Germany {ibrahim, rehwald, andres, pretschn}@cs.tum.edu

² Brainloop AG (Munich) antoine.scemama@brainloop.com

Abstract In the context of insiders, preventive security measures have a high likelihood of failing because insiders ought to have sufficient privileges to perform their jobs. Instead, in this paper, we propose to treat the insider threat by a detective measure that holds an insider accountable in case of violations. However, to enable accountability, we need to create causal models that support reasoning about the causality of a violation. Current security models (e.g., attack trees) do not allow that. However, they are a useful source for creating causal models. In this paper, we discuss the value added by causal models in the security context. Then, we capture the interaction between attack trees and causal models by proposing an approach to extract the latter from the former. Our approach considers insider-specific attack classes like collusion attacks and causal-modeling-specific properties like preemption relations. We present an evaluation of the correctness of the product, i.e., the resulting causal models and the efficiency of the process, i.e., the extraction approach.

1 Introduction

Security and privacy are crucial in systems that deal with *sensitive customer assets* (e.g., trade secrets). Adversaries are constantly trying to compromise the integrity, confidentiality, or availability of such assets. These attempts are carried out by *insiders* or *outsiders* of the system. In this paper, we are chiefly interested in *insiders*, specifically *malicious insiders* such as a rogue employee. In fact, according to the Cyber Security Intelligence Index by IBM X-Force Research [28], insiders carried out 60% of all attacks in 2015.

Insiders are mostly not malicious. Typically, there is a trust base between a company and its employees, not to mention the legal contracts an employee signs upon starting a job. However, an insider can, among other things, tamper with records in the database, leak or delete documents. Consequently, such acts may lead to *reputation damage, legal costs, and reimbursements* [12]. Reports show that *insiders* carry out the most significant, costly and lengthy attacks [28,12]. Such attacks are likely to succeed, and their impact is significant [30].

In the context of insiders, preventive measures have a high likelihood of failing because insiders ought to have sufficient privileges for their jobs. They may abuse

their privileges. The term “abuse” makes this problem especially hard due to the unpredictable nature of insiders and the necessity of their privileges.

We propose addressing the insider threat using a detective approach [35] that helps a company to attribute malicious acts. Detective approaches, such as *accountability*, provide a mechanism to answer questions about security incidents (e.g., “why was the document leaked?”) and attribute responsible parties a posteriori. Attack attribution is the process of identifying the perpetrator of a cyber-attack [21]. This mechanism increases forensic readiness [29,33], and establishes the basis for taking legal action against an attacker [9]. As such, attribution can be considered in many cases as a deterrent measure [9,32].

Insider attack attribution does not inherit the challenges facing attribution such as tools prepositioning [36], and the anonymity of the Internet [9]. Still, there are no robust approaches to attribute insider attacks. Attack attribution surveys [36,9,32] show that most of the attribution literature focused on the IP level in network attacks, which is still inconclusive. Instead, in this paper, we tackle insider attacks attribution through an automated reasoning capability.

Accountability, fundamentally, means preserving evidence and supporting reasoning about the causal relationships [7] within the collected evidence. Actual causality, as an essential ingredient for accountability, was studied in different fields of computer science; however, it was not really utilized in security [13]. For that, we mainly adopt the definition of Halpern and Pearl (HP) [6,7] to infer *actual causality*. HP is formal foundation to answer causal queries in a way that matches the human way of thinking. This enables us to explain insider attacks. However, the first challenge towards this adoption is the creation of the *causal models* which are required by HP. We propose to solve this problem by relating causal models to security models such as attack trees [31].

Graphical security models [16] such as attack trees (AT) [31] are appealing to scientists for their formal syntax and semantics [20,27], to managers for their visual nature, and to engineers for their systematic categorization of threats. Thus, an AT is used for the purposes of risk estimation, cost approximation, and defense planning. We aim to add forensics analysts to the list of AT beneficiaries, and supporting causality inference to the list of purposes. However, ATs are not readily sufficient to be used for after-the-fact forensic analysis.

Our goal is to create models that attribute blame to a human, i.e., an insider. However, an AT does not usually include potential attackers (suspects). This is what differentiates ATs from causal models. Thus, we analyze the implications of adding suspects to ATs. Then, we detail a complete approach to extract causal models from AT and show their utilization to infer causality automatically [11].

We focus on models for insider threat for two reasons. Firstly, because we think that accountability is a deterrent measure against insider threats. Secondly, while creating those models, we can exploit the unique property of insider threats, i.e., the ability to identify *suspects* beforehand. Our contributions are: **a)** A proposal of utilizing causal models in insider threat attribution and forensics. This is supported by a discussion of the usefulness of such models in the context of insiders. **b)** An extraction approach of causal models from attack trees. This

transformation enables *attributing suspects, creating exogenous variables, recommending preemption relations, and generating ready-to-analyze models*. **c)** An open-source tool (*ATCM*) that implements the approach with an evaluation of the efficiency, the validity of the approach, and the effectiveness of the model.

2 Preliminaries

We review the formalism of attack trees in Section 2.1, we elaborate on the foundations of *causality* in Section 2.2, with an example in Section 2.3.

2.1 Foundations of Attack Trees

AT [31] model potential security threats within a system and the steps necessary to perform an attack. The root node contains the ultimate goal of an attack tree while the sub-nodes describe activities that are necessary to conduct the respective parent activity/goal. The relationship between a node and its children can be either *OR* or *AND* (represented by a circular line below the node).

Depending on the required purpose, attack trees have been defined using different semantics such as multi-set semantics [20], linear-logic semantics [8], timed automata [17], Markov decision process [2], and propositional logic [27]. In this paper, we aim to reason about the actual causality relations among binary events, i.e., whether the occurrence or absence of a specific event was the cause of another event. Hence, we use the equation-propositional semantics similar to [27]. Such formalism is simple, expressive, and general. The main difference between our definition and the definition in [27] is that we create a propositional formula for each node in the tree (excluding the leaves), while the whole tree is represented with a minimized formula of the root in [27].

For the formal definition, we follow Mauw and Oostdijk’s [20] way of defining an attack tree. However, we adapt it to use propositional logic semantics. Formally, Definition 1 expresses attack trees.

Definition 1. *Attack Tree* is a 4-tuple $AT = (\mathcal{N}, \rightarrow, n_0, [[n]])$ where

- \mathcal{N} is a finite set of attack nodes
- $n_0 \in \mathcal{N}$ is the root node
- $\rightarrow \subseteq \mathcal{N} \times \mathcal{N}$ is a finite set of acyclic relations.
- $[[n]]$ is a function that returns a propositional formula for each $n \in \mathcal{N}$, the formula represents the semantical dependency of a node on its children nodes.

2.2 Actual Causality

HP is the influential formal framework proposed by Joseph Halpern and Judea Pearl [7] to infer actual causality. This framework is based on *counter-factual reasoning* (CF), in which we think of alternative worlds where if the cause is removed, the effect does not occur. Essentially it is a simple but-for test, i.e., but for the existence of some event X , would Y have occurred. The naive CF

reasoning fails to deal with many examples in the literature [19]. To reason about causal relationships, HP uses a structural equations model (SEM). The set of the equations is what HP refers to as a *causal model*. The utilization of structural equations in this context allows for *interventions*. Intervention is the act of changing a value in the model and checking the effects that it has on other values. This concept is beneficial for inferring causality because it allows us to answer the metaphysical counter-factual queries.

Causal Model A causal model [6] (Definition 2) uses variables to describe the world. They take different values to present states of a property of the world, and they have a causal influence on each other, which is modeled by the equations [6]. Each equation represents a mechanism in the modeled world. The variables are classified into *exogenous* and *endogenous* variables. *Exogenous* variables are determined outside the model; they represent the factors that the modeler does not consider as causes, but rather given. In contrast, the *endogenous* variables represent the factors that we consider; their values are determined by the exogenous variables and other endogenous variables within the model.

Definition 2. Causal Model [6] M is 4-tuple $M = (\mathcal{U}, \mathcal{V}, \mathcal{R}, \mathcal{F})$, where

- \mathcal{U} is a set of exogenous variables,
- \mathcal{V} is a set of endogenous variables,
- \mathcal{R} associates with every $Y \in \mathcal{U} \cup \mathcal{V}$ a set $\mathcal{R}(Y)$ of possible values for Y ,
- \mathcal{F} associates with $X \in \mathcal{V}$ $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} \setminus \{X\}} \mathcal{R}(Y)) \rightarrow \mathcal{R}(X)$.

We restrict our focus now to acyclic models in which there is always a unique solution to the equations given the values of the exogenous variables \mathcal{U} . A given set of values for the exogenous variables is called a *context*. A model is illustrated using a graph with the variables from $\mathcal{U} \cup \mathcal{V}$ as nodes. There is an edge from a node X to a node Y , if F_Y depends on X in the causal model.

Reasoning about Causality We start by presenting the formal notations used by HP [6]. They are necessary for the definition of an actual cause which is presented in Definition 3. A sequence of variables X_1, \dots, X_n can be abbreviated as a vector \vec{X} , values of the variables are denoted by small letters x_1, \dots, x_n . Analogously, $X_1 = x_1, \dots, X_n = x_n$ is abbreviated $\vec{X} = \vec{x}$. The values of all exogenous variables \mathcal{U} , also called (actual) *context*, is written as \vec{u} . Variable Y can be set to value y writing $Y \leftarrow y$, i.e., we substitute the equation of Y with a constant. For a causal model $M = (\mathcal{U}, \mathcal{V}, \mathcal{R}, \mathcal{F})$ and a vector \vec{X} of variables in \mathcal{V} , a *submodel* $M_{\vec{X} \leftarrow \vec{x}}$ can be obtained by setting \vec{X} to \vec{x} in all functions \mathcal{F} and removing \vec{X} from \mathcal{V} in the model \mathcal{M} . A *primitive event*, given a model \mathcal{M} , is a formula of the form $X = x$ for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$. A *basic causal formula* is of the form $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k] \varphi$, where φ is a Boolean combination of primitive events. Y_1, \dots, Y_k (abbreviated \vec{Y}) are distinct variables in \mathcal{V} , and $y_i \in \mathcal{R}(Y_i)$, that are intervened on, i.e., their functions are substituted with constants. A causal formula ψ can be evaluated in M given a context \vec{u} . We write $(M, \vec{u}) \models \psi$ if ψ evaluates to true in the causal model M given context \vec{u} . The statement $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ implies that solving the equations

in the submodel $M_{\vec{y} \leftarrow \vec{y}}$ with context \vec{u} yields the value x for variable X . Definition 3 shows the three conditions of an actual cause.

Definition 3. Actual Cause ([6]) $\vec{X} = \vec{x}$ is an actual cause of φ in (M, \vec{u}) if the following three conditions hold:

AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x})$ and $(M, \vec{u}) \models \varphi$.

AC2. There is a set \vec{W} of variables in \mathcal{V} and a setting \vec{x}' of the variables in \vec{X} such that if $(M, \vec{u}) \models \vec{W} = \vec{w}$, then $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}] \neg \varphi$.

AC3. \vec{X} is minimal in satisfying the first two conditions.

AC1 sets a trivial condition: $\vec{X} = \vec{x}$ can only be a cause of φ , if $\vec{X} = \vec{x}$ and φ are true under (M, \vec{u}) . AC2 is the core of the definition. It checks the counterfactual relation between the cause and the effect, i.e., changing the cause $\vec{X} = \vec{x}$ leads to φ changing to $\neg \varphi$. However, the definition allows us to keep variables \vec{W} at their actual value \vec{w} . The tuple $(\vec{W}, \vec{w}, \vec{x}')$ is called a *witness* of $\vec{X} = \vec{x}$ being a cause of φ . AC3 is a minimality condition and ensures that only essential events are part of the cause. In the following, we briefly discuss the benefits of HP specifically to attribute security attacks by re-using the knowledge from AT. Although we are not adding extra knowledge to the models, we argue that presenting the knowledge as a causal model is crucial for postmortem analysis.

Distinguishing between exogenous and endogenous variables, at first sight, does not appear to be revolutionary. However, this distinction enables the choice of what to count as a possible cause (*endogenous*) and what not to (*exogenous*), hence, it treats cases of *irrelevance*. Consequently, it allows us to limit our attribution based on the goal. If we are looking for legal evidence, then we can include possible human actors in the set of endogenous variables. If we are looking for an explanation of an intrusion, then we can include the running services as endogenous variables. Furthermore, HP correctly classifies the *non-occurrence* of events as causes. For example, an administrator “forgetting” to install the latest update of the firmware on a server can be a cause of an exploit.

A typical problem of causality definitions, which HP deals with, is *preemption*. It resembles the confusing cases where several potential causes exist and coincide, but one cause preempts the others. The problem for simple CF definitions is that if the earlier cause A , had not been there, cause B would have triggered the effect anyway (just a bit later). Thus, A is not classified as a cause. HP deals with this by using \vec{W} from Definition 2 and auxiliary variables. Accounting for preemption, in insiders attacks, is beneficial. Specifically, in attacks with different strategies of attacking. For example, an administrator copying a DB backup file, although this is a policy violation, is not the actual cause of the data breach that happened. The copy act was preempted by a privilege abuse of another employee. Further, differentiating actual causes in cases of preemption is crucial when preventive measures such as an intrusion detection system IDS are deployed. For example, an IDS may preempt an attack from succeeding although the basis steps of the attack were carried out.

Conjunction and Disjunction. HP can consider a combination of events as a cause. There are attacks that are carried out by multiple steps, and hence

are modeled using an *AND* gate. For example, to read a service’s memory, an attacker accesses the machine, then attaches a debugger to the running process. On the other hand, there are attacks that can be carried out using different techniques or by exploiting different vulnerabilities. For example, to steal the master key from a system, the attacker can either obtain it decrypted from memory *or* encrypted from the database (the attacker then has to decrypt it). A more interesting scenario would be if two insiders cooperated in performing an attack, i.e., a collusion attack. Such attacks are a major threat class of insiders [14]. We will see in Section 3 that our approach exploits this ability in HP.

2.3 Malicious insiders Example

We introduce a model of an insider behavior that leads to stealing a master encryption-key in production. This is a simplified real-world example, inspired by an industrial partner. It is simple enough to be explained, as well as strong enough to show the usefulness of HP, especially with *preemption*. An excerpt from the causal graph is shown in Figure 1. Basically, this model represents one strategy to steal the key (*MKS*) by obtaining its encrypted version and decrypting it (as opposed to stealing it decrypted which is omitted for readability).

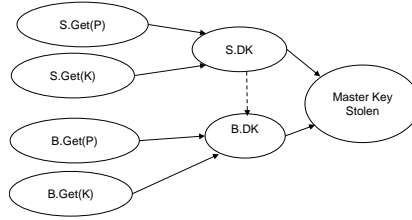


Figure 1: An excerpt of steal key model

The attack can be executed by one of two administrators (assuming no collusion), Suzy (*S*) or Billy (*B*). The two are the *suspects* because they have sufficient privileges in the system. However, *S* has more expertise in the system and the technology. The event of *S* or *B* decrypting the key is denoted by the variables *S.DK*, *B.DK* respectively. For that, each of them needs to read the pass-phrase from a script (*Get(P)*) and read the key from the database (*Get(k)*). For now, let us assume an arbitrary causal connection between *S.DK* and *B.DK* which is meant to represent a *preemption relation*, i.e., a bias to represent *S*’s stronger abilities. This relation is a dashed arrow because we think that such relations can be dynamically altered by a modeler to express different concepts, e.g., *S has higher privileges, B has a better history in the company, S came earlier in the morning the day that the incident was reported*. The exogenous variables are

omitted from the model; we have four exogenous variables that set the values for $Get(P)/Get(k)$ for both S and B , i.e., \mathcal{U} is $\{S.Get(P)_{exo}, S.Get(k)_{exo}, B.Get(P)_{exo}, B.Get(k)_{exo}\}$. The model is not sufficient for causality inference, we still need to set the *context* (exogenous variables). This is done through logging, and auditing.

In a deterministic context, we can use the causal model to answer questions, like $Q1$: *is Suzy the cause of stealing the master key?* or better $Q2$: *what is the actual cause of exposing the master key?* Moreover, we can reason about the actual cause in situations where we have two possible suspects. For example, $Q3$: *Is Billy's decryption of the key or Suzy's the actual cause of stealing it?* All of these questions cannot be answered using an attack tree only. Even if we have attributed the attack tree with the potential suspects, we still cannot infer actual causality directly in cases of preemption or missing events. The equations follow, the dashed part of the equations shows the *preemption* relation.

- $S.DK = S.Get(P) \wedge S.Get(K)$
- $B.DK = B.Get(P) \wedge B.Get(K) \wedge \boxed{\neg S.DK}$
- $MKS = S.DK \vee B.DK$

Given the context $(1, 1, 1, 1)$ when considering the ordering of the variables as provided by the definition of \mathcal{U} , let us answer $Q2$ by checking the conditions from Definition 3. Specifically, we check if $(S.Get(k))$ a cause of (MKS) , with $W = \{B.DK\}$. Equation (1) shows the crucial steps (of checking AC2) to conclude that $S.Get(K)$ is the cause. With an empty set W , AC2 does not hold (case of preemption), but with $W=\{B.DK\}$ (Step 3 Equation (1)), the effect does not happen (Step 4) and hence $S.Get(k)$ is a cause.

<i>Step 1</i>	$S.Get(K) = 0$	Intervening on x	
<i>Step 2</i>	$S.DK = 1 \wedge 0 = 0$	Other variables state	
<i>Step 3</i>	$B.DK = 0$	Cannot change this variable	(1)
<i>Step 4</i>	$MKS = 0 \vee 0 = 0$	Effect is not happening	

Other contexts are simpler to show causality inference had there been no preemption. Another interesting question would be *is B.Get(K) a cause?* The answer is no. In the given context no matter how W is set MKS will still be True.

3 Attack trees to Causal models

Although attack trees are widely used to model attacks on a system, they are not readily *sufficient* to attribute blame. Mainly because they normally do not include the attacker, rather, they represent the attack strategies. That said, they are a promising starting point to create causal models since they express the dependencies among attacker acts, and they match the properties of a causal model. First, ATs are already a propositional combination of events with (*OR*, *AND*) relations. The ability to express ATs in boolean algebra makes it trivial to express it as a causal model. Second, HP focuses on acyclic models; attack trees are acyclic. We present our approach for *extracting causal models from AT* by starting with a general methodology of causal modeling and then instantiating it to the case of attack trees and insiders.

3.1 A Methodology for Causal Modeling

Causality is model relative (see Section 2.2). Thus, the creation of a model is a crucial step for causal analysis. One goal of this paper is to propose a domain-specific causal modeling methodology that refers to the following activities.

1. *Suspect Attribution*: Representing each potential suspect in the model.
2. *Variable Selection*: Listing the different factors that are considered in the model. They represent the causes, effects, and the environment. Each factor is expressed as a variable in the model.
3. *Semantics Expression*: Representing how these variables affect each other using propositional logic operators like *and*, *or* and *negation*.
4. *Variable Classification*: Classifying what can be considered as a cause (or effect) (endogenous) and what not (exogenous). We will see later on that this activity is lightly enforced in our methodology.
5. *Variable Augmentation*: Incorporating any useful knowledge about these variables and their relations. Examples are preemption relations and the independent probabilistic distribution.

Based on the above, we transfer the nodes of an AT into variables of a causal model. In Section 3.2, we transform the original AT \mathcal{T} to an *attributed* AT \mathcal{T}' .

3.2 Suspect Attribution

To bring it closer to causal models, we add *suspects* to AT. This idea is motivated by our aim to bridge the gap between AT and causal models. As shown in this section, the way that we add the suspects is crucial in determining the scope of the causal queries that can be answered using the resulting model. We use the tree shown in Figure 2 (corresponds to the example in Section 2.3) to illustrate our approach. To the best of our knowledge, no prior work has tried to explore approaches to restructure attack trees to include *roles* in an automated manner. *Instances* of roles (e.g., data-center admin Suzy) are the potential attackers (suspects) that have privileges to perform an insider attack. We refer to the process of adding suspects to an attack tree as *suspect attribution*.

Suspect attribution is merely an *unfolding* (duplicating) task of parts of the tree followed by *allotting* the new parts to one suspect. To create a new path for each suspect, we keep the parent node of the gate as is. Then we introduce an intermediate level of new nodes that correspond to insiders. The allotment is represented by renaming the nodes to include the suspect identifier, e.g., *Billy.Read.Pass.Phrase*. The challenging part of this process is determining the location where we start unfolding.

Unfolding a tree can be done at different levels. However, depending on the internal structure, this may produce trees that model different attack vectors. Consequently, the range of the possible causal-queries that can be analyzed using the resulting models depends on the unfolding level. For example, let us consider attributing the left branch of the tree in Figure 2 with *two* instances of an *admin* role, i.e., Billy and Suzy. We can do that at *level two* (root level is one). The

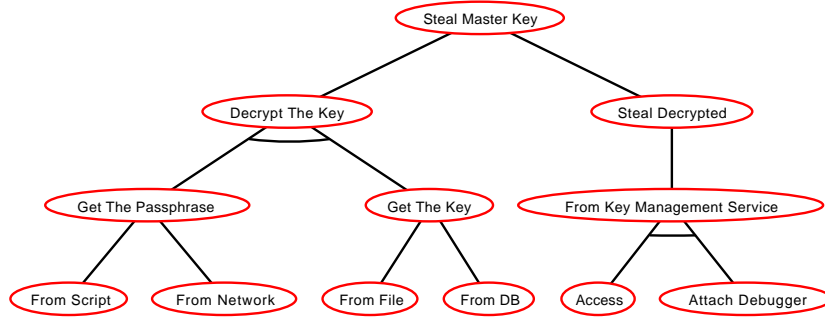


Figure 2: Expose key attack tree

resulting tree is represented in Figure 3. This tree clearly models the possible ways to steal the master key by *either* Billy or Suzy. The complete attack paths in the tree allow expressing the behavior of *one* suspect performing an attack.

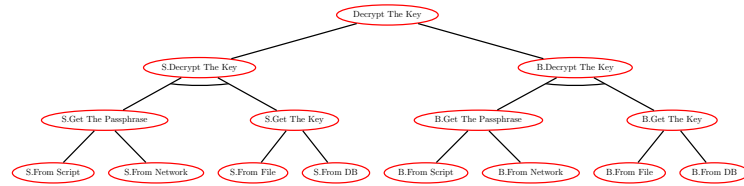


Figure 3: L-2 Unfolding

Alternatively, we can attribute the suspects at the *third* level ($L-3$). Interestingly, the resulting attack tree, as seen in Figure 4, models more possibilities than the previous case. Especially, we now can model attack paths with a possibility of *collusion* between insiders [14]. As a result, attacks that involve *both* Suzy and Billy cooperating to steal the master key are now covered in this tree, and hence, causal-queries to blame them are possible on the resulting model. Since collusion attacks are plausible among insiders [14], the second attribution must be used, especially because it already includes the attacks (and queries) that are covered by the higher level attribution ($L-2$). This comparison is an instance of the specialization concept proposed by Horne et al. [8].

Actually, the *attribution level* is not the crucial factor in determining the expressiveness of the attribution. Somewhat, it depends on the structure and the semantics of the branch (first-level subgraph). Specifically, if we have an *AND* gate in the branch, the expressiveness of the model will depend on the attribution level. If we want to include the possibility of collusion attacks, *then the unfolding should happen at a level that is greater than the AND gate level*.

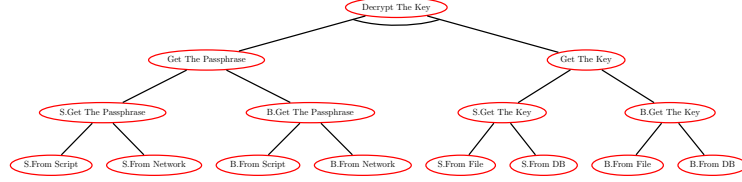


Figure 4: L-3 Unfolding

Although, unfolding after the last AND gate allows considering any possibility of colluding attacks, in some cases it may be unnecessary. For example, let us consider the second branch in Figure 2. If we attribute suspects after the fourth level, then we assume that the suspects may collude by having one accessing a container *and* the other attaching a debugger. This is unlikely to happen. Still, it produces a model that can be used for single-agent queries.

We propose to generate causal models from *attributed AT* based on different *attribution levels*. The levels are determined for each branch based on its structure. However, they can be overridden by a decision of the modeler.

Semantics of Attribution Let us start with **AND Gates**. An AND gate is visualized in the left column of Table 1. The semantics of the node is given by the formula associated with it, i.e., $a = b \wedge c$. We discussed how to unfold such a gate, at the first level which does not account for collusion attacks (middle column), or at the second level (right column).

The semantics of unfolding the (L_1) with two suspects (denoted by ' and ") is shown in second row (steps 1 – 3) of Table 1. The last step shows a disjunctive normal form (DNF) of the formula. Similarly, the right column shows the formulas and simplification of unfolding at (L_2) . Comparing the forms shows that the possible attack scenarios of (L_1) unfolding are included in the (L_2) unfolding (this can be seen as a specialization [8] of attack trees). In other words, the formula (L_1) *implies* (L_2) , i.e., $L_1 \implies L_2$ is a *tautology*. Thus, causal queries of the single blame can also be answered when unfolding on the second level.

Unfolding allows us to attribute possible suspects of an attack to the best of the modelers' knowledge. Simplifying the unfolded gates into their DNF proves the *preservation* of the original gate semantics, i.e., $a = b \wedge c$. Essentially the occurrence of the two concrete actions (b, c) combined causes an event (a) . This is expressed in each clause of the DNFs. Informally, a clause is one instance of the original formula. We have to keep in mind, that this transformation is built on the assumption that the list of suspects is the universe of all the possible agents that can perform this attack. This assumption allows us to say that the semantics of the transformed tree (or branch) is now refined to enumerate all the possible scenarios, each presented as a clause that combines single or multi-suspects. Lastly, the case of unfolding OR gates is similar and simpler because the complication of the unfolding level is eliminated. Regardless of the level, an original formula like $a = b \vee c$, will be unfolded to $a = b' \vee c' \vee b'' \vee c''$.

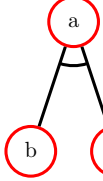
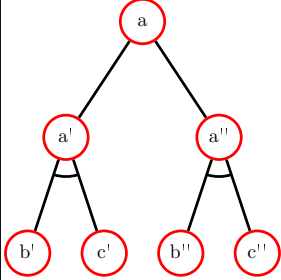
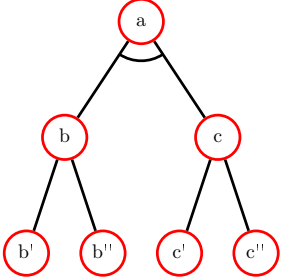
Gate	$L - 1$	$L - 2$
		
Semantics	<ol style="list-style-type: none"> 1. $a = a' \vee a''$ 2. $a' = b' \wedge c'$ 3. $a'' = b'' \wedge c''$ 4. $a = (b' \wedge c') \vee (b'' \wedge c'')$ 	<ol style="list-style-type: none"> 1. $a = b \wedge c$ 2. $b' = b' \vee b''$ 3. $c = c' \vee c''$ 4. $a = (b' \vee b'') \wedge (c' \vee c'')$ 5. $a = (b' \wedge c') \vee (b' \wedge c'')$ $\vee (b'' \wedge c') \vee (b'' \wedge c'')$

Table 1: Unfolding AND

3.3 Attributed Attack Tree Transformation

Since we are reusing the existing knowledge in the attributed attack trees, the three activities: *variable selection*, *semantics expression*, and *variable classification* are trivial. Basically, we consider each node as an *endogenous* variable that defines whether or not an attack step has been conducted. Since the nodes are connected with different operators, we use them to construct the equations and therefore express the semantic relationships between the variables. Before we do the transformation, we need to extend the tree, i.e., duplicate its leaves.

In attack trees, a leaf node represents an atomic step that is not further refined [31]. When transferring leaves into *endogenous* variables of a causal model, they lack corresponding formulas. Alternatively, we can consider them as *exogenous* variables that represent the environment (context), but then they cannot be regarded as potential causes in our reasoning. Thus, we extend the tree with a duplicate set of leaves. In other words, each leaf on the tree gets an inbound edge from a new node that has the same name with an *_exo* suffix. Tree extension aids us in *classifying the variables*, and it also maintains the possibility that any node in the original tree can be considered as a cause. Definition 4 is a tree extension function, where $E(T)$ copies the set of leaves of a tree T .

Definition 4. Extension Rule The relation $T(\mathcal{N}, \rightarrow, n_0) \Rightarrow T''(\mathcal{N}'', \rightarrow'', n_0)$ is defined by the following rule.

- $\mathcal{N}'' : \mathcal{N} \cup \text{rename}(E(T), \text{_exo})$; where $\text{rename}(A, \text{suffix})$ is a function that renames nodes in set A to with a given suffix.
- $\rightarrow'' : \{\rightarrow\} \cup \forall_{m \in E(T)} (m \rightarrow m\text{_exo})$

We should note that the same node can occur multiple times in AT. However, in our causal model, exactly one instance of a variable exists. For the scope of this

paper, we only allow node re-occurrence among leaves. So far, we discussed *four* steps in our extraction process which are related to the AT. Now, we are ready to create the model from the extended and attributed AT. We will illustrate that by a formal mapping that depends on the definitions (Section 2).

Definition 5. Attack Tree To Causal Model

$AT = (\mathcal{N}, \rightarrow, n_0, [[n]])$ is mapped to a $M = (U, V, R, F)$ i.e. $AT \mapsto M$ as follows

- $\mathcal{U} = E(AT)$, where $E(AT)$ returns the leaf nodes of a tree AT
- $\mathcal{V} = \mathcal{N} \setminus E(AT)$, where \setminus is the difference between two sets.
- $\mathcal{R} = \{0, 1\}$.
- \mathcal{F} associates with each $X \in \mathcal{V}$ a propositional formula $F_X = [[X]]$, which corresponds to the semantical formula from the AT

3.4 Adding Preemption Relations

So far we discussed how to map the structure (variable and dependencies), the semantics (formulas), and their causal importance (endogenous or exogenous). Now we discuss *variable augmentation*. We augment the model with suspect-related information that is useful to create *preemption relations*. *Preemption relations* are a significant extension of the model. They represent relations between variables that express the same event conducted by a different suspect (e.g., *Billy.Get Key/Suzy.Get Key*). They are decisive in models that have potential identical causal relations [7] (see the example in Section 2.3). HP introduces a treatment for such cases by relating them “somehow”. We adopt this in the context of insiders by introducing a relation between the variables *one level* after suspect attribution. The relation is based on what we call the *suspiciousness metric* (SM). SM provides an order relation over the set of suspects conducting a particular type of attack. In other words, it is a value given to each potential attacker to aggregate their abilities in performing an event or willingness to commit an attack. This can be related to a risk assessment of insiders. For example, it represents the level of Suzy’s privileges in a system, Billy’s criminal record, or a combination of these factors. The values of *SM* are used to reflect the possible disparity among suspects, which can be global (a value of the attacker ability for all possible attacks) or local (a value of attacker ability for a specific attack). For simplicity, we limit ourselves to global SM.

Semantically, the preemption relation is represented by a *not* clause ($\neg X$) added to the *less suspicious* (i.e., smaller value) suspect about the *higher suspicious* suspect. For example, $B.DK = B.Get(P) \wedge B.Get(K) \wedge \boxed{\neg S.DK}$. The dashed box represents a preemption relation.

3.5 Tool Support

Having introduced our approach, we now present our tool **ATCM** (**Attack Tree to Causal Model**). *ATCM* is a command line tool that implements our

approach.³ As the name suggests, *ATCM* takes an attack tree and suspects' specification as an input and generates a causal model. Attack trees are usually created using a broad variety of tools. In order to get access to the information stored in such a AT, the latter needs to be exportable to a format that can be easily accessed and used by us. Examples of tools fulfilling this requirement are *ADTool* [15,5] which provides an XML-representation of the models created with them. Consequently, we are able to use those as input for *ATCM*.

In general, *ATCM* incorporates a three-step approach: parsing, transformation, and extraction. First of all, we need to create a machine-readable object, i.e., binary, representation out of a given XML-File that defines an attack tree (Parsing). For this purpose, we have developed our own parsing components. However, since this object representation is specifically tailored to each of the supported file formats, we want to transform the latter into a uniform tree representation, which comprises both attack and other similar models such as fault trees, while ensuring that no semantic information is lost (Transformation). For this representation, we are using the *Model Exchange Format (MEF)* (<https://open-psa.github.io/mef/>) in a slightly simplified form.

The advantage of abstracting the specific format such as ADT format is that the most essential functionality of this tool, i.e., the extraction of the causal model, needs to be developed only once. This reduces its error-proneness and increases maintainability. Once an attack has been transferred into this uniform representation, the described generation of the causal model can begin (Extraction). We export the results in a human-readable report and generate a causal graph in the *DOT* format, which is a commonly used for describing graphs in a textual format and can be rendered into a visualization by multiple tools.

4 Evaluation

In our evaluation, we analyze the following qualities: *the efficiency* of the model extraction, *the validity*, and *the effectiveness* of the resulting models. For the first, we discuss (in Section 4.1) the performance cost and the size expansion of the tree in relation to different variables. In Section 4.2, we focus on the quality of the model. Clearly, we do not aim to discuss the expressiveness of AT since their refinement and granularity are decided by the modeler. However, we discuss the validity of our models in relation to the input AT. Lastly, we discuss how to use the causal model in a technical setting to infer causality.

We use *four* classes of use-cases in our experiments. Table 2 shows the particular attack trees of each class, along with the number of nodes in the tree. Each class contains one or more trees that cover different sources as follows: **1)** HP examples: We use two famous examples from the causality domain [6]. This class is mainly used for the discussion of the validity. **2)** Insiders from industry: This class includes a real-world attack tree which comes from an industrial partner. It represents insider's strategies to *steal a master key* from a deployment of an

³ *ATCM* is available at: <https://github.com/amjadKhalifah/ATCM>

Class	Use Case	Nodes	# Potential Attackers
HP	HP ₁	3	2
	HP ₂	2	2
Insider (Industry)	Steal Master Key	12	{2, 8}
Insider (Literature)	BecomeRootUser ₁	8	{2, 8}
	BecomeRootUser ₂	11	{2, 8}
Artificially Generated	Artificial ₁	255	{2, 8}
	Artificial ₂	1017	{2, 8}
	Artificial ₃	3057	{2, 8}

Table 2: Use Cases of our evaluation

enterprise solution. **3)** Insiders from Literature: This class includes two attack trees borrowed from [34]. They represent privilege escalation. The first uses windows command line and scheduler, and the other uses Metasploit and Internet Explorer. **4)** Artificially generated trees: This class contains three trees that we generated automatically. They do not hold any semantic value. The aim of using them is to analyze the efficiency of extraction. In our experiments, we will vary the number of attackers and test our model extraction for 2, and 8 suspects.

4.1 The Efficiency of the Extraction

Depending on the size, the structure of the AT, the *attribution level* l of each branch, and the number of suspects s , the size of the resulting model will vary. Since we are attributing branches at different levels, the size of the resulting model is the sum of attributed branch-sizes plus one. This is expressed as $((\sum_{i=1}^n |b_{li}(s)|) + 1)$, where n is the number of branches, and $|b_{li}(s)|$ is the size (number of nodes) of branch i attributed at level l with s suspects. We express the attributed branch size $|b_{li}(s)|$ as a function of suspects and its original size.

Definition 6. Attributed Branch Size $|b_{li}(s)|$

$$|b_{li}(s)| = (s \cdot (|b_i| - |b_i|_{l>L>1} + |b_i|_{Leafs}) + |b_i|_{l \geq L > 1})$$

- $|b_i|, |b_i|_{Leafs}$ are the sizes of the original branch b_i and the number of its leaves,
- $|b_i|_{l>L>1}, |b_i|_{l \geq L > 1}$ are the size of the exclusive and inclusive subtree between the branch root and attribution level. Inclusion refers to counting the root and the leaves or excluding them.

We clearly see that our approach increases the number of nodes in the tree. For future work, we plan to make use of the variance brought by manipulating the factors that control the size. For example, filtering the model to include tree branches, including less number of suspects, and testing models attributed at different levels. Next, we evaluate the efficiency of the extraction process. Table 3 shows the execution time $exec(s)$ and the model size n of four AT (their properties are shown as n: number of nodes, l: depth of the tree, and b: number of branches). We have attributed the four trees with 2 and 8 suspects. We attributed each tree at *root-level, middle-level, and leaf-level*. We created benchmarks, based on *Java Microbenchmark Harness* to measure the execution

				2 Suspects						8 Suspects					
				Top		Middle		Leafs		Top		Middle		Leafs	
AT	n	l	b	n	exec(s)	n	exec(s)	n	exec(s)	n	exec(s)	n	exec(s)	n	exec(s)
SMK	12	5	2	37	0.0002	36	0.0002	36	0.0003	139	0.0004	126	0.0004	108	0.0004
Be.Root1	8	4	1	24	0.0002	25	0.0002	23	0.0002	90	0.0004	91	0.0004	71	0.0004
Be.Root2	11	4	1	32	0.0002	35	0.0002	32	0.0003	122	0.0006	125	0.0006	98	0.0006
T ₁	255	8	2	767	0.0069	767	0.0117	767	0.0512	3059	0.0283	2879	0.0460	2303	0.1925
T ₂	1017	8	8	3065	0.0354	3065	0.1133	3065	0.7473	12233	0.1380	11513	0.4610	9209	2.99
T ₃	3057	8	16	6129	0.0939	6129	0.4084	6129	2.94	24465	0.3700	23025	1.65	18417	11.97

Table 3: Evaluation of the Efficiency

time. The benchmarks measure the time from parsing an AT until the creation of the corresponding causal model. The values shown in Table 3 have been obtained by running 10 warm-up and 20 measurement iterations on a Windows 10 machine equipped with 8GB of RAM and a quad-core Intel® i7 processor.

For the small use cases (SMK, Be.Root1, and Be.Root2) the execution time is small (below 0.7ms). The interesting part is with the artificial trees, where we see a clear proportional increase of execution time with the *deeper* attribution levels. This is due to our recursive algorithm. Model size, on the other hand, is of less importance in that context, we can see that a 23025 node model took 1.7 sec to be extracted (L-4), while a 9209 node model took 2.9 secs (L-8). Nevertheless, these values do not exhibit a bottleneck. Hence, based on this empirical evaluation, our approach should be efficient enough for any reasonable-sized AT.

4.2 The Validity of the Approach

There are no properties that discuss the validity of a causal model. Rather, scientists have dealt with modeling by example. We use a similar approach. We apply our approach to problematic examples in the literature [7] and compare the results. Our goal is to check if we were able to automate the method of creating models by splitting the general knowledge represented as trees from the suspects. Although those examples are not security attacks, we model them as such.⁴ To that end, we followed the following process. *First*, represent the abstract causal knowledge as a tree (Table 4 middle column). *Second*, configure the actors in the scenarios, e.g., Billy and Suzy. *Third*, generate the model (Table 4 right column). *Lastly*, compare the generated model with the model presented in the papers.

For space limit, we only present two examples (**Arsonists**, **Rock Throwing**). Arsonists (example 3.2 in [7]): Suppose that *two* arsonists drop lit matches in different parts of dry forest, and both cause trees to start burning. Either match by itself suffices to burn down the whole forest. HP describes the essential structure with 3 endogenous variables ML_1 and ML_2 , where $ML_i = 0$ if

⁴ Arsonists and Rock-Throwing are typical examples in the causality literature. We may consider setting a forest on fire as an attack on the forest, with lighting matches being a possible step of an attack. We may also consider shattering a bottle an attack on the bottle, with throwing a stone being a possible step of an attack. The point here is to show that our mechanism produces valid results also for well-known examples.

Example	HP Model	Attack Tree	Our Model
Arsonists	<pre> graph TD A1ML_exo --> A1ML A2ML_exo --> A2ML A1ML --> FB A2ML --> FB </pre>	<pre> graph TD FB --> ML </pre>	<pre> graph TD A2ML_exo --> A2ML A1ML_exo --> A1ML A2ML --> ML A1ML --> ML ML --> FB </pre>
Rock-Throwing	<pre> graph TD ST_exo --> ST BT_exo --> BT ST --> SH BT --> BH SH --> BS BH --> BS </pre>	<pre> graph TD BS --> H H --> T </pre>	<pre> graph TD ST_exo --> ST BT_exo --> BT ST --> SH BT --> BH SH --> H BH --> H H --> BS </pre>

Table 4: Models From HP Examples

arsonist i does not drop the lit match and 1 otherwise, and similarly, a variable FB for forest burns down. Rock Throwing (example 3.2 in [6]: “Suzy and Billy both pick up rocks and throw them at a bottle denoted in the model as (ST and BT). Suzy’s rock gets there first, shattering the bottle denoted in the model as (BS). Since both are perfectly accurate, Billy’s would have shattered the bottle had it not been preempted by Suzy’s throw.” Table 4 shows that the models vary a bit. This variation is negligible because it does not affect the semantics from a causal perspective. Our model can be proved easily to be a conservative extension ([7]) of the model from HP.

4.3 The Effectiveness of the Model

To evaluate the effectiveness of our models, we briefly show how they are used in a production environment. We experimented with a technical setting of the example from Section 2.3. First, we created the corresponding *model* using ATCM. Second, we set the *context* for two concrete scenarios: *Sce-1*, in which **S**uzy stole the key with the existence of preemption, and *Sce-2*, in which **B**illy did. Third,

using the work in [11], we reason about the causality. We tested the models in an environment that contains a set of micro-services and third-party software that are deployed as docker containers. To set the *context*, we utilized monitoring tools namely, auditD to monitor file accesses, and Couchbase audit to monitor queries. These tools generate logs that we used to set the exogenous variables. For our initial prototype, the context was set manually. For example, a sentence from auditD like ... "MESSAGE" : "PATH name=../ script.txt ..auid= 1001 uid= 1001.. is translated into $S_{From_Script_exo} = 1$ (Suzy's id=1001). $\mathcal{U} = \{S_{Script_exo}, S_{DB_exo}, S_{File_exo}, S_{NW_exo}, \dots\}$ Billy's variables}. Accordingly, we have two contexts, namely *Sce-1* $\{1, 1, 1, 1, 1, 1, 1\}$ and *Sce-2* $\{0, 0, 0, 0, 1, 1, 1\}$ when we consider the ordering of the variables.

Due to a recent tool (HP2SAT) that aimed for an efficient computation of actual causality [11], we analyzed the two contexts using the *steal master key* 8-suspects model with 91 endogenous and 48 exogenous variables. We used the two questions from Section 2.3: *Q1: is Suzy the cause of stealing the key?*, *Q2: Is Billy's decryption of the key or Suzy's the actual cause of stealing it?*.

Sce-1 represented the situation of having multiple tentative suspects. The results matched our ground truth, i.e., Suzy was concluded to be responsible for the incident. Although this may seem intuitive, it was only enabled by the fact that we made our knowledge explicit using a causal model. The analysis of *Q1* took 3.07ms and consumed 3.2MB of memory. For *Sce-2*, it was easier to conclude that Billy is the reason for stealing the key since the context was clearer (Suzy and other suspects did not log into the system). The analysis of the model for *Q2* took 2.78ms and consumed 3.2MB of memory.

5 Related Work

Security. To the best of our knowledge, no previous work has tried to generate HP models for malicious insiders. However, the thorough work on attack and defense modeling is interesting. Kordy et al. [16] surveyed the DAG-based models. Their main classification of the models is either tree or Bayesian network (BN) based. Although a BN is similar to a causal model, there are two differences in utilizing them in security. First, BNs are used for the probabilistic inference of an attack likelihood and prediction. However, we aim to use the causal models for inferring *actual* causality. Second, a causal model contains a semantic perspective represented by the SEM, while BN only contains a dependency relation supported by conditional probability table. In this direction, we see the work by Qin et al. [25] which indeed converts attack trees to BN to correlate alerts to predict attacks. Similarly, Althebyan and Panda [1] present a BN model to evaluate and analyze a system after an insider attack. Their evaluation and analysis do not include attributing the attacker. Poolsapassit and Ray [24,26] use AT in a similar way. They do not convert it to other models but rather combine it with insider's intent to predict malicious activity. In [24] they use AT to investigate logs. These two papers are related to our goal but different in the approach of converting AT to causal models annotated with possible suspects. Most of the

work reporting on insiders [30,23] aims to detect the attacks at run-time [14]. Although our work can be combined with such approaches, this is fundamentally different since we consider postmortem attribution. Chinchani et al. [4] proposed a modeling language for insiders. This is interesting, however, we used AT for reasons of industry utilization and tool support [15].

For attack attribution, researchers [32,9] have identified three techniques: digital forensics, Malware based analysis, and indirect attribution techniques that use statistical models to identify attackers. Most of these techniques target outsider attackers. Unlike our approach, digital forensics tools mainly face the challenge of scalability with the size of logs [32], whereas we can elicit requirements of logging from our modes. That is, we only monitor the properties that set our context. Malware based analysis targets a different attack vector than us. Indirect attribution techniques are interesting since they use a statistical model. However, they require massive amounts of data. In contrast, we make use of explicit knowledge represented in attack trees.

Causality is a cross-domain concept. An overview of different fields of causality applications is presented by Halpern in [7]. An example of research based on the HP definition is the work by Kuntz et al. [18], who are using counterexample traces of model checking tools to construct fault trees. This is similar to our target in general but different in two key aspects. First, it does not leverage security threat models to construct causal models that are used to infer causality in the postmortem. Second, our approach is to model only unwanted behavior while they utilize behavioral models of the systems. Similar to our approach, Ibrahim et al. [10] created holistic causal models for Cyber-Physical Systems from different source models. Their approach focused on combining those models without focusing on one attack vector like insiders. Pearl examines causal modeling in [22]. Further literature examining model discovery is listed in a paper by Chen and Pearl [3]. These approaches are data-driven methods that differ from our approach of creating causal models from other models.

6 Conclusions and Future work

To handle the insider threat, we proposed enabling accountability through supporting causal reasoning. To that end, we presented a methodology that introduces HP causal models into the security domain. We showed that such models are beneficial in the context of insiders and we considered AT as a source for creating them. However, we identified suspect attribution as a crucial step in the conversion. Thus, we introduced a method to add suspects to AT considering the possibility of them colluding. Also, we focused on creating models that include preemption relations. This work can then be combined with causality reasoners to enable forensics analysis of insider accidents. Although it is hard to evaluate models reasonably, we showed that our approach is *efficient* to extract *valid and useful* models. Next, we plan to automate the step of context setting which is concerned with the logging capabilities.

References

1. Althebyan, Q., Panda, B.: A knowledge-based bayesian model for analyzing a system after an insider attack. In: Proceedings of The Ifip Tc 11 23 rd International Information Security Conference. pp. 557–571. Springer (2008)
2. Aslanyan, Z., Nielson, F.: Model checking exact cost for attack scenarios. In: International Conference on Principles of Security and Trust. Springer (2017)
3. Chen, B., Pearl, J.: Graphical tools for linear structural equation modeling. Tech. rep., DTIC Document (2014)
4. Chinchani, R., Iyer, A., Ngo, H.Q., Upadhyaya, S.: Towards a theory of insider threat assessment. In: Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on. pp. 108–117. IEEE (2005)
5. Gadyatskaya, O., Jhavar, R., Kordy, P., Lounis, K., Mauw, S., Trujillo-Rasua, R.: Attack trees for practical security assessment: Ranking of attack scenarios with adtool 2.0. In: Quantitative Evaluation of Systems - 13th International Conference, QEST 2016. pp. 159–162 (2016)
6. Halpern, J.Y.: A modification of the halpern-pearl definition of causality. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, (2015)
7. Halpern, J.Y.: Actual causality. MIT Press (2016)
8. Horne, R., Mauw, S., Tiu, A.: Semantics for specialising attack trees based on linear logic. *Fundamenta Informaticae* 153(1-2), 57–86 (2017)
9. Hunker, J., Hutchinson, B., Margulies, J.: Role and challenges for sufficient cyber-attack attribution. Institute for Information Infrastructure Protection (2008)
10. Ibrahim, A., Kacianka, S., Pretschner, A., Hartsell, C., Karsai, G.: Practical causal models for cyber-physical systems. In: Badger, J.M., Rozier, K.Y. (eds.) *NASA Formal Methods*. pp. 211–227. Springer International Publishing, Cham (2019)
11. Ibrahim, A., Rehwald, S., Pretschner, A.: Efficiently checking actual causality with sat solving. In: *Dependable Software Systems Engineering*, p. to appear (2019)
12. Institute, P.: 2015 cost of cyber crime study: Global
13. Kacianka, S., Beckers, K., Kelbert, F., Kumari, P.: How accountability is implemented and understood in research tools. In: *International Conference on Product-Focused Software Process Improvement*. pp. 199–218. Springer (2017)
14. Ko, L.L., Divakaran, D.M., Liau, Y.S., Thing, V.L.: Insider threat detection and its future directions. *International Journal of Security and Networks* 12(3) (2017)
15. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: Adtool: Security analysis with attack-defense trees. In: *Quantitative Evaluation of Systems - 10th International Conference, QEST*. pp. 173–176 (2013)
16. Kordy, B., Piètre-Cambacédès, L., Schweitzer, P.: Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer science review* 13, 1–38 (2014)
17. Kumar, R., Ruijters, E., Stoelinga, M.: Quantitative attack tree analysis via priced timed automata. In: *International Conference on Formal Modeling and Analysis of Timed Systems*. pp. 156–171. Springer (2015)
18. Kuntz, M., Leitner-Fischer, F., Leue, S.: *From Probabilistic Counterexamples via Causality to Fault Trees*. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
19. Lewis, D.: Counterfactuals and comparative possibility. *Journal of Philosophical Logic* 2(4), 418–446 (Oct 1973), <https://doi.org/10.1007/BF00262950>
20. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: *Information Security and Cryptology - ICISC 2005, 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers*. pp. 186–198 (2005)

21. Nicholson, A., Janicke, H., Watson, T.: An initial investigation into attribution in scada systems. In: ICS-CSR (2013)
22. Pearl, J.: Causality. Cambridge University Press (2009)
23. Phyto, A., Furnell, S.: A detection-oriented classification of insider it misuse. In: Third Security Conference (2004)
24. Poolsapassit, N., Ray, I.: Investigating computer attacks using attack trees. Advances in digital forensics III pp. 331–343 (2007)
25. Qin, X., Lee, W.: Attack plan recognition and prediction using causal networks. In: Computer Security Applications Conference, 2004. 20th Annual. IEEE
26. Ray, I., Poolsapassit, N.: Using attack trees to identify malicious attacks from authorized insiders. In: ESORICS. vol. 3679, pp. 231–246. Springer (2005)
27. Reháč, M., Staab, E., Fusenig, V., Pěchouček, M., Grill, M., Stiborek, J., Bartoš, K., Engel, T.: Runtime monitoring and dynamic reconfiguration for intrusion detection systems. In: International Workshop on Recent Advances in Intrusion Detection. pp. 61–80. Springer (2009)
28. Research, I.X.F.: 2016 cyber security intelligence index
29. Rowlingson, R.: A ten step process for forensic readiness. International Journal of Digital Evidence 2(3), 1–28 (2004)
30. Salem, M.B., Hershkop, S., Stolfo, S.J.: A survey of insider attack detection research. In: Insider Attack and Cyber Security, pp. 69–90. Springer (2008)
31. Schneier, B.: Attack Trees - Modeling security threats. Dr. Dobb's Journal (1999)
32. Shamsi, J.A., Zeadally, S., Sheikh, F., Flowers, A.: Attribution in cyberspace: techniques and legal implications. Security and Communication Networks 9(15) (2016)
33. Tan, J.: Forensic readiness. Cambridge, MA:@ Stake pp. 1–23 (2001)
34. Tu, M., Xu, D., Butler, E., Schwartz, A.: Forensic evidence identification and modeling for attacks against a simulated online business information system. Journal of Digital Forensics, Security and Law 7(4), 4 (2012)
35. Weitzner, D.J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., Sussman, G.J.: Information accountability. Communications of the ACM 51(6) (2008)
36. Wheeler, D.A., Larsen, G.N.: Techniques for cyber attack attribution. Tech. rep., INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA (2003)