Defining Atomicity (and Integrity) for Snapshots of Storage in Forensic Computing

Jenny Ottmann[a,*], Frank Breitinger[b] and Felix Freiling[a,*]

[a]Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
[b]School of Criminal Justice, University of Lausanne, 1015 Lausanne, Switzerland

ARTICLE INFO

Keywords:
storage acquisition
instantaneous snapshot
correctness
integrity

ABSTRACT

The acquisition of data from main memory or from hard disk storage is usually one of the first steps in a forensic investigation. We revisit the discussion on quality criteria for "forensically sound" acquisition of such storage and propose a new way to capture the intent to acquire an *instantaneous* snapshot from a single target system. The idea of our definition is to allow a certain flexibility into when individual portions of memory are acquired, but at the same time require being consistent with causality (i.e., cause/effect relations). Our concept is much stronger than the original notion of atomicity defined by Vömel and Freiling (2012) but still attainable using copy-on-write mechanisms. As a minor result, we also fix a conceptual problem within the original definition of integrity.

1. Introduction

Data from storage devices or main memory are crucial pieces of evidence today. The acquisition of such data usually means to copy the data from storage to another storage (controlled by the analyst) in a way that preserves as much of its evidential value as possible. A common way to define a "good" copy is to formulate a set of quality metrics that capture the intention of forensic soundness.

Attaining good quality copies appears seemingly simple if storage can be "frozen". As an example, there has been little debate about the classical way to produce a forensic copy of a hard disk using `dd` as described by Carrier (2005). This is in contrast to the acquisition of main memory which — apart from approaches that literally "freeze" RAM (Halderman, Schoen, Heninger, Clarkson, Paul, Calandrino, Feldman, Appelbaum and Felten, 2009) —- has received considerable attention if the acquisition target is not a virtual machine (Vömel and Freiling, 2012; Pagani, Fedorov and Balzarotti, 2019; Inoue, Adelstein and Joyce, 2011; Campbell, 2013; Lempereur, Merabti and Shi, 2012; Gruhn and Freiling, 2016). In the form of solid state drives, hard disks have turned into increasingly active devices which has made forensic data acquisition in the classical sense impossible (Nisbet, Lawrence and Ruff, 2013). In practice, circumstances may prohibit freezing altogether even though it may be technically feasible. Examples are production servers that cannot be paused for operational reasons. On such systems, acquisition is often improvised and part of a live analysis.

*Corresponding authors.
Email addresses: `jenny.ottmann@fau.de` (J. Ottmann); `frank.breitinger@unil.ch` (F. Breitinger); `felix.freiling@fau.de` (F. Freiling)
URL: `https://FBreitinger.de` (F. Breitinger)
ORCID(s): `0000-0003-1090-0566` (J. Ottmann); `0000-0001-5261-4600` (F. Breitinger); `0000-0002-8279-8401` (F. Freiling)

1.1. Inconsistencies in RAM acquisition

As mentioned, quality criteria for data acquisition have most often been discussed in the context of volatile memory because of the common problems that occur if RAM is acquired inconsistently.

Page smearing, for example, is a common problem on systems under heavy load or with more than 8 GB of RAM according to Case and Richard III (2017). When page smearing occurs, the page tables in the memory snapshot are not consistent with the contents of the physical pages because changes were made to the referenced physical pages after the page tables were acquired. This can result, for example, in pages being attributed to wrong processes or inconsistencies in kernel data structures. Some of these inconsistencies can lead to problems during the memory analysis or hinder an analysis completely if it uses kernel data structures. But not all inconsistencies have to be apparent during an analysis. Therefore, precise measurement criteria, to define the quality of a memory imaging method, can help to evaluate tools without having to rely on visible problems.

1.2. The quest for suitable quality criteria

To be able to qualify the effects of RAM acquisition, Vömel and Freiling (2012) introduced three criteria to evaluate the quality of a memory snapshot: *correctness*, *integrity*, and *atomicity*. A correct snapshot contains all memory regions of the main memory and for each region exactly the value it had in main memory at the time of the acquisition. Thus, to achieve correctness not only a correct implementation is necessary but the used system components must return the correct values as well. The integrity criterion focuses on memory content changes between the start of the acquisition process and the acquisition of each memory region. Integrity is violated for memory content that changed after the acquisition was started and before it could be copied by the acquisition process. The atomicity criterion in contrast allows changes of memory contents if the acquired memory regions are causally consistent. This means that no

memory region content in the snapshot was influenced by changes to a memory region that are not part of the snapshot.

Recently, Pagani et al. (2019) criticized the atomicity definition by Vömel and Freiling (2012) for being "extremely difficult to measure in practice". Instead, they suggested a criterion called *time consistency*. A snapshot is time consistent if there "exists a hypothetical atomic acquisition process that could have returned the same result". However, they do not provide a precise formalization of this concept.

1.3. Related work

There exists a large body of work that investigates the creation of snapshots in distributed concurrent systems often with the aim to detect predicates on the state of a distributed computation (Chase and Garg, 1998). In this work, focus has been on *asynchronous* distributed systems where the best available notion of time is causality (Mattern, 1989; Schwarz and Mattern, 1994). In such systems, concurrent execution of events makes the global state "relativistic", i.e., it is often not possible to exactly say in which state the system is or has been (Cooper and Marzullo, 1991; Gärtner and Kloppenburg, 2000; Chu and Brockmeyer, 2008).

In forensic computing, data acquisition is (currently) usually performed in a synchronous environment. While concurrency arises even in such systems from different threads that operate on shared memory, such systems provide a common centralized clock that can be used to order events. If events can potentially be totally ordered, the sequence of global states through which the system progresses is well defined. In contrast to the assumptions made in previous theoretical work that describes algorithms for predicate detection in synchronizable systems (Stoller, 2000), real systems usually do not keep track of timestamps of individual events. The application of complex snapshot algorithms in forensics appears not advisable anyway since taking forensic snapshots should minimize interference with the observed system. So, while the literature on distributed systems gives many insights into the problem area, we are not aware of work that is of direct help.

Other related work is concerned with measurement of the quality of snapshots. Early work avoided the need to define quality criteria by simply comparing the output of a tool with the memory content of the machine from which the snapshot was taken (Inoue et al., 2011; Campbell, 2013; Lempereur et al., 2012). Vömel and Stüttgen (2013) were the first to perform a practical evaluation of memory acquisition tools against the abstract quality criteria of Vömel and Freiling (2012). They implemented the evaluation platform using *Bochs* and took a white-box testing approach. With the help of inserted hypercalls, three tools were evaluated. Correctness could be measured exactly by comparing the memory image created by the acquisition process to an image created in parallel by the host. Atomicity could not be measured exactly as this would have required to keep track of all causal dependencies in the guest, a task deemed nearly infeasible by the authors. Instead, possible atomicity violations were measured by keeping track of which threads accessed already acquired pages and then modified a page that was not already acquired. Therefore, the results present an upper bound of atomicity violations. Integrity was estimated by comparing a memory image taken by the host shortly before the acquisition process was loaded into the guest memory with one taken by the host shortly after the acquisition process finished.

Building on the results by Vömel and Stüttgen (2013), Gruhn and Freiling (2016) took correctness for granted and followed a black-box approach to measure atomicity and integrity. Because their method does not rely on modifying the source code of tools, more tools could be evaluated, including direct memory access (DMA) and cold boot. For the tests they wrote a program which allocates sequentially numbered memory regions and one to extract the numbered regions from a memory snapshot. The numbers serve as a counter that allows to *estimate* the level of atomicity and integrity.

1.4. Contributions

In this paper, we revisit Vömel and Freiling (2012) and follow the demand formulated by Pagani et al. (2019) for more "permissive" quality metrics for the acquisition of storage: We formalize two new definitions of atomicity which we call *instantaneous* and *quasi-instantaneous consistency*. Both can be seen as possible formalizations of the notion of "time consistent" by Pagani et al. (2019).

Instantaneous consistency resembles the quality of an "ideal" snapshot taken from a frozen system and implies quasi-instantaneous consistency. But although being slightly weaker in guarantees, a quasi-instantaneous snapshot is indistinguishable from an instantaneous snapshot. We show that quasi-instantaneous snapshots can be achieved (by performing memory snapshots using the idea of copy-on-write). Moreover, under certain assumptions quasi-instantaneous consistency implies the (classic) causal consistency of Vömel and Freiling (2012) so quasi-instantaneous snapshots do not violate cause-effect relations. Since the common memory snapshot techniques based on software generally do not even guarantee causal consistency, we also raise the question of how to assess a memory snapshot regarding its level of atomicity.

As a minor contribution, we propose a new definition of integrity that is refined from Vömel and Freiling (2012) and removes some of its theoretical weaknesses. We formulate all concepts independent from concrete storage technologies so that they can be applied to any block-based digital storage, be it volatile or persistent.

1.5. Outline

The following section introduces the model used to formalize our concepts. Section 3 continues with a formal definition of the two new forms of atomicity. Section 4 provides an overview of methods with which these consistency criteria can be achieved, while Section 5 discusses some ideas on how to evaluate snapshots of storage with respect to the new metrics. Section 6 briefly presents our result on the notion of integrity. Old and new concepts are compared

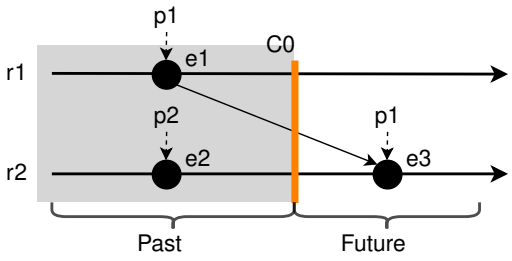

Figure 1: Space/time diagram of a computation and one possible cut C0. The events to the left of the cut are part of the past, those to the right part of the future.

in Section 7. Section 8 discusses legal implications of our concepts for concrete investigations. Finally, Section 9 concludes the paper.

2. Model

We now define the notation to describe computations on memory regions and the timing relations of the events that happen therein (the definitions are adapted from Zheng and Garg (2019) using the timing notation of Chu and Brockmeyer (2008)).

2.1. Processes, memory regions and events

We consider a finite set $P = \{p_1, \ldots\}$ of processes (or threads) that perform operations on a set of n memory regions $R = \{r_1, \ldots, r_n\}$. Performing an operation results in an event $e = (p, r)$, where $e.p$ denotes the process that performed e and $e.r$ denotes the memory region on which the process performed the operation. The set of all events is denoted E. We assume that operations on a single memory region are performed sequentially (e.g., by using hardware arbitration or locks).

2.2. Space/time diagrams and cuts

We use space/time diagrams, commonly used in distributed computing (Mattern, 1989), to depict computations. The sequence of events within the memory regions serves as timeline from left to right, and the sequential activities of processes are depicted as arrows that connect events. An example is shown in Fig. 1 with two memory regions r_1 and r_2, a process p_1 executing events e_1 and e_3 and a process p_2 executing event e_2.

A *cut* through the space/time diagram is indicated by a line that intersects each memory region exactly once. Formally, a cut is a subset of events of the computation and can be regarded as separating the events into a "past" (to the left of the cut) and a "future" (to the right of the cut). Fig. 1 shows an example of a cut through a computation.

2.3. Causal order on events

A computation is modeled as a tuple $(E, \rightarrow)$ where E is the set of events and $\rightarrow$ is the *causal order* on E, i.e., the smallest transitive relation such that

1. if $e.p = f.p$ and e immediately precedes f in the sequential order of that process, then $e \rightarrow f$, and
2. if $e.r = f.r$ and e immediately precedes f in that memory region, then $e \rightarrow f$.

The order $\rightarrow$ corresponds to Lamport's happened-before relation (Lamport, 1978).

The order $\rightarrow$ merely encodes which events might have influenced which other events, i.e., if $e \rightarrow f$ or $f \rightarrow e$ then either e may have influenced f or vice versa. However, if neither $e \rightarrow f$ nor $f \rightarrow e$ we say that e and f are *concurrent*, i.e., it is not possible to order the two events regarding causality.

2.4. Observability of causal relations

The causal order relation between events is by definition independent of the concrete values that processes read from or write to memory regions. For example, in the computation depicted in Fig. 1 all events e_1, e_2 and e_3 could be read events that do not modify the content of the memory regions. Causal dependencies, therefore, may not be observable unless they are somehow reflected by the stored values. A minimum requirement for events to be observable is that they perform a state change of the memory region, e.g., to change the stored value from 0 to 1. Events that always update the state of a memory region to a different state as before are called *modifying events*.

Merely requiring that events modify the state of a memory region does not imply that state changes can always be observed. The reason for this is that subsequent state changes can annihilate effects of previous state changes. For example, event e_2 in Fig. 1 could change the value of memory region r_2 from 0 to 1, and event e_3 could change it back from 1 to 0. The fact that an event has occurred is not observable if the starting and ending state of r_2 is inspected. This can be avoided by demanding that every event assigns a "fresh" value to the memory region. This is the case, for example, if the stored value is a counter that is incremented with every event. Events that change the value of the memory region to a new unique value are called *uniquely modifying events*.

Techniques to observe causal relationships in distributed systems (like vector clocks (Mattern, 1989)) are commonly based on the assumption of uniquely modifying events. We will revisit these concepts later when exploring the compatibility between our new consistency notions and causal order.

2.5. Consistent global states

Cuts are often considered as representations of global states of the computation. Fig. 2 depicts multiple possible cuts through the computation shown in Fig. 1. For example, cut c_0 is the initial cut (no event has happened yet), c_1 is the cut where e_1 is the only event that has happened, and c_4 is the cut where e_2 and e_3 have happened but not e_1. The causal order $\rightarrow$ on events induces a partial order on these states that form a lattice. The lattice of all such global states of the computation shown in Fig. 2 is depicted in Fig. 3.

Note, in the absence of any notion of real-time, it cannot be determined whether event e_1 happened before e_2 or not