On Evidence Preservation Requirements for Forensic-Ready Systems

Dalal Alrajeh
Imperial College London
London, UK

Liliana Pasquale
University College Dublin
Dublin, Ireland

Bashar Nuseibeh
The Open University, UK, &
Lero, Ireland

ABSTRACT

Forensic readiness denotes the capability of a system to support digital forensic investigations of potential, known incidents by preserving in advance data that could serve as evidence explaining how an incident occurred. Given the increasing rate at which (potentially criminal) incidents occur, designing software systems that are forensic-ready can facilitate and reduce the costs of digital forensic investigations. However, to date, little or no attention has been given to how forensic-ready software systems can be designed systematically. In this paper we propose to explicitly represent *evidence preservation requirements* prescribing preservation of the minimal amount of data that would be relevant to a future digital investigation. We formalise evidence preservation requirements and propose an approach for synthesising specifications for systems to meet these requirements. We present our prototype implementation—based on a satisfiability solver and a logic-based learner—which we use to evaluate our approach, applying it to two digital forensic corpora. Our evaluation suggests that our approach preserves relevant data that could support hypotheses of potential incidents. Moreover, it enables significant reduction in the volume of data that would need to be examined during an investigation.

CCS CONCEPTS

• **Software and its engineering** → *Requirements analysis*; • **Applied computing** → *Evidence collection, storage and analysis*;

KEYWORDS

Forensic-ready systems, requirements, specification synthesis

ACM Reference Format:
Dalal Alrajeh, Liliana Pasquale, and Bashar Nuseibeh. 2017. On Evidence Preservation Requirements for Forensic-Ready Systems. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 11 pages. https://doi.org/10.1145/3106237.3106308

1 INTRODUCTION

Digital forensic investigations are concerned with the discovery, collection, preservation, analysis, interpretation and presentation of digital data from digital sources, for proof of incident and ultimately for prosecution of criminal activity [16, 36]. Such data often comprises log entries indicating the occurrence of events in the digital sources placed within the environment in which an incident can occur. Despite the availability of digital forensics tools for evidence acquisition and examination (e.g., [2, 10, 14]), these tools are designed to be used only *after* an incident occurs and an investigation commences. However, some of the relevant data may not be available then because, for example, it was stored in a volatile memory or it has been intentionally tampered with by an offender. Moreover, digital forensic tools do not select among the data that might be relevant for investigating a specific incident, thus requiring investigators to sift through large volumes of data to determine what may be considered as relevant evidence. This can be a cumbersome and an error-prone process [29, 54].

Software systems should be *forensic-ready* [45], i.e., able to support digital forensic investigations of *potential*, *known* incidents by preserving in advance data that may serve as evidence explaining how an incident occurred. Given the increasing rate at which (potentially criminal) incidents occur, designing software systems that are forensic-ready can facilitate and reduce the costs of digital forensic investigations. However, existing research has provided only generic guidelines capturing operational and infrastructural capabilities for organisations to achieve forensic readiness [15, 42]. Little or no attention has been given to how forensic-ready systems can be designed and verified systematically, nor to how to ensure their suitability for the specific environments in which they will be deployed [51]. Without a formal conceptualisation of a forensic-ready system and a software design methodology for achieving it, ensuring the soundness of any automated investigative process or its outcome becomes difficult, or even impossible [43].

Forensic-ready systems should satisfy *evidence preservation requirements*, i.e. ensure preservation of *relevant* and *minimal* evidence. On the one hand, preservation of an excessive amount of data often introduces resource and performance issues [40], and increases the cognitive load on investigators who have to make sense of a large data-set. On the other hand, preservation of an insufficient amount of data provides an incomplete picture of how an incident would have occurred, thus making way for misguided decisions and potentially wrong convictions [12].

This paper addresses some of the challenges in the systematic design of forensic-ready systems by making the following contributions: (i) precise definition of evidence preservation requirements and the concepts upon which these requirements depend; (ii) a method for generating preservation specifications that satisfy evidence preservation requirements and (iii) a prototype tool for generating such specifications automatically, which we use to evaluate our approach. More specifically, we first provide formal definitions

ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3106308

of the domain model of a forensic-ready system including the environment in which incidents may occur and hypotheses about such incidents. We also formalise preservation specifications and requirements. We then present a synthesis approach that combines deductive reasoning and inductive learning to generate preservation specifications from a formal description of an environment and hypotheses. To the best of our knowledge this is the first work that conceptualises the requirements of evidence preservation and demonstrates its benefits in reducing data needed to be examined during an investigation.

In this work, we assume a domain expert provides a set of correct incident hypotheses. To achieve this aim s/he can follow a proper risk assessment methodology (using approaches such as [9]) over the threats/incidents considered most likely and highly-critical. Specifications are expressed declaratively in Linear Temporal Logic (LTL) [39], a commonly used formalism for specifying a system behaviour, and prescribe constraints over when events happening at the digital sources in the environment must be logged. We also assume that a designated software controller, the *forensic readiness controller*, is responsible for the enactment of the specification by interacting with the digital sources through a uniform interface. We evaluate our approach by applying it to two substantive case studies publicly available. Our evaluation suggests that our approach preserves relevant data to explain potential incidents and enables significant reduction in the volume of data that would need to be examined during an investigation.

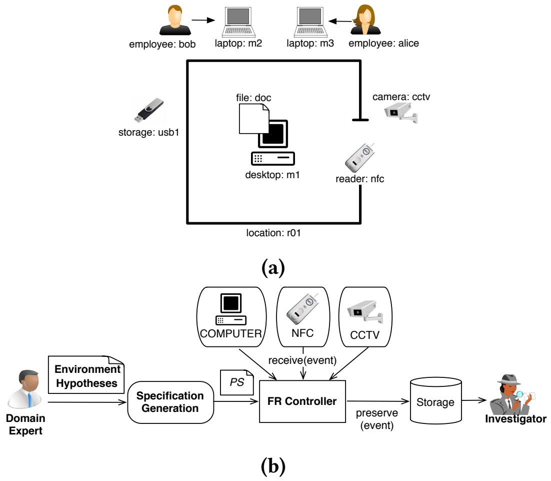
The rest of the paper is organised as follows. Section 2 presents a motivating example and an overview of our approach. Sections 3 and 4 formalise the domain model of a forensic-ready system and preservation specifications and requirements. Sections 5 and 6 explain our approach to generate specifications and its implementation. Section 7 presents the results of our evaluation. Section 8 gives an overview of related work, and Section 9 concludes.

2 MOTIVATING EXAMPLE AND OVERVIEW

We motivate our work using an example of a corporate fraud incident, inspired by the Galleon Group case [35]. We consider an environment (Fig. 1a) within an enterprise building , where two employees, *bob* and *alice*, work and are provided with laptops (*m2* and *m3*, respectively) by the company. A sensitive document *doc* is stored on the server machine *m1* located in the office *r01*. Access to *r01* is controlled by an NFC reader (*nfc*) and is monitored by a CCTV camera (*cctv*). Both *alice* and *bob* are authorised to access *r01* and to login to *m1*.

Suppose that a digital investigation related to the exfiltration of the *doc* is initiated. An investigator may hypothesise that the *doc* was copied onto a storage device mounted on *m1*. To verify this, she must first speculate the activities that may have occurred within the environment and reconstruct different possible scenarios based on these activities. A possible scenario is that *alice* enters room *r01*, logs into *m1*, mounts *usb1* on *m1* and copies *doc* onto *usb1*. Another is that *bob* enters room *r01* but logs into *m1* using *alice*'s credentials, and then copies the *doc* onto *usb1*.

Once the scenarios have been identified, the investigator must establish which of the digital devices holds data that might be relevant to the scenarios. Then she must search through the storage of these devices (e.g., logs for all readers and CCTVs, and hard drives for all computers) for relevant information. However, the investigator may fail to find information about storage devices mounted on *m1* because the system log of *m1* only retains information about the last device that was mounted. The large number of events that can occur also does not make it plausible to preserve all events in advance for later examination. For example, corporate computers can generate over 100 millions of events per week[1], while the number of accesses recorded by CCTV cameras and card readers can exponentially grow with the number of employees and rooms in a building. Moreover, not all the events are relevant to support the speculated scenarios. For instance, only copies of the *doc* taking place while a storage device is mounted and a user is logged on *m1* are relevant for our example.



(a)

(b)

Figure 1: Setting of our example (a) and overall approach (b)

Our approach (Fig. 1b) aims to design a *FR controller* that receives events from digital sources within the environment and selectively preserves them in a secure storage. These events can be acquired and examined by an investigator during future digital investigations. We provide an automated approach (*Specification Generation*) to generate a preservation specification (*PS*) for the FR controller. We assume a domain expert (e.g., security administrator or software engineer) provides a description of the environment and a predefined set of hypotheses about incidents of concern. The description of the environment also includes information about what activities can be monitored by the digital sources. To generate a specification, we first check whether the hypotheses are feasible within the environment (i.e., they may hold if certain activities take place in the environment). If this is the case, the approach generates a set of possible sequences of low-level system events (called potential histories) that demonstrate this. The approach then verifies whether the existing specification already ensures the preservation of events that correspond to these generated histories. If not, then our approach inductively synthesises a preservation specification that configures when an event occurring in a digital source within

[1] We estimated having 50 Events Per Second (EPS) during non peaks and 2500 EPS during peaks. An organisation that experiences peaks for 5% of the total time will have an average of 215 EPS (125 EPS for non-peak and 90 for peak).

the considered environment should be preserved, according to their relevance to the hypotheses. For our motivating example, our approach would prescribe preservation of events indicating copies of the *doc* only if a storage device was previously mounted and a user has previously logged on *m1*. In the next two sections we define the artefacts which constitute the domain of a forensic-ready system (environment, histories and hypotheses) and their relation to preservation specifications and requirements.

3 FORENSIC DOMAIN MODEL

We provide here a formal underpinning of the concepts and terminology that are commonly used in digital forensic domain [11] to describe the environment in which an incident can occur, histories and the incident hypotheses.

3.1 Environment Description

The environment description is a set of descriptive statements about: (i) the *context* in which an incident may occur, (ii) the *behaviour* that may be exhibited within the environment and (iii) their *interactions*.

A *context description* C is a collection of descriptive (non-behavioural) declarations about the types (e.g., employees and locations), and instances of entities present in an environment (e.g., *bob* and *r01*), and relations between instances, such as *bob* is entitled to access *r01* (*hasbadge(bob,r01)*).

Definition 3.1 (Context Description). A context description C is a tuple $\langle Y, I, \gamma, K\rangle$ where Y is a set of types, I a set of instances, $\gamma : I \to Y$ is function that assigns an instance in I to its type in Y, and K is a set of context relations over instances in I, such that for every $k \in K, k \subseteq I_1 \times ... \times I_n$.

We denote the universe of context relations as $\mathcal{K}$. A *context relation literal* is an expression of the form k or $\neg k$ for some $k \in \mathcal{K}$. Given a set of context relation literals KL, we write $\alpha(KL)$ to denote the set of unique context relations in KL.

Returning to our running example, a context description in this case includes types such as *Emp* and *Comp*, instances including *bob* and *m1* with assignments $\gamma(bob) = Emp$ and $\gamma(m1) = Comp$, and context relations such as *isLocatedIn(m1, r01)*, meaning that computer *m1* is placed in location *r01*, and *isStoredIn(doc, m1)*, meaning that *doc* is stored in *m1*.

A *behavioural description* $\mathcal{B}$ specifies the events that may occur within an environment. In a digital investigation setting, these may be at different levels of abstraction. Similar to [8, 11], we distinguish between two types of events to represent these levels of abstraction: primitive and complex. A *primitive event* represents the occurrence of an atomic action that can be observed by an investigator from a digital device (e.g., using a hard drive analysis tools). An example of primitive event can be *swipe_card(alice,nfc)* indicating the *nfc* reading *alice*'s card tag. A *complex event* indicates the execution of complex human activities and can involve one or more primitive events, other complex events and contextual conditions. For example, a complex event indicating *alice* entering room *r01* involves the following primitive events: *alice*'s card tag being read by the *nfc* reader and her entrance in *r01* being recorded by the *cctv*. These events can happen at the same time or in any order. The complex event also involves the following contextual conditions: *alice* is not inside *r01* and does not possess a badge to access *r01*. This is expressed through a composite definition.

Definition 3.2 (Composite Definition). Let $\mathcal{A}^p$ and $\mathcal{A}^c$ be the universe of primitive and complex events respectively. A *composite definition* is a tuple $\langle A^p, A^c, KL, L, \preceq, \lambda\rangle$ where $A^p \subseteq \mathcal{A}^p$, $A^c \subseteq \mathcal{A}^c$, $\alpha(KL) \subseteq \mathcal{K}$, L is a finite set of time-labels, $\preceq \subseteq L \times L$ is a partial order relation over L (that is reflexive, anti-symmetric and transitive) and $\lambda : L \to \mathcal{P}(A^p \cup A^c \cup KL)$ is a labelling function.

Let $\mathcal{D}$ be a set of composite definitions. We define a relation $\triangleleft \subseteq \mathcal{A}^c \times \mathcal{D}$ to associate a composite definition $d \in \mathcal{D}$ with the complex event $e \in \mathcal{A}^c$ it defines. In our example, the complex event *enter*(*alice*,*r01*) may be defined as *enter*(*alice*,*r01*) $\triangleleft\langle$ {*swipe_card*(*alice*, *nfc*), *cctv_access*(*alice*,*r01*,*cctv1*) }, $\emptyset$, {¬ *in*(*alice*, *r01*), *hasBadge*(*alice*,*r01*)}, $\{l_1, l_2\}$,$\emptyset$, {l_1 → {*swipe_card*(*alice*,*nfc*),¬ *in*(*alice*, *r01*), *hasBadge*(*alice*,*r01*)}, l_2 →{*cctv_access*(*alice*,*r01*, *cctv1*), ¬ *in*(*alice*, *r01*)}}$\rangle$.

For our example, to trigger complex event *enter*(*alice*, *r01*), primitive events *swipe_card*(*alice*,*nfc*) and *cctv_access*(*alice*,*r01*, *cctv1*) can occur in any order; however, both have to occur. When *swipe_card*(-*alice*,*nfc*) and *cctv_access*(*alice*,*r01*, *cctv1*) occur *alice* should not be in *r01*. Moreover *alice* should be authorised to access *r01* (*hasBadge*) when *swipe_card*(*alice*,*nfc*) occurs. Moreover, complex event *mount*(*usb1*,*m1*) event occurs when the system log in *m1* records the mounting of a storage device (primitive event *sys_mount*(*usb1*, *m1*)) while *alice* or *bob* are logged to *m1* (context condition *logged*(*e*,*m1*)). This is defined as *mount*(*usb1*,*m1*) $\triangleleft\langle$ {*sys_mount*(*usb1*, *m1*) }, $\emptyset$, {*logged*(*e*,*m1*)}, $\{l_1\}$,$\emptyset$, {l_1 → { *sys_mount*(*usb1*, *m1*), *logged*(*e*,*m1*)}}$\rangle$.

A behavioural description includes the composite definitions associated with the complex events that can occur in the environment. A behavioural description is formally defined as follows.

Definition 3.3 (Behavioural Description). A behavioural description $\mathcal{B}$ is a tuple $\langle \mathcal{A}^p, \mathcal{A}^c, \mathcal{K}, \mathcal{D}, \triangleleft\rangle$ such that for every $e \triangleleft d$, $e \in \mathcal{A}^c$, $d = \langle L_d, \preceq_d, \lambda_d, A^p_d, A^c_d, KL_d\rangle \in \mathcal{D}$, $A^p_d \subseteq \mathcal{A}^p$, $A^c_d \subseteq \mathcal{A}^c$ and $\alpha(KL_d) \subseteq \mathcal{K}$.

Complex events are expected to interact and bring about changes to the context in which they occur. To capture this effect, we adopt notions of fluents for event-driven systems [22, 31]. Given the set $\mathcal{K}$ of context relations, each $k \in \mathcal{K}$ is defined by two disjoint sets of complex events from $\mathcal{A}^c$ (called initiating and terminating events, respectively) and an initial value (true or false), written according to the following schema: $k \equiv \langle IN_k, TR_k, init_k\rangle$ such that $IN_k \cap TR_k = \emptyset$ and $IN_k \cup TR_k \subset \mathcal{A}^c$. The set of associations of this form are called *interaction definitions*, and are denoted $\mathcal{I}$. In our running example the interaction definition of context relation *in*(*alice*, *r01*) is defined by $\langle$\{*enter*(*alice*, *r01*)\}, \{*exit*(*alice*, *r01*)\}, *false*$\rangle$. This interaction indicates that context relation *in*(*alice*, *r01*) is initially false; it is initiated by complex event *enter*(*alice*, *r01*) and terminated by complex event *exit*(*alice*, *r01*).

From C, $\mathcal{B}$ and $\mathcal{I}$ we define an environment description.

Definition 3.4 (Environment Description). An environment description $\mathcal{E}$ is a tuple $\langle C, \mathcal{B}, \mathcal{I}\rangle$ where $C = \langle Y, I, \gamma, \mathcal{K}\rangle$ is a context description, $\mathcal{B} = \langle \mathcal{A}^p, \mathcal{A}^c, \mathcal{K}, \mathcal{D}, \triangleleft\rangle$ a behavioural description and $\mathcal{I}$ is a set of context relation definitions such that $\forall k \equiv \langle IN_k, TR_k, init_k\rangle \in \mathcal{I}.\ k \in \mathcal{K}$ and $IN_k \cup TR_k \subseteq \mathcal{A}^c$.

3.2 Histories

A *history* is a sequence of (concurrent) events that captures the evolution of an environment in which the digital devices and evidence sources operate [11]. It is *potential* if it refers to at least one event that has been speculated and *actual* if all the events have been observed from digital sources within the environment. In this paper, we focus on potential histories for defining preservation requirements.

A history may describe events at various levels. It is called a primitive (resp. complex) history, denoted σ (resp. ω), if all the events that appear in it are primitive (resp. complex). We write $\omega = ce_1, \ldots ce_n$ to denote a complex history where ce_i is the set of complex events occurring concurrently at position i, and similarly for a primitive history σ.

An environment description $\mathcal{E}$ is interpreted over a sequence of primitive and complex events (referred to as a hybrid history υ). Its satisfaction is determined with respect to the satisfaction of complex events' composite definitions in $\mathcal{B}$ w.r.t. to $\mathcal{I}$.

For the satisfaction of an event's composite definition, we consider the notion of a 'narration' (a total order over the partial order given in a complex event's definition). For a narration to be constructed, each complex event appearing in a definition is refined until all complex events are reduced to their primitive events and context relation literals. The result of this refinement procedure applied to definition d is a set of composite definitions $\delta(d)$. [2]

Given $\delta(d)$, a narration of d is captured with respect to one of the elements in $\delta(d)$. We will use the notation $\upsilon|_{A^p}$ (reps. $\upsilon|_{A^c}$) to denote the projection of υ over primitive (reps. complex) events in A^p (resp. A^c).

Definition 3.5 (Narration of Composite Definition). Let $\mathcal{B}$= $\langle \mathcal{A}^p, \mathcal{A}^c, \mathcal{K}, \mathcal{D}, \triangleleft \rangle$ be a behavioural description and $d = \langle L_d, \preceq_d, \lambda_d, A^p_d, A^c_d, KL_d \rangle$ a composite definition in $\mathcal{D}$. Let $\delta(d)$ be the set of definitions obtained refining d. A narration of d is a hybrid history $\sigma = he_1, \ldots, he_m$, if there exists a $d' \in \delta(d)$ and a total order $l_1 \prec \ldots \prec l_n$ over $L_{d'}$ such that:

- for all $l_i, l_j \in L_{d'}$, if $l_i \prec l_j$ then $a < b$ (where $1 \leq a, b \leq m$),
- $\lambda_{d'}(l_i)|_{A^p} = (he_a)|_{A^p}$
- $\lambda_{d'}(l_i)|_{A^c} = (he_a)|_{A^c}$.

where $\lambda(l)|_{A^p}$ and $\lambda(l)|_{A^c}$ denote the set of primitive events and complex events respectively assigned to time-label l.

For instance the following are three example narrations for *enter*(*alice, r01*)'s composite definition:

$$\upsilon_1 = (\{swipe\_card(alice, nfc), cctv\_access(alice, r01, cctv1), enter(alice, r01)\}_1)$$
$$\upsilon_2 = (\{swipe\_card(alice, nfc)\}_1, \{cctv\_access(alice, r01, cctv1), enter(alice, r01)\}_2)$$
$$\upsilon_3 = (\{cctv\_access(alice, r01, cctv1)\}_1, \{swipe\_card(alice, nfc), enter(alice, r01)\}_2)$$

Interaction descriptions are interpreted over complex histories. Given $k \equiv \langle IN_k, TR_k, init_k \rangle \in \mathcal{I}$, k is true at position b in a complex history $\omega = ce_1, \ldots, ce_b, \ldots, ce_n$ iff either the following holds:

- $init_k \wedge \forall a \in \mathcal{N}, e_{TR_k} \in TR_k.(0 < a < b) \rightarrow e_{TR_k} \notin ce_a$;
- $\exists a \in \mathcal{N}. (a < b) \wedge (e_{IN_k} \in ce_a \wedge \forall g \in \mathcal{N}, e_{TR_k} \in TR_k.((a < g < b) \rightarrow e_{TR_k} \notin ce_g)$;

otherwise it is said to be false. We assume histories in which terminating and initiating events for a context relation do not occur concurrently. We define below satisfaction of a complex event.

Definition 3.6 (Complex Event Satisfaction). Given an environment description $\mathcal{E}$=($\mathcal{B}$,$\mathcal{C}$,$\mathcal{I}$), a composite definition d associated with complex event e ($e \triangleleft d \in \mathcal{B}$) and a hybrid history υ, υ is said to satisfy $e \triangleleft d$ with respect to $\mathcal{E}$ if for every decomposition $\upsilon = xyz$, if $y = he_a, \ldots, he_g, \ldots, he_b$ is a narration of d with respect to $d' \in \delta(d)$ and order $l'_1 \prec \ldots \prec l'_j \prec \ldots \prec l'_n$ then:

- $e \in (he_b)|_{A^c}$
- if $kl \in \lambda_{d'}(l'_j)|_{KL}$ then $\upsilon|_{\mathcal{A}^c}, g \models kl$

where $\lambda(l)|_{KL}$ denotes the set of context relation literals assigned to time-label l.

The environment description $\mathcal{E}$ is said to be satisfied in a hybrid history if every complex event in A^c is satisfied in that history. We write $\Upsilon(\mathcal{E})$ to denote the set of hybrid histories that satisfy $\mathcal{E}$.

Figure 2 shows a hybrid history satisfying $\mathcal{E}$ of our example. We project the primitive and complex events composing the hybrid history onto primitive and complex histories, respectively. The primitive history represents the case in which (1) the *nfc* reads *alice*'s card tag, (2) the *cctv* records *alice* passing through the door of *r01*, and *m1* logs (3) the login performed by user *bob*, (4) the *mount* of storage device *usb1*, (5) the copy od the *doc* by user *bob*, and (6) the *unmount* of *usb1*. If a portion of the primitive history represents a narration of a composite definition associated with a complex event, such event is assumed to occur. For example, the sequence of primitive events at time 1 and 2 corresponds to narration υ_2 of the composite definition of *enter*(*alice,r01*). While the primitive event at time 4 represents a narration for the composite definition of event *mount*(*usb1, m1*). On top of Figure 2 we indicate the context relations that hold at each time instant and omit those that are not satisfied, e.g., *mounted*(*usb1,m1*) starts holding when complex event *mount*(*usb1,m1*) occurs and stops holding when complex event *unmount*(*usb1,m1*) happens.

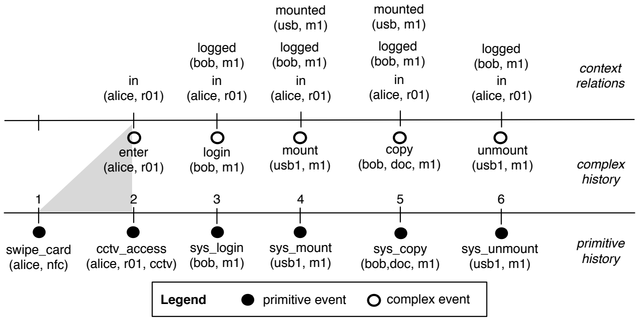


Figure 2: Example of hybrid history

3.3 Hypotheses

The term *hypothesis* in a digital investigation is a conjecture that may refer, for instance, to past events in the lifetime of digital devices [52]. In this paper, we focus on one type of hypothesis

[2] See https://github.com/lpasquale/minorityReport for an outline of a refinement algorithm for obtaining $\delta(d)$.

relevant to developing forensic-ready systems, the *environment construction hypothesis*. This form of hypothesis postulates about the feasibility of events occurrence and presence of contextual conditions of interests within the environment. It may be captured as an event's composite definition $h \triangleleft d$, where h is a complex event marking the satisfaction of a hypothesis, and with A^p in d being empty and A^c and KL containing only complex events and context relation literals respectively. The events and the contextual conditions expressed in the hypothesis represent how an incident may occur within the environment. The incident of our example refers to the unauthorised exfiltration of the sensitive document *doc*. One way in which the *doc* can exfiltrate is because an unauthorised copy to an external storage device was performed. This hypothesis is defined as

$$\begin{array}{l} \mathit{IllegalCopy} \triangleleft \\ \quad \langle \emptyset, \{\mathit{copy}(\mathit{bob}, \mathit{doc}, \mathit{m1})\}, \{\mathit{mounted}(\mathit{usb1}, \mathit{m1})\}, \\ \quad \{l_1\}, \emptyset, \{l_1 \rightarrow \{\mathit{mounted}(\mathit{usb1}, \mathit{m1}), \mathit{copy}(\mathit{bob}, \mathit{doc}, \mathit{m1})\}\}\rangle \end{array}$$

In other words, a copy of the document is performed while an external storage device (*usb1*) is mounted on *m1*.

Hypotheses are interpreted over *finite* complex histories. Their satisfaction is given by the definition below.

Definition 3.7 (Hypotheses Satisfaction). A hypothesis h (with definition $h \triangleleft d$) is said to be satisfied in a complex history ω at position b, i.e., $\omega, b \models h$, if there exists a decomposition $\omega = xyz$ such that $y = ce_a, ..., ce_g, ..., ce_b$ is a narration of d with respect to $d' \in \delta(d)$ and order $l'_1 \prec ... \prec l'_j \prec ... \prec l'_n$ and if $kl \in \lambda_{d'}(l'_j)|_{KL}$ then $\upsilon|_{\mathcal{A}^c}, g \models kl$.

We distinguish between *supportable* and *refutable* hypotheses in environment $\mathcal{E}$.

Definition 3.8 (Hypotheses Supportability and Refutability). Let $\Upsilon(\mathcal{E})$ be the set of hybrid histories satisfying $\mathcal{E}$. A hypothesis h (with definition $h \triangleleft d$) is said to be *supportable* in $\mathcal{E}$ if there exists a hybrid history $\upsilon \in \Upsilon(\mathcal{E})$ such that for some b, $\upsilon|_{\mathcal{A}^c}, b \models h$. It is said to be refutable if there exists a history $\upsilon \in \Upsilon(\mathcal{E})$ such that for all b, $\upsilon|_{\mathcal{A}^c}, b \not\models h$.

We sometime abstract away from the position b and write $\upsilon|_{\mathcal{A}^c} \models h$ for a history satisfying h at some point b. We will denote the set of hybrid histories in $\Upsilon(\mathcal{E})$ supporting at least one hypothesis in $\mathcal{H}$ as $\Upsilon^+(\mathcal{E})$ and those refuting every hypothesis in $\mathcal{H}$ as $\Upsilon^-(\mathcal{E})$. Returning to our example, the *IllegalCopy* hypothesis is supportable in our example environment ($\mathcal{E}$) since there exists a decomposition of a hybrid history (see Figure 2) satisfying $\mathcal{E}$, that yields a narration of the definition of the *IllegalCopy* hypothesis, i.e.,

$$\begin{array}{rcl} x & = & \{\mathit{swipe\_card}(\mathit{alice}, \mathit{nfc})\}_1, \\ & & \{\mathit{cctv\_access}(\mathit{alice}, \mathit{r01}, \mathit{cctv1}), \mathit{enter}(\mathit{alice}, \mathit{r01})\}_2, \\ & & \{\mathit{sys\_login}(\mathit{bob}, \mathit{m1}), \mathit{login}(\mathit{bob}, \mathit{m1})\}_3, \\ & & \{\mathit{sys\_mount}(\mathit{usb1}, \mathit{m1}), \mathit{mount}(\mathit{usb1}, \mathit{m1})\}_4 \\ y & = & \{\mathit{sys\_copy}(\mathit{bob}, \mathit{doc}, \mathit{cctv1}), \mathit{copy}(\mathit{bob}, \mathit{doc}, \mathit{cctv1})\}_5, \\ z & = & \{\mathit{sys\_unmount}(\mathit{usb1}, \mathit{m1})\}_6 \end{array}$$

such that $\upsilon|_{\mathcal{A}^c}, 5 \models \mathit{mounted}(\mathit{usb1}, \mathit{m1})$.

As we will see later in Section 4, we are interested in *minimal* hybrid histories that satisfy a hypothesis. We define minimality of histories with respect to hypotheses as follows.

Definition 3.9. [Minimally Supportive Histories] Let $\Upsilon(\mathcal{E})$ be a set of hybrid histories satisfying $\mathcal{E}$ and h be a hypothesis supportable by $\mathcal{E}$. The hybrid history $\upsilon = he_1, ..., he_m \in \Upsilon^+(\mathcal{E})$ is said to be minimally supportive of h in $\mathcal{E}$ iff the history υ^* obtained be removing any primitive event $a \in A^p$ from $(he_i)|_{A^p}$ of υ is in $\Upsilon(\mathcal{E})$ and no longer supports h.

For instance, the hybrid history in Figure 2 is not minimally supportive of the hypothesis *IllegalCopy* since the history obtained by removing $\mathit{sys\_unmount}(\mathit{usb1}, \mathit{m1})$ from $\upsilon_6|_{A^p}$ still satisfies *IllegalCopy*. We sometimes write $\mathit{min}(\upsilon, h)$ (resp. $\mathit{min}(\Upsilon, h)$) as a shorthand for the minimally supportive history υ^* (resp. histories) of h obtained from υ.

4 PRESERVATION SPECIFICATIONS

We are concerned with deriving specifications *PS* for a forensic readiness controller comprising domain pre- and post-conditions as well as required pre- and trigger-conditions, expressed in LTL. These conditions control the execution of operations of the form *preserve*(*a*,*ts*)—where *a* indicates the occurrence of a primitive event in the environment, and *ts* marks the time-stamp (from the system clock) at which the occurrence was observed by the FR controller. We consider ts_i to be an abstraction over real-time clock variables that may be obtained following techniques such as [13, 28]. The generation of such abstractions is outside the scope of the paper. We assume an ordered set of timestamps to be isomorphic to the set of natural numbers.

The domain pre-condition of operation *preserve*(*a*,*ts*) specifies that this operation cannot take place if the occurrence of event *a* at *ts* has already been preserved. The domain post-condition specifies that operation *preserve*(*a*,*ts*) ensures preservation of the occurrence of event *a* at *ts* in the next time instant. We assume these ar given for each operation. We Assertions 1 and 2 specify, respectively, the domain pre- and post-conditions of operation $\mathit{preserve}(\mathit{sys\_copy}(e, d, m), \mathit{ts})$

$$\forall \mathit{ts} : \mathit{Timestamp}, e : \mathit{Emp}, d : \mathit{Doc}, m : \mathit{Comp}$$

$$\mathrm{G}(\mathit{preserved}(\mathit{sys\_copy}(e, d, m), \mathit{ts}) \rightarrow \neg \mathit{preserve}(\mathit{sys\_copy}(e, d, m), \mathit{ts})) \tag{1}$$

$$\mathrm{G}(\mathit{preserve}(\mathit{sys\_copy}(e, d, m), \mathit{ts}) \rightarrow \mathrm{X}\mathit{preserved}(\mathit{sys\_copy}(e, d, m), \mathit{ts})) \tag{2}$$

Required pre-conditions are assertions that condition the execution of *preserve*(*a*,*ts*) upon having received notification about the occurrence of a primitive event in the environment, *receive*(*a*, *ts*). Required trigger-conditions are conditions upon the (non-)preservation of other primitive events. An example of a preservation specification of operation $\mathit{preserve}(\mathit{sys\_copy}(e, d, m), \mathit{ts})$ is

$$\forall \mathit{ts} : \mathit{Timestamp}, e : \mathit{Emp}, d : \mathit{Doc}, m : \mathit{Comp}$$

$$\mathrm{G}(\neg \mathit{received}(\mathit{sys\_copy}(e, d, m), \mathit{ts}) \rightarrow \mathrm{X}\neg \mathit{preserve}(\mathit{sys\_copy}(e, d, m), \mathit{ts})) \tag{3}$$

$$\begin{array}{l} \forall \mathit{ts} : \mathit{Timestamp}, e : \mathit{Emp}, d : \mathit{Doc}, m : \mathit{Comp} \\ \exists \mathit{ts}_1, \mathit{ts}_2 : \mathit{Timestamp}.\ \mathit{ts}_1 < \mathit{ts}_2 \wedge \mathit{ts}_2 < \mathit{ts} \\ \mathrm{G}(\mathit{received}(\mathit{sys\_copy}(e, d, m), \mathit{ts}) \wedge \\ \mathit{preserved}(\mathit{sys\_login}(e, d, m), \mathit{ts}_1) \wedge \mathit{preserved}(\mathit{sys\_mount}(s, m), \mathit{ts}_2) \\ \wedge \nexists \mathit{ts}_3, \mathit{ts}_4 : \mathit{Timestamp}.\ (\mathit{ts}_3 > \mathit{ts}_1 \wedge \mathit{ts}_3 \leq \mathit{ts} \wedge \mathit{ts}_4 > \mathit{ts}_2 \wedge \mathit{ts}_4 \leq \mathit{ts} \wedge \\ \mathit{preserved}(\mathit{sys\_logout}(e, m), \mathit{ts}_3) \wedge \mathit{preserved}(\mathit{sys\_unmount}(s, m), \mathit{ts}_4)) \rightarrow \\ \qquad \mathrm{X}\mathit{preserve}(\mathit{sys\_copy}(e, d, m), \mathit{ts})) \end{array} \tag{4}$$

Assertion 3 specifies the required pre-condition, i.e., receiving notification of the occurrence of *sys_copy*(*e*,*d*,*m*). Assertion 4 expresses a trigger-condition forcing the FR controller to preserve occurrence of $sys\_copy(e, d, m)$ if it has already preserved information about an employee's logging onto a computer and the mounting of a storage device on that computer, but no subsequent occurrence about the employee logging out or unmounting of the storage device is recorded. The preservation specification *PS* defines a FR controller's storage capacities as a set of executable sequences of *preserve* operations of the form

$$\pi = \{preserve(a_1^1,\ ts_1),\ \ldots,\ preserve(a_1^k,\ ts_1)\}_1,\ \ldots,$$
$$\{preserve(a_m^1,\ ts_m),\ \ldots,\ preserve(a_m^r,\ ts_m)\}_m$$

We say that π is a potential log if π satisfies *PS* (according to standard trace semantics of LTL) and for each $preserve(a_j^i, ts_j) \in \pi(j)$, its required trigger-condition is non-vacuously satisfied in π at position j. The notation $\pi(j)$ indicates the set of operations that occur at position j.

We restrict our definition of preservation specifications to those that ensure the *minimality* and *relevance* of all potential logs to hypotheses under consideration. Such a specification is referred to as forensic-ready. To express forensic-ready preservation specifications, we first consider the notion of a specification covering potential histories.

Definition 4.1. [Specification Coverage] Let *PS* be a preservation specification and $\upsilon = (he_1, ..., he_n)$ a hybrid history. Then *PS* is said to *cover* υ, iff there exists a potential log $\pi = (fe_1, ..., fe_n) \in \Pi(PS)$ isomorphic to $\upsilon|_{\mathcal{A}^p}$, i.e., for every primitive event $a \in (he_i)|_{\mathcal{A}^p}$, $preserve(a, ts_i) \in fe_i$.

The isomorphism with respect to potential histories guarantees the preservation of events related to an incident. Furthermore, since the isomorphism is defined with respect to minimally supportive histories of hypotheses in $\mathcal{H}$ this ensures minimality of preserved event occurrences. It also, together with the requirement for hypotheses $\mathcal{H}$ to be refutable by potential histories of $\mathcal{E}$, $\Upsilon^-(\mathcal{E})$, supports relevance of events stored through *preserve* operations.

Definition 4.2. [Forensic-ready Specification] Let $\mathcal{E}$ be an environment description and $\mathcal{H}$ a hypothesis that is both supportable and refutable in $\mathcal{E}$ by $\Upsilon^+(\mathcal{E})$ and $\Upsilon^-(\mathcal{E})$ respectively. Let *PS* be a preservation specification. Then *PS* is said to be forensic-ready with respect to $\mathcal{H}$ in $\mathcal{E}$ iff *PS* covers every history in $min(\Upsilon^+(\mathcal{E}), h)$ for every $h \in \mathcal{H}$ and does not cover any $\Upsilon^-(\mathcal{E})$.

Any FR controller whose specification is forensic-ready with respect to $\mathcal{H}$ in $\mathcal{E}$ is sufficient to guarantee evidence preservation requirements of relevance and minimality.

5 SPECIFICATION GENERATION

Based on our formulation above, we propose a systematic approach (Figure 3) for generating forensic-ready preservation specifications. Our approach takes as input an environment description $\mathcal{E}$, a set of speculative incident hypotheses $\mathcal{H}$, elicited, for instance, by a domain expert, and an initial preservation specification *PS*, written in LTL, which contains domain pre- and post-conditions of preservation operations. We assume that the description of the environment is correct and the speculative hypotheses of concern are known at design-time. The approach provides as output either: (i) a confirmation that (some) hypotheses are not supportable in the environment; (ii) a confirmation that the FR controller does not have the capabilities to ensure the forensic-readiness of its preservation specification; or (iii) a preservation specification that is guaranteed to be forensic-ready with respect to $\mathcal{H}$ in $\mathcal{E}$. The approach comprises three phases as described below.

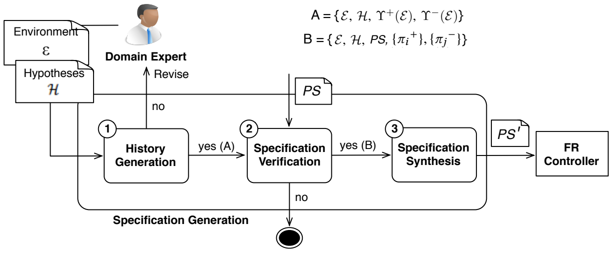

Environment
ε
Hypotheses
$\mathcal{H}$
Domain Expert
Revise
no
A = {$\mathcal{E}$, $\mathcal{H}$, $\Upsilon^+(\mathcal{E})$, $\Upsilon^-(\mathcal{E})$}
B = {$\mathcal{E}$, $\mathcal{H}$, PS, {π_i^+}, {π_j^-}}
PS
1
History Generation
yes (A)
2
Specification Verification
yes (B)
3
Specification Synthesis
PS'
FR Controller
no
Specification Generation

Figure 3: Specification generation approach.

1) History Generation. In this phase, we search for hybrid histories $\Upsilon^+(\mathcal{E})$ and $\Upsilon^-(\mathcal{E})$ that minimally support and that refute $\mathcal{H}$, respectively. The existence of histories in $\Upsilon^+(\mathcal{E})$ ensures that the hypotheses of interest are feasible within the intended environment. If $\Upsilon^+(\mathcal{E})$ is empty, this means that either the hypothesis cannot occur within the environment described, and thus it will not require to be considered during a digital investigation, or that the environment description and/or the speculative hypotheses are incorrect and need revision. The histories $\Upsilon^-(\mathcal{E})$ operate as a proxy for the synthesis phase to ensure only relevant event occurrences are preserved.

2) Specification Verification. Given the generated $\Upsilon^+(\mathcal{E})$, we check if each history is potentially covered by the preservation specification, i.e., there exists a corresponding potential log in $\Pi(PS)$. If some history in $\Upsilon^+(\mathcal{E})$ is not, then this may be owing to one of two cases: (i) the FR controller and the digital devices do not have the capabilities needed to, respectively, preserve the potential logs and monitor the relevant events; or (ii) they do but require an operational preservation specification to be synthesised to ensure their preservation. In the case of the former, the approach terminates, indicating a need for additional capabilities. In the latter case, corresponding potential logs $\{\pi_i^+\}$ and $\{\pi_j^-\}$ are produced and passed onto the third phase.

3) Specification Synthesis. The synthesis phase considers $\mathcal{E}$, $\mathcal{H}$, *PS*, $\{\pi_i^+\}$ and $\{\pi_j^-\}$. It searches, within a space of candidate expressions restricted to safety LTL expressions, a new set of required pre- and trigger-conditions that would prescribe the preservation of all potential logs $\{\pi_i^+\}$ but not in $\{\pi_j^-\}$. These are added to *PS*. The new specification *PS'* is given as provided to the FR controller that is responsible for its enactment. Note the steps above are conducted for a set of given hypotheses. If new ones are provided, then a new specification must be generated.

6 TOOL IMPLEMENTATION

As a proof of concept, we have implemented a prototype tool[3] for synthesising forensic-ready preservation specifications. Our solution uses a (i) declarative language based on the Event Calculus (EC) logic program [31] to represent and reason about the environment

[3] The source code of the tool is publicly available at https://github.com/lpasquale/kEEPER/tree/keeper_CLI.

descriptions, speculative hypotheses, preservation specifications and potential histories and logs in a uniform way, (ii) an off-the-shelf Boolean constraint solver for logic programs, called clingo [20], to compute potential histories and logs that satisfy or refute hypotheses, and (iii) and a logic-based learner, called XHAIL [41], to synthesise preservation specifications that cover all histories supportive of a hypothesis. Our choice of EC logic program as a language is due to its successful deployment in the context of requirements operationalisation [4, 5] and reasoning about evidence in digital investigations [56]. The encoding of the input as well as the execution of the three phases in Section 5 are done automatically. The user is expected to provide the initial input, and the maximum length of the potential histories to be considered in the approach. Our use of the solver as the underlying history generation and specification verification engine is motivated by its capacity to handle difficult (NP-hard) search problems for EC programs. For encoding of preservation specifications, we follow the translation in [5]. Details of the encoding for the environment description and hypotheses are available at https://github.com/lpasquale/kEEPER/tree/keeper_CLI/RunningExample. Although the approach is demonstrated for a particular language, the principles behind it could be applied to other formalisms and solvers.

In brief, the history generation phase first tries to find two sets of models of the program $L_{\mathcal{E}} \cup L_{\mathcal{H}}$. Each element in the first set is a model of $L_{\mathcal{E}} \cup L_{\mathcal{H}}$ that comprises a history L_{v^+} of maximum length n that is minimally supportive of a hypothesis in $\mathcal{H}$. Each element in the second set contains a history L_{v^-} of length n that refutes all hypotheses in $\mathcal{H}$. This is done by solving a constraint that requires a hypothesis $L_h \in L_{\mathcal{H}}$ to be satisfied by at least one potential history consistent with $L_{\mathcal{E}} \cup L_h$. The solver searches for the optimal solution being defined as the fewest event occurrence in a history which equates to the minimally supportive history of h. We denote the set of minimally supportive histories as $L_{\Upsilon^+(\mathcal{E})}$ and the minimally refuting histories as $L_{\Upsilon^-(\mathcal{E})}$.

The specification verification phase considers the histories $L_{\Upsilon^+(\mathcal{E})}$ and $L_{\Upsilon^-(\mathcal{E})}$, program $L_{\mathcal{E}} \cup L_{\mathcal{H}}$ and program L_{PS}. This phase performs several calls to the solver to check for consistency of each history $L_{\Upsilon^+(\mathcal{E})}$ and $L_{\Upsilon^-(\mathcal{E})}$ respectively with the specification L_{PS} (phase 2). The solver searches for models that satisfy the program $L_{\mathcal{E}} \cup L_{\mathcal{H}} \cup L_{PS} \cup L_v$ (for each $L_v \in L_{\Upsilon^+(\mathcal{E})} \cup L_{\Upsilon^-(\mathcal{E})}$) and a constraint requiring there to be an isomorphic potential log $L_{\pi_i^+}$ in the model. If a potential log for a supportive history cannot be found, then the program is unsatisfiable. In this case the approach outputs those potential histories to the user for further consideration (e.g., amending the FR controller's capabilities).

Otherwise all computed potential logs $L_{\Pi^+(PS)}$ and $L_{\Pi^-(PS)}$ that correspond to histories in $L_{\Upsilon^+(\mathcal{E})} \cup L_{\Upsilon^-(\mathcal{E})}$ respectively and the program $L_{\mathcal{E}} \cup L_{PS}$ are passed to a logic-based learner. The aim of the learner is to search through a candidate space (given by a language bias as in [5]) to compute required pre-conditions L_{ReqPre} and required trigger-conditions $L_{ReqTrig}$ such that $L_{\mathcal{E}} \cup L_{PS} \models L_{\Pi^+(PS)}$ where $\models$ is an entailment operator defined with respect to stable model semantics [21], whilst ensuring that $L_{\mathcal{E}} \cup L_{PS} \not\models L_{\pi^-}$ for any $L_{\pi^-} \in L_{\Pi^-(PS)}$. The programs $L_{ReqPre} \cup L_{ReqTrig}$ are then translated back to LTL following the method described in [5].

7 EVALUATION

Our evaluation aims to assess whether the synthesised preservation specification prescribes to preserve i) *relevant events* and ii) *the minimal amount of events* necessary to support speculative hypotheses of potential incidents. To achieve this aim, we apply our prototype tool to two case studies publicly available for research and training purposes. Each case study comprises data that would normally be available to an investigator for examination (when a FR controller is not implemented). The investigator can use this data to explain, if possible, how a particular incident occurred. For each incident, we manually modelled the environment and the speculative hypotheses in EC. From this, we used our tool to generate preservation specifications automatically. The EC models of the case studies and the generated specifications are available online[4].

We compare the events that the generated specification prescribes to preserve with those that would be available to an investigator when the system that does not satisfy evidence preservation requirements, i.e. the information available to an investigator is represented by the data-sets provided with the case studies. To assess relevance, we verify whether our specification prescribes to preserve events that were relevant to satisfy the speculative hypotheses. We also check if occurrence of those events can be inferred from the available data-set. To assess minimality we verify that our approach prescribes preserving fewer events. In particular, we compare the number of events that our approach prescribes to preserve with those that can be inferred from the data-set. We also measure the number of events, whose occurrence can be inferred from the data-set, which were irrelevant to support the satisfaction of the hypotheses.

7.1 Relevance and Minimality

The first incident scenario we considered is set in a university, where students and academic staff can send emails by using the university and students' residence internal network. The model of the environment includes different agents who can be academics or students, and can teach or attend courses, respectively. It also includes locations, such as university and students residences, routers (each of them placed in a location), emails and their corresponding sender/recipient email and IP addresses. We also model whether an email address is a university address for staff and students or it is an external address.

The primitive events we model cover events whose occurrence can be inferred from the data-set[5]. This includes the TCP packets captured from the routers located inside the students' residence. Therefore, we consider the routers as digital sources within the environment and use primitive events to represent network data streams. We model the following primitive events: IMAP/POP network traffic (we indicate primitive events related to emails sent from external addresses to an academic as *SUE*); incoming HTTP traffic (we indicate HTTP messages used to set-up a cookie as *SC*); general outgoing HTTP traffic (*EM*) and specific outgoing HTTP traffic towards anonymous email services (*SAE*). Some of the complex events we model include: (i) emails received by a specific email address (ii) cookie setting from an external address to an IP; (iii)

[4] https://github.com/lpasquale/kEEPER/tree/keeper_CLI

[5] http://digitalcorpora.org/corpora/scenarios/nitroba-university-harassment-scenario

sending of HTTP messages from an IP address and a browser agent; (iv) sending of anonymous emails from an IP address and a browser agent. The complex event indicating setting of a cookie initiates the state *cookieSet* for a specified email and IP address.

An incident of concern is related to the receipt of harassment emails by academics. The following speculative hypotheses were constructed: *h1*: an email is sent to an academic by someone using an external address; *h2*: an anonymous email is sent by an individual who can be identified for example through the cookie and his/her browser agents; *h3*: an anonymous email is sent by an individual who cannot be identified. *h1* is satisfied when complex event (i) takes place for which the sender email address is external and the recipient email address is owned by a university staff member. For supporting *h1*, the implemented specification recommended preserving all incoming IMAP/POP network traffic (*SUE*) related to emails sent from external addresses to an academic. *h2* is satisfied when a cookie is set for a specific email and IP address, complex event (iii) takes place for which HTTP traffic originates from the same IP address with which the cookie is associated, and subsequently complex event (iv) takes places for which the IP address and the browser agent have been previously associated with outgoing HTTP traffic. For *h2*, the specification required preserving (a) incoming HTTP traffic adopted to set-up a cookie (*SC*), (b) outgoing HTTP traffic from the same address to which the cookie was set (*EM*) and (c) outgoing HTTP traffic to send anonymous emails (*SAE*). *h3* is satisfied when complex event (iv) takes place. Therefore, for *h3* the specification requires preserving all *SAE* events.

Table 1 shows the total time necessary to generate a specification for each hypothesis, and the time required by each phase of the approach: histories generation (HG), specification verification (SV) and specification synthesis (SS). For each hypothesis, the number of supporting histories (out of the total number generated) and negative histories necessary to compute a specification, including the maximum length of the histories, are shown. A higher number of positive and negative histories could have been given as input to the synthesis activity without affecting the generated specification. The maximum time was taken for the most complex hypothesis (*h*2) which also required the provision of negative histories.

Table 1: Performance in the harassment case study.

	Instances			Execution time (s)			
	#Pos	**#Neg**	**Length**	**HG**	**SV**	**SS**	**Total**
h1	1 / 4	0	1	∼0	0.01	0.23	0.24
h2	1 / 32	4	3	0.08	0.19	39.913	40.183
h3	1 / 8	0	1	0.01	0.03	0.301	0.341

We implemented the specification of a FR controller able to extract data stream from the data-set. Extracted data streams can support *h2* since an incoming set-cookie message associated with `jcoach@gmail.com` and received by IP `192.168.015.004` was preserved. Outgoing HTTP messages from the same IP address and associated with a `Mozilla` browser have also been recorded; the same browser appears to have been used to send the anonymous email. This supports our theory that our approach would preserve data that might represent relevant evidence if such an incident were to occur. This would support investigators in prioritising their efforts, while ensuring that other events related to alternative scenarios would have been preserved if such scenarios occurred. Moreover, our approach also prescribes to preserve events that might not be proactively retained by digital sources. For example, the data-set does not include events about IMAP/POP network traffic (*SUE*) necessary to support hypothesis *h1*. This might be due to the fact that network traffic was collected for a limited amount of time and was not retained.

To assess minimality we compare the total number of events whose occurrence can be inferred from the data-set with those that our specification would prescribe to preserve. The full data-set includes 577, 760 data streams (application level messages) exchanged in 15, 508 communications between different IP addresses. The number of data streams corresponds to the total number of events that an investigator would normally have to examine. The number and type of event that our specification prescribed to preserve for each hypothesis is shown in Table 2; the total number of events is only 0.71% of the data streams in the entire data-set. Moreover, not all the events preserved were necessary to support the hypotheses. For our scenario, only 956 data streams corresponding to HTTP traffic originating from the `Mozilla` browser were necessary to support *h2*. Therefore, although our specification consistently reduces the amount of data to be analysed by an investigator, it does not completely ensure the minimality requirement since 2874 (69%) data streams were not relevant to support *h*2.

Table 2: Number of events preserved.

		SUE	**SC**	**EM**	**SAE**
# Events	h1	0	–	–	–
	h2	–	2	3830	300
	h3	–	–	–	
		Total: 4132 events			

We also applied our approach to a more complex corporate exfiltration scenario[6]. The model of the environment includes the company's employees, their email addresses, computers, employees' access rights to computers, storage devices that could be mounted and programs that are installed on each computer. The available data-set consists of an image of the hard drive of a Windows machine. Thus, primitive events we modelled represent changes in the file system of a Window machine that can be observed from a hard drive image. These include users' logins, mount and unmount of devices, installation of programs, and sent or received emails. Owing to space, we will not provide details of the model and the specification generated for all the hypotheses of this example and refer the reader to the project webpage.

An incident of concern is related to the exfiltration of a confidential document of a company from the computer of the chief financial officer (cfo). Six hypotheses were constructed for this incident. Examples of these hypotheses are: *h2*: the document is sent via email to an external email address and *h5*: the document is copied while an external storage device is mounted. To support *h2* the specification requires preserving user logins to a computer in which the document is stored and, while a user is logged, sending of emails to a non-corporate address including the confidential document attached. As *h5* is equal to the hypothesis of our running example, it lead to the same preservation specification. For this

[6] http://digitalcorpora.org/corpora/scenarios/m57-jean

incident scenario the hypotheses we modelled were more complex and a higher number of supporting and refuting histories were generated. This increased the time the approach took to learn a specification. Table 3 shows the time to generate a preservation specification for each hypothesis.

We manually acquired the events identified from the computer hard drive using Autopsy [10]. These were not sufficient to support any of the hypotheses because some of the events that our approach prescribes to preserve are not retained in a computer hard drive image. For example, we cannot support hypothesis *h*2 speculating that the document might be sent as an email attachment by an employee to a non-corporate email address. In particular, although an event in the data-set indicates that an email with the attached confidential document was sent from the cfo's email address (`jean@m57.biz`) to an external address (`tuckgorge@gmail.com`), we cannot conclude which user was logged on the machine since the data-set only provides information about the last user login. A similar situation arise with hypothesis *h5* speculating that the document may be copied to an external device, because mounting of a storage device and copy of a file are events that are not retained. If our specification was implemented it would have ensured preservation of events necessary to support *h*2 and *h*5 when they occurred.

Table 3: Performance in the exfiltration case study.

	Instances			Execution time (s)			
	#Pos	**#Neg**	**Length**	**HI**	**SV**	**SG**	**Total**
h1	2 / 12	2	2	0.01	0.05	7.756	7.816
h2	1 / 4	1	2	0.01	0.05	1.852	1.912
h3	4 / 18	14	3	0.1	0.28	1733.82	1734.2
h4	4 / 16	9	3	0.5	0.18	894.197	894.877
h3	4 / 18	14	3	0.1	0.28	1733.82	1734.2
h4	4 / 16	9	3	0.5	0.18	894.197	894.877
h5	1	4	3	0.05	0.2	43.851	44.101
h6	1	4	3	0.5	0.21	170.356	171.066

To assess minimality, Fig. 4 shows, for each hypothesis, the number of events our approach would have preserved from the hard drive image. We compared these figures with those that an investigator would have examined from the data-set (No-FR). Our approach would have resulted in significantly fewer events to be examined for hypotheses *h1–h6*. For example, to support *h2* it would be necessary to identify the mail clients among the installed applications (133), inspect all the outboxes associated with the accounts registered with the mail clients (23 emails for the cfo's outbox and no emails for the Administrator outbox) and identify the users' last login (3). We cannot claim the same for *h5* the generated specification requires preserving users' last logins, mounted devices and file access operations, which are not present in the data-set.

7.2 Discussion

The paper aims to ensure relevant events are preserved that may serve as evidence, thus reducing the amount of data investigators would have to search through. This is what is evaluated. The paper is not concerned with reactive investigations nor aiding open-ended investigations. We make the assumption that the speculative hypotheses of an incident are given in advance. Therefore a forensic-ready system will be prepared to investigate only the incidents known a-priori. This the assumption on which forensic readiness guidelines for organisations are based on. Training experts (e.g., system/security administrator) to identify these is part of the business requirements for implementing forensic measures [45] and is outside the scope of our work. Furthermore, as the environment and the speculative hypotheses are expected to be known a priori, there is a risk that this knowledge can be used to thwart the forensic-ready system itself — an individual might adjust her behaviour to avoid preservation of events indicating her involvement in an offense. Thus, applications of our approach would require maintenance of confidentiality of the system specification.

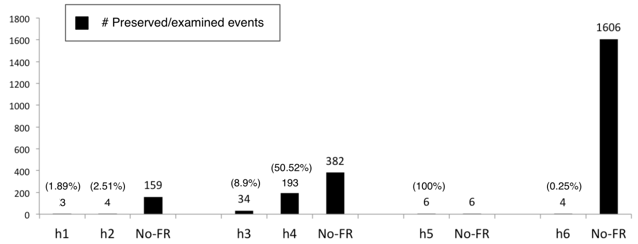


Figure 4: Number of events to be examined.

The performance of our approach decreases when 'richer' positive and negative histories are used [5] (a saturation point is reached for 18 histories). This is caused by the increase in the EC model size when grounded. The time taken to synthesise a specification increases linearly with the length of the considered preservation histories. A saturation point is reached with histories having length 11. Note that the scalability results purely depend on the open-source prototype tool[7] used to support the specification synthesis. This could be significantly improved by deploying learning techniques for context-dependent learning [32], and distributed reasoning [34].

To show that our formalisation could yield a practical solution, we provide a proof-of-concept implementation, putting aside usability issues. The definition of the model of the environment and the hypotheses of the university harrassment and the corporate exfiltration scenarios required 2 and 3.5 days of work, respectively, to one of the paper's authors. We are developing a graphical interface that would mask the complexity of the formal specification and help practitioners represent potential incidents and how they may occur in the environment. Such graphical interface is based on the model-driven engineering principle to hide the complexity of the EC language used to represent the environment and hypotheses within a model. A model-based representation has the potential to ensure correctness of the models by-design and encourage re-usability of the environment and hypotheses among experts.

8 RELATED WORK

Existing research on forensic readiness has mainly focused on identifying high-level strategies which organisations can implement to be forensic-ready. For example, Elyas et al. [15] use focus groups to elicit required forensic readiness objectives (e.g., regulatory compliance, legal evidence management) and capabilities (organisational factors and forensic strategy). Reddy and Venter [42] present a forensic readiness management system taking into account event

[7] https://github.com/stefano-bragaglia/XHAIL

analysis capabilities, domain-specific information (e.g., policies procedures and training requirements), and costs (e.g., staff, infrastructure and training costs). However, none of these approaches has addressed the problem of how to implement forensic readiness in existing IT systems— inspite of the standardisation of forensic readiness processes (ISO/IEC 27043:2015[8]) which prescribes the planning and implementation of pre-incident collection and analysis of evidence activities.

Shield et al. [48] propose performing continuous proactive evidence preservation. However, in large scale environments like cloud systems, monitoring all potential evidence is not a viable solution, as it might be cumbersome to analyse. Pasquale et al. [37] propose a more targeted approach, where evidence preservation activities aim to detect potential attack scenarios that can violate existing security policies. However, this approach is less selective as it prescribe to preserve any type of event within a history leading to an incident, independently of other events that have previously occurred or preserved. Existing work on data extraction for investigative purposes, such as E-Discovery [25], although supporting retrieval of data for an investigation, it does not provide a solution to engineer a forensic-ready system prescribing what data should be preserved depending on its relevance to future investigations.

With the growth of digital forensics as a discipline, interest in rigorous approaches has increased. For example, Carrier [11] provides guidelines about the types of hypotheses that should be formulated and the analysis to be performed to verify those hypotheses during a digital investigation. Others [1, 8, 27, 47] have focused on providing a unified representation of heterogeneous log events to automate event reconstruction. Similar to us, all these approaches distinguish between primitive events having a direct mapping to raw log events and complex level events, which can be determined by the occurrence of primitive ones. Formal techniques have also been used to represent and analyse the behaviour of the environment in order to identify the root causes that allowed an incident to occur [50] or possible incident scenarios [23]. Other work is specialised on identifying attackers' traces (e.g., evidence and timestamps improperly manipulated by an attacker), from violations of invariant relationships between digital objects [49] or by applying model checking techniques on a set of events expressed in a multi-sorted algebra [7]. However, none of this work addresses the problem of how hypotheses can be expressed formally, how they relate to sequences of primitive and complex events supporting them and how to achieve preservation requirements.

The requirements engineering community [26, 33, 53, 55] proposed numerous techniques for modelling security and privacy requirements to enable the design of systems less vulnerable to potential attacks and privacy breaches. However, only preliminary attempts have been made towards engineering forensic-ready systems [6, 38] and investigating how forensic-readiness requirements are considered during systems development lifecycles [24]. Although preservation requirements can be considered as a specific type of monitoring requirements [17, 18, 44, 46], the nature of the specifications for forensic-ready systems is different in its scope (environment and hypotheses) and characteristics. These are aspects that have not been covered in previous work.

Recent studies in program analysis, such as [19, 57], have highlighted the importance of providing software developers with automated support in making logging decisions and difficulty in constructing specifications to guide logging behaviour. In [57] for instance, the authors present a method for learning what and when to log from past logs of software developers. This differs conceptually from what we present here since incident-related histories are domain specific, showing how particular hypotheses may be met within particular environments. For forensic-ready systems, justification of preservations need to be made explicit and in readable form, which is supported by the learning technique that we deploy. We believe however that our approach could help software developers in making informed decisions and insights on what logs to preserve to enhance forensic-readiness of systems. Closest to our work with in a forensic setting is that of [3, 30]. However [3] focuses on defining "ideal" logging preferences for databases that is independent of the incidents of concern and hence still poses a risk of inadequate logging. The work of [30] instead is limited to eliciting from natural language descriptions of software artefacts the set events (as verb-object pairs) and an empirical classification of such events to determine logging requirements.

9 CONCLUSION

This paper represents a first step towards a rigorous approach to developing forensic-ready systems. We defined a framework for formalising evidence preservation requirements of such systems. We use this to synthesise specifications that guarantee a minimal amount of data, constituting potentially relevant evidence to support given speculative hypotheses of incidents of concern, is preserved. We also provided a proof-of-concept implementation that has been evaluated on two incident scenarios. Our results demonstrate that our approach preserves relevant events and provides insight into whether existing software/devices have the necessary capabilities for preserving evidence. Moreover, the size of preserved data is smaller than what would have been examined during an investigation otherwise. Our approach does not propose replacing the role of engineers nor investigators. It also assumes that domain experts are involved in modelling the environment and selecting the relevant histories to be covered by the preservation specifications.

In the future, we plan to investigate how our approach may be adapted to dynamic situations at run-time in which environments and hypotheses may change over time. We are developing a graphical designer aimed to facilitate the practitioners' task of designing the model of the environment and generation of hypotheses. Finally when generating a preservation specification, we will consider systematic approaches for synthesis when conflicts with other requirements, such as legal requirements, are present which may forbid preserving relevant data for privacy reasons.

ACKNOWLEDGEMENTS

This work is supported by ERC Advanced Grant no. 291652 (ASAP), SFI Grants 10/CE/I1855, 13/RC/2094 and 15/SIRG/3501, and the Imperial College Research Fellowship.

[8] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44407

REFERENCES

[1] J. Abbott, J. Bell, A. Clark, O. De Vel, and G. Mohay. Automated Recognition of Event Scenarios for Digital Forensics. In *Proc. of the 2006 ACM Symposium on Applied Computing*, pages 293–300. ACM, 2006.

[2] AccessData. FTK – AccessData Digital Forensics Software. http://www.accessdata.com/products/digital-forensics/ftk, 2014.

[3] O. M. Adedayo and M. S. Olivier. Ideal log setting for database forensics reconstruction. *Digit. Investig.*, 12(C):27–40, 2015.

[4] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Learning operational requirements from goal models. In *Proc. of the 31st International Conference on Software Engineering*, pages 265–275, 2009.

[5] D. Alrajeh, J. Kramer, A. Russo, and S. Uchitel. Elaborating requirements using model checking and inductive learning. *IEEE Trans. Software Eng.*, 39(3):361–383, 2013.

[6] F. Alrimawi, L. Pasquale, and B. Nuseibeh. Software Engineering Challenges for Investigating Cyber-physical Incidents. In *Proc. of the 3rd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, pages 34–40, 2017.

[7] A. R. Arasteh, M. Debbabi, A. Sakha, and M. Saleh. Analyzing Multiple Logs for Forensics Evidence. In *Digital Investigations*, volume 4, pages 82–91, 2007.

[8] F. P. Buchholz and C. Falk. Design and Implementation of Zeitline: a Forensic Timeline Editor. In *Proc. of the Digital Forensics Research Workshop*, 2005.

[9] A. Cailliau and A. van Lamsweerde. A probabilistic framework for goal-oriented risk analysis. In *Proc. of the 20th International Requirements Engineering Conference*, pages 201–210, 2012.

[10] B. Carrier. The Sleuth Kit (TSK) & Autopsy: Open Source Digital Forensic Tools. http://www.sleuthkit.org, 2014.

[11] B. D. Carrier. *A Hypothesis-Based Approach to Digital Forensic Investigations.* PhD thesis, Purdue University, 2006.

[12] E. Casey, A. Blitz, and C. Steuart. Digital Evidence and Computer Crime, 2014.

[13] E. Clarke, F. Lerda, and M. Talupur. An Abstraction Technique for Real-time Verification. *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*, pages 1–17, 2007.

[14] e-fense. Helix. http://www.e-fense.com/products.php, 2014.

[15] M. Elyas, A. Ahmad, S. B. Maynard, and A. Lonie. Digital Forensic Readiness: Expert Perspectives on a Theoretical Framework. *Computers & Security*, 52:70–89, 2015.

[16] B. E. Endicott-Popovsky and D. A. Frincke. Embedding Forensic Capabilities into Networks: Addressing Inefficiencies in Digital Forensics Investigations. In *Proceedings of the 2006 IEEE Workshop on Information Assurance*, pages 133–139, 2006.

[17] M. S. Feather, S. Fickas, A. Van Lamsweerde, and C. Ponsard. Reconciling System Requirements and Runtime Behavior. In *Proc. of the 9th International Workshop on Software Specification and Design*, page 50, 1998.

[18] S. Fickas and M. S. Feather. Requirements Monitoring in Dynamic Environments. In *Proc. of the 2nd International Symposium on Requirements Engineering*, pages 140–147, 1995.

[19] Q. Fu, J. Zhu, W. Hu, J.-G. Lou, R. Ding, Q. Lin, D. Zhang, and T. Xie. Where do developers log? an empirical study on logging practices in industry. In *Companion proc. of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 24–33, 2014.

[20] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, and M. T. Schaub, T.and Schneider. Potassco: The potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.

[21] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proc. of 5th Intl. Conf. on Logic Programming*, pages 1070–1080, 1988.

[22] D. Giannakopoulou and J. Magee. Fluent model checking for event-based systems. In *proc. of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering 2003 held jointly with 9th European Software Engineering Conference*, pages 257–266, 2003.

[23] P. Gladyshev and A. Patel. Finite State Machine Approach to Digital Event Reconstruction. *Digital Investigations*, 1:130–149, 2004.

[24] G. Grispos, J. García-Galán, L. Pasquale, and B. Nuseibeh. Are you ready? Towards the engineering of forensic-ready systems. In *Proc. of the 11th International Conference on Research Challenges in Information Science*, pages 328–333, 2017.

[25] M. R. Grossman and G. V. Cormack. Technology-Assisted Review in E-Discovery Can Be More Effective and More Efficient Than Exhaustive Manual Review. *Rich. JL & Tech.*, 17:1, 2010.

[26] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008.

[27] C. Hargreaves and J. Patterson. An Automated Timeline Reconstruction Approach for Digital Forensic Investigations. *Digital Investigation*, 9:S69–S79, 2012.

[28] T. A. Henzinger and O. Kupferman. From quantity to quality. In *Proc. of the International Workshop on Hybrid and Real-Time Systems*, pages 48–62, 1997.

[29] R. J. Heuer. *Psychology of Intelligence Analysis.* Center for the Study of Intelligence, 1999.

[30] J. King, R. Pandita, and L. Williams. Enabling forensics by proposing heuristics to identify mandatory log events. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, HotSoS '15, pages 6:1–6:11, 2015.

[31] R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.

[32] M. Law, A. Russo, and K. Broda. Iterative learning of answer set programs from context dependent examples. In *proc. of the 32nd International Conference on Logic Programming*, 2016.

[33] L. Liu, E. S. K. Yu, and J. Mylopoulos. Security and Privacy Requirements Analysis within a Social Setting. In *Proc. of the 11th IEEE International Conference on Requirements Engineering*, pages 151–161. IEEE Computer Society, 2003.

[34] J. Ma, F. Le, D. Wood, A. Russo, and J. Lobo. A declarative approach to distributed computing: Specification, execution and analysis. *TPLP*, 13(4-5):815–830, 2013.

[35] New York Times. Raj Rajaratnam's Galleon Group Founder Convicted in Insider Trading Case. topics.nytimes.com/top/reference/timestopics/people/r/raj_rajaratnam/index.html.

[36] G. Palmer. A Road Map for Digital Forensics Research. Technical report, Air Force Research Lab, Rome, 2001.

[37] L. Pasquale, S. Hanvey, M. Mcgloin, and B. Nuseibeh. Adaptive Evidence Collection in the Cloud Using Attack Scenarios. *Computers & Security (In Press)*, 2016.

[38] L. Pasquale, Y. Yu, M. Salehie, L. Cavallaro, T. T. Tun, and B. Nuseibeh. Requirements-Driven Adaptive Digital Forensics. In *Proc. of the 21st International Requirements Engineering Conference*, pages 340–341, 2013.

[39] A. Pnueli. The temporal logic of programs. In *Proceedings of SFCS77*, pages 46–57, 1977.

[40] D. Quick and K.-K. R. Choo. Big Forensic Data Reduction: Digital Forensic Images and Electronic Evidence. *Cluster Computing*, 19(2):723–740, 2016.

[41] O. Ray. Nonmonotonic abductive inductive learning. *J. Applied Logic*, 7(3):329–340, 2009.

[42] K. Reddy and H. S. Venter. The Architecture of a Digital Forensic Readiness Management System. *Computers & Security*, 32:73–89, 2013.

[43] S. Rekhis and N. Boudriga. Formal digital investigation of anti-forensic attacks. In *Proc. of the 5th IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 33–44, 2010.

[44] W. N. Robinson. A Requirements Monitoring Framework for Enterprise Systems. *Requir. Eng.*, 11(1):17–41, 2006.

[45] R. Rowlingson. A Ten Step Process for Forensic Readiness. *International Journal of Digital Evidence*, 2(3):1–28, 2004.

[46] M. Salifu, Y. Yu, A. K. Bandara, and B. Nuseibeh. Analysing Monitoring and Switching Problems for Adaptive Systems. *Journal of Systems and Software*, 85(12):2829–2839, 2012.

[47] B. Schatz, G. Mohay, and A. Clark. Rich Event Representation for Computer Forensics. In *Proc. of the 5th Asia-Pacific Industrial Engineering and Management Systems Conference*, pages 1–16, 2004.

[48] C. Shields, O. Frieder, and M. Maloof. A System for the Proactive, Continuous, and Efficient Collection of Digital Forensic Evidence. In *Digital Investigations*, volume 8, pages 3–13, 2011.

[49] T. Stallard and K. N. Levitt. Automated Analysis for Digital Forensics Science: Semantic Integrity Checking. In *Proc. of the Annual Computer Security Application Conference*, pages 160–167, 2003.

[50] P. Stephenson. Modeling of Post-Incident Root Cause Analysis. *International Journal of Digital Evidence*, 2, 2003.

[51] C. Taylor, B. Endicott-Popovsky, and D. A. Frincke. Specifying Digital Forensics: A Forensics Policy Approach. *Digital investigation*, 4:101–104, 2007.

[52] S. Tewelde, S. Gruner, and M. Olivier. *Advances in Digital Forensics XI: 11th IFIP WG 11.9 International Conference, Orlando, FL, USA, January 26-28, 2015, Revised Selected Papers*, chapter NOTIONS OF HYPOTHESIS IN DIGITAL FORENSICS, pages 29–43. Springer International Publishing, 2015.

[53] T. T. Tun, A. K. Bandara, B. A. Price, Y. Yu, C. B. Haley, I. Omoronyia, and B. Nuseibeh. Privacy Arguments: Analysing Selective Disclosure Requirements for Mobile Applications. In *Proc. of the 20th International Requirements Engineering Conference*, pages 131–140, 2012.

[54] J. Van den Bos and T. Van Der Storm. Domain-specific optimization in digital forensics. In *International Conference on Theory and Practice of Model Transformations*, pages 121–136, 2012.

[55] A. van Lamsweerde. Elaborating Security Requirements by Construction of Intentional Anti-Models. In *Proc. of the 26th International Conference on Software Engineering*, pages 148–157. IEEE Computer Society, 2004.

[56] S. Y. Willassen. Using Simplified Event Calculus in Digital Investigation. In *Proc. of the ACM Symposium on Applied Computing*, pages 1438–1442, 2008.

[57] J. Zhu, P. He, Q. Fu, H. Zhang, M. R. Lyu, and D. Zhang. Learning to Log: Helping Developers Make Informed Logging Decisions. In *Proc. of the 37th International Conference on Software Engineering*, pages 415–425, 2015.