Computer Profiling for Preliminary Forensic Examination

Andrew Marrington[1](✉), Farkhund Iqbal[1], and Ibrahim Baggili[2]

[1] Advanced Cyber Forensics Research Laboratory, College of Technological Innovation, Zayed University, P.O. Box 19282, Dubai, United Arab Emirates
marrington@computer.org, farkhund.iqbal@zu.ac.ae

[2] Tagliatela College of Engineering, University of New Haven, 300 Boston Post Road, West Haven, CT 06516, USA
ibaggili@newhaven.edu

Abstract. The quantity problem and the natural desire of law enforcement to confront suspects with evidence of their guilt close to the time of arrest in order to elicit a confession combine to form a need for both effective digital forensic triage and preliminary forensic examination. This paper discusses computer profiling, a method for automated formal reasoning about a computer system, and its applicability to the problem domain of preliminary digital forensic examination following triage. It proposes an algorithm for using computer profiling at the preliminary examination stage of an investigation, which focusses on constructing an information model describing a suspect's computer system in the minimal level of detail necessary to address a formal hypothesis about the system proposed by an investigator. The paper concludes by discussing the expanded utility of the algorithm proposed when contrasted to existing approaches in the digital forensic triage and preliminary examination space.

Keywords: Computer profiling · Triage · Formal methods · Preliminary examination

1 Introduction

The *quantity problem* in digital forensics was described by Carrier as the large amount of data which needs to be analyzed in the course of a digital investigation [1]. The quantity problem is necessarily addressed in digital investigations through data reduction techniques. A well-known example of such a technique in the investigation of a computer hard disk is the elimination of files known to be of no interest to the investigation from the examiner's view of the file system through the use of known file filters. The quantity problem contributes significantly to the time needed to complete any digital investigation, and consequently is a cause of significant delay to law enforcement. This problem becomes acute when an investigation incorporates numerous computers, storage media and other digital devices, many of which may be of no evidentiary value.

P. Gladyshev et al. (Eds.): ICDF2C 2013, LNICST 132, pp. 207–220, 2014.
DOI: 10.1007/978-3-319-14289-0_14

Unfortunately, a time consuming complete forensic examination may be required simply to find that the device in question contains no relevant digital evidence. It is more efficient for the examiner's time and effort to be focused on the devices and media which are considered to have the greatest potential evidentiary value. This is analogous to the paramedic, whose ministrations must first be focused on the injured person with the best chance of survival given timely treatment. In the medical domain, this allocation of medical resources is called *triage*. Even once the examiner's efforts have been focussed on the devices and media which have the greatest evidentiary value, there is a further need to quickly examine those devices and media in enough detail necessary to inform the rest of the investigation and/or to provoke an earlier confession from a suspect, both of which may be significantly delayed by waiting for a length in-depth examination in the lab.

Casey et al. argue that simply defaulting to in-depth forensic examination alone risks unacceptably delaying investigations involving digital evidence, and consequently argue for three stages to digital forensic investigations [2]:

1. Survey/Triage Forensic Inspection.
2. Preliminary Forensic Examination.
3. In-depth Forensic Examination.

As defined in [3], in medical terms triage is "a process for sorting injured people into groups based on their need for or likely benefit from immediate medical treatment." In the field of digital forensics, triage means identifying the digital devices or media which contain evidence relevant to the investigation at hand [2]. After this stage is completed, the devices and media identified can be examined. An in-depth forensic examination will produce the most complete results, but will take a long time, and may be subject to further delays due to backlogs at the digital forensics laboratory (which could stretch into the months or even years). Such a delay in the forensics lab can delay the entire investigation, and the impact the digital evidence may have to the course of that investigation may be reduced if it is not available in the early stages of the investigation. For example, a suspect may be less likely to confess if they have had a number of months to anticipate the discovery of certain files on their computer hard disk and have had time in the meantime to seek legal counsel. Consequently, Casey et al. argue for a stage between triage and in-depth examination, called *preliminary forensic examination*, "with the goal of quickly providing investigators with information that will aide them in conducting interviews and developing leads" [2].

In this work we propose the application of automated and formal *computer profiling* to the triage and preliminary examination stages described by Casey et al. [2]. The term *profiling* is used to describe many different approaches conceived for many different purposes in the digital investigations literature, such as criminal profiling [4], or user log transaction profiling [5]. In this work, profiling means that the computer system will be automatically described according to a formal model to support formal reasoning for the purpose of allowing an investigator to quickly answer questions about the computer system as a preliminary examination activity. The result of this process, the computer profile, can be tested against an investigator's formally stated hypothesis. The computer profile can also be used by the investigator to test other evidentiary statements, such as witness testimony, against the digital evidence recovered from the computer system. The results of this evaluation of hypotheses and evidentiary statements against the computer profile can both inform the decision to conduct and guide a subsequent thorough forensic examination of the system. Unlike the triage stage, some prior evidence (e.g. witness statement) or suspicion is assumed in this preliminary examination stage process. We believe that profiling may have particular utility in the evaluation of theories of the crime, and in interviewing witnesses and suspects, as both can be expressed formally and evaluated against the profile.

The approach we take in this work is based on a model for the forensic description of computer systems (i.e. profiling) described by Marrington et al. [6] and extended and improved upon by Batten and Pan [7]. This provides the basis for the description of a computer system to support formal reasoning. The formalism is useful as it provides for practical implementation which can support preliminary forensic examination in a generic sense, rather than in the form of a case-specific implementation. The contribution of this work is to describe algorithms for the implementation of a computer profiling tool for preliminary forensic examination, to support efficient querying of a computer profile to answer key questions about a computer system prior to a manual forensic examination of that computer system. In Sect. 2, we discuss the literature about both digital forensic triage and preliminary examination (Subsect. 2.1) and formal models for digital forensics (Subsect. 2.2).

2 Related Work

2.1 Triage and Preliminary Examination in Digital Forensics

In any criminal investigation, a suspect is more likely to confess to a crime when caught in the act or confronted with some evidence of their guilt. Much of the existing digital forensics literature describes all forensic activities designed to produce results more quickly (especially for the purposes of provoking an earlier confession) has described those activities as forensic triage activities. We believe that some of these techniques are either equally or more properly activities which could belong in Casey et al.'s preliminary forensic examination stage [2], or which at the very least inform the investigator's activity in the preliminary stage. Consequently, all of these activities, whether they are strictly triage or preliminary examination stage activities, or a combination of the two, are related to the work described in this paper.

Much of the existing literature in the digital forensic triage domain has considered the scenario of a fast, on-scene automated or semi-automated examination of the system to allow law enforcement officers to confront suspects with evidence of guilt in situ – a sort of "digital forensics breathalyzer". Such triage approaches are case specific – for example, in a child pornography case, a graphical triage utility might display thumbnails of image files on the screen [8].

Confronted with the illicit material they are alleged to have downloaded, a suspect may confess at the scene to law enforcement officers. Subsequently, the suspect's computer system will still be examined manually in the usual fashion, and the evidence produced by this "traditional" digital forensic examination will be the evidence produced in court.

The breathalyzer analogy is useful to illustrate the relationship between the on-scene triage examination and the more thorough examination in the laboratory. In many jurisdictions, once a suspect blows over the legal blood-alcohol limit on the breathalyzer, a blood test is taken to verify the results of the breathalyzer. It is this subsequent blood test which is considered to produce more reliable scientific evidence, but the initial breathalyzer results are often enough to provoke a suspect to confess to driving under the influence.

A forensic examiner needs to first identify various primary and secondary sources of evidence including computer internal memory and external storage media [8]. The internal memory of a computer is volatile in nature and its' contents may be lost if not handled immediately after the confiscation of a suspect's computer. Volatile memory may contain traces of forensically crucial information including open DLL files, function calls, references to called functions, system's files and processes, and operating system's utilities and artefacts. Therefore, once a crime-scene is cordoned off, it is imperative to start data acquisition first from fragile volatile containers and then from static storage devices.

Unlike traditional file-based data acquisition practice where most of the internal salient features of potential evidence are obscured, bulk-based evidence collection is best suited to triage. Garfinkel has developed *bulk_extractor*, a tool that facilitates evidence extraction and feature identification from confiscated computers with applications in both triage and preliminary examination [9].

2.2 Formal Models in Digital Forensics

Gladyshev and Patel proposed the use of a finite state machine (FSM) model for the purposes of reconstructing the history of a computer system [10], an approach also employed by Carrier and Spafford [11]. A finite state machine is a general computational model composed of a finite set of states, the transitions between those states, and various actions. A finite state machine is usually described as follows:

$$M = (Q, \Sigma, \delta, q_0, F)$$

where Q is the finite set of states of the machine, Σ is the input alphabet (or the finite set of possible events), $q_0 \in Q$ is the initial state of the machine, $F \subseteq Q$ is the set of final states, and δ is the transition function mapping $Q \times \Sigma$ to Q. At any given point in its history, a computer system can be in only one state. When an event (such as user input) happens, the computer system may change state or may stay in the same state – as expressed by the transition function $\delta(q, \sigma)$, which gives a state for each state $q \in Q$ and each event $\sigma \in \Sigma$. The state of the system at the time of the investigation can be considered to be the final state $f \in F$, and event reconstruction is fundamentally about tracing back the history of the machine from f to q_0 [10].

Gladyshev and Patel employ transition back-tracing to enumerate all of the possible scenarios leading to the final state q. A scenario is a run of finite computations, producing a series of transitions (that is, a series of events and states at the time of those events) leading the system M to end in state q. Essentially, a scenario describes some hypothesis explaining the evidential statement. Those scenarios arriving at the final state q which are inconsistent with the rest of the evidential statement (for example, witness statements, print-outs, trusted logs) can be discarded. This leaves only those scenarios which are consistent with the evidential statement. The evidential statement can be expanded to include investigative hypotheses, such that scenarios which disprove those hypotheses would be discarded by the expansion of the evidential statement to include them. Evidential statements are built from non-empty chronological sequences of observations. An observation is a statement to the effect that some property p was observed over time. Each observation is expressed as a triple $o = (P, min, opt)$, where P is the set of all computations of M possessing the observed property p, and min and opt are positive integers specifying the duration of observation. Each computation performed by the computer system includes an event $\sigma \in \Sigma$ and a state $q \in Q$, and causes the system to transition into another state (or back into the same state). A run of computations is a sequence of computations such that each computation causes the system to transition into the state which is part of the next computation in the run. A run of computations r is said to explain the observation o if every element of the run r possesses the observed property p (that is, for all i where $0 \le i < |r|, r_i \in P$), and if $min \le |r| \le (min + opt)$ [10]. In this way, hypotheses are tested for computational feasibility.

It is difficult, however, to describe a complete practical computer system according to the finite state machine based-models. The problem of mapping the complete state space of a practical computer system is extremely computationally demanding. The models and techniques they describe could still be used in this fashion where the digital system under investigation (or a simplified model or subsection of it) is simple enough, given realistic computational limits, to be modelled as a finite state machine, but then this requires some degree of abstraction, which risks abstraction error [1].

Marrington et al. describe another model to describe computer systems for forensic purposes, one based on information available to investigators rather than a computational model which may be difficult to construct in practice [6]. In this model, the computer system is described as a 4-tuple cp:

$$cp = (O, AR, T, \textsc{Evt})$$

where O is the set of all the objects on the computer system (encapsulating users, groups, files, applications, and so on), AR is the set of relations on O capturing the relationships between different objects on the computer system, T is the set of all the times in the computer system's history (which need not be explicitly populated but which is included for the sake of completeness), and $\textsc{Evt}$ is the set of all events in the computer system's history. Batten and Pan refine this model, allowing for the dynamic resizing of the set of all objects on the computer system O as the investigation progresses [7]. All components of this computer profiling model are constructed from the information which can be retrieved from the computer system at the time of its seizure. Although it lacks the power of computational models, this information-based model is practical, and the concept of relationships in particular provides for quite useful querying by an investigator, as shown in [7].

In the profiling model there are different types of relationships, built from binary predicates, which capture the nature of the connection between two objects [6]. A generic relation R on O is a set of ordered pairs of $O \times O$, and if the predicate $related(a, b)$ is true, where $a \in O, b \in O$), then we say aRb. Relations describing more specific predicates, such as $author(a, b)$ (i.e. that a is the author of some document b), are represented with more specific relationship types in [6], such as $a\mathrm{R}\textsc{Author}b$. Relations also have different properties [6,7]:

1. A relation R on O is *reflexive* if $\forall o \in O, oRo$.
2. A relation R on O is *symmetric* if aRb implies bRa for all objects $a \in O$ and $b \in O$.
3. A relation R on O is *transitive* if aRb and bRc implies aRc for all objects a, b and c in O.

Batten and Pan also prove that if a relation is reflexive, symmetric and transitive, then it is an *equivalence* relation by transitive closure (i.e. that if aRb is reflexive, symmetric and transitive, then $(a) = (b)$) [7]. As should be obvious from the discussion of objects and relationships in logical terms, computer profiling happens at a high level of abstraction, equivalent to a high-level of abstraction in Carrier's Complex Computer History Model [11] as opposed to the low-level Primitive Computer History Model. This means that profiling is prone to abstraction error as discussed in [1]. However, as we are applying profiling to the preliminary examination stage, we expect that the issue of abstraction error will be mitigated by subsequent forensic examination of the file system.

Our work builds on the work of Marrington et al. in [6] and of Batten and Pan in [7], employing the computer profiling model to the problem of preliminary examination. In so doing, we hope to practically provide some of the features of the FSM-based models of Gladyshev and Patel [10] and Carrier and Spafford [11], specifically with regard to testing hypotheses and evidentiary statements against the model. The principle difference is that we test these statements against a practically populated information model rather than a powerful, complete but unobtainable (or extremely abstracted) computational model.

3 Profiling for Preliminary Examination

Hard disk drive capacities are growing far in excess of the rate of improvement in bus speeds and seek times. This has informed our design for profiling-based preliminary forensic examination, because it means that it is desirable to extract the least information necessary from the suspect computer system to sustain or refute an evidentiary statement. Batten and Pan demonstrate the utility of dynamically resizing the set of all objects on the computer system O throughout the course of an investigation in order to more efficiently answer questions about the computer system's profile [7]. We apply this theoretic approach to the problem domain of preliminary examination, as we believe it supports the selective extraction of evidence from the hard disk (as opposed to requiring the examiner to acquire a complete disk image).

We suggest an iterative profiling/querying process for preliminary forensic examinations. At each iteration we build a computer profile, query it, and if the query does not return a result, we expand the profile and repeat. It is envisioned that this process would be repeated for a maximum number of iterations, which we shall simply refer to as n. At each iteration, we build a computer profile cp_i:

$$cp_i = (O_i, AR_i, T, \textsc{Evt}_i).$$

Note that we anticipate that the times in T will not change between iterations of the profiling triage process, although the other sets may expand each iteration. T will not change between iterations because it refers to all the times in the history of the computer system, which does not vary, nor do we anticipate that in practice T would have to be explicitly populated during the profiling process. In Subsect. 3.2 we start to describe the construction of this model, and expand on it with each subsequent iteration as described in Subsect. 3.3.

3.1 Setup

In order to minimize how much data must be read from the hard disk, each profiling iteration only reads from a portion of the disk. The complete disk will only be read if the profiling process reaches iteration n (the final iteration). The objective for triage is to build a profile which is complete enough at an early enough iteration to address the investigator's queries about the computer system without requiring n iterations.

In order to support iterative profiling, our algorithm requires that the computer's hard disk/s be divided into n "slices". Other than defining what should be included in the first slice ($slice_0$), our approach does not specify how the hard disk should be divided into these slices. If profiling is taking place on a live system, then it makes sense to divide the hard disk into slices according to the logical structure of the file system. If profiling is taking place on a previously acquired hard disk image, then it might make sense to divide the disk according to disk geometry (e.g. by sector). In the context of a preliminary forensic examination, we believe that the best approach would be to apply profiling to the original media through a write-blocker device – in this way, we avoid completely acquiring the media, but we do not give up the ability to search unallocated space. The exact mechanism chosen is an implementation issue – we simply specify that:

1. $slice_0$ includes the user home directories and main system log storage location; and
2. $slice_1$ to $slice_{n-1}$ are roughly equal in size to each other (but not necessarily to $slice_0$).

3.2 Initial Iteration

We start by building a very small set O_0 at our initial iteration, recording some metadata about each object in a database (roughly corresponding to the properties of objects discussed in [6]). This metadata includes, for each object, any associated timestamps and their descriptions, and source locations. Metadata can be a rich source of digital evidence, and can answer "Who, What, Where, When, How?" questions about files [12]. In terms of the profiling model, metadata can help us to discover relationships and events. At this stage, the set O_0 consists of a handful of important objects which can be identified from $slice_0$ of the suspect's hard disk/image:

1. The computer system's users (derived from the Windows registry, `/etc/password` or other sources).
2. The userland applications installed on the system (derived from the Windows registry or walking common directories like `c:\Program Files\` or `/usr/bin`).
3. The files in each user's home directory.

We then build AR_0, the set of relations on O_0. This set initially consists of relationships which we can derive simply from the metadata from the file system of the objects discovered in O_0. We should also determine the type of each relationship thus discovered. Relationship type can be derived from the metadata – if metadata is arranged as a set of keyword/value pairs, then the keyword indicates of the relationship type, whereas the value indicates the other object in the relationship.

For example, if an application or file, (x), is owned by the user y, then xRy, so we add (x, y) to R. In this example, the type of the relationship would be ROWNER, as the file x belongs to the user y, as indicated by the keyword/value pair $(owner, y)$ in the metadata of the file x. This relation is *symmetric* as the reverse is also true – x belongs to y and y is the owner of x.

The set EVT_0 is constructed out of the events in the system logs (e.g. the Windows Event Logs), and out of events inferred from the timestamps stored in the metadata for each object in O_0. Each of the common file system metadata timestamps (modified, accessed, created) provides some indicator of an event relating to the file [12]. The set EVT may be expanded in future iterations.

This completes cp_0, which is now queried by the investigator according to the method described in Sect. 4. If the query returns a result, then the investigator's query statement has been sustained, and the investigator may chose to either switch to another query statement for testing, or finish with the preliminary examination activity. If no match was found, or the investigator chooses to continue to test another query statement, then we proceed to the next iteration.

3.3 Iteration i

At iteration i (where $0 < i < n$), we expand upon the profile produced in the previous iteration to add more information. Adapting the three-stage approach per iteration employed by Batten and Pan [7], we expand the set of all objects and the set of all relationships as they allow, but we also expand the set of all events. Unlike Batten and Pan's work, which permits resizing in both directions, our sets grow with each iteration – they are not reduced. All of the elements of cp_i have the initial value at the start of this iteration of the elements of cp_{i-1} – e.g. O_i starts as O_{i-1} and has new objects added to it during this iteration.

There are two distinct components to our expansion of the profile in each iteration i:

1. Searching $slice_i$ for new objects.
2. Searching the metadata of the objects in $(O_i - O_{i-1})$ for new relations and new events.

After completing both of these steps, cp_i is completed, and queried. If the query returns a result, then the profiling process finishes at this iteration unless otherwise directed to continue to test other statements. If no match was found, or the investigator chose to continue profiling, then we proceed to the next iteration.

Searching the Disk Slice. At this stage we enumerate or carve all of the files in the section of the disk $slice_i$, depending on whether we are conducting a live analysis of a hard disk or examining a disk image. A new object is created for each file discovered in $slice_i$, and the file metadata is recorded in a database for later reference. We add new objects to O_i.

Searching File Metadata. For all of the objects in the set given by $(O_i - O_{i-1})$ (i.e. all of the objects discovered in this iteration), we search the metadata of each object and create events and relationships. Assuming that the metadata is a set of keyword/value pairs, then:

– When the value takes the form of a timestamp, the keyword describes some event which took place at that time, which can be added to $\textsc{Evt}_i$.
– When the value takes the form of another file's name, an application, a username (or other person's name), etc., the keyword describes the type of a relation between the object and another object described by the value, which can be added to AR_i.

4 Querying the Computer Profile

After the computer profile has been constructed, it can be queried with statements formally expressing hypotheses or evidentiary statements. These statements are formed in a similar fashion to the approach described by Gladyshev with respect to his FSM-based model [10], but instead of being tested for computational feasibility, the statements are tested for consistency with the profile.

Since a computer profile consists of objects, relationships, times and events, it can be queried for any of these things. A query may be as simple as a statement for evaluation that a particular object, relationship, time or event exists within the profile:

- $o \in O$ if the file, user or application being searched for was found.
- aRb if the objects a and b are found and are related.
- $t \in T$ if the computer was in operation at time t.
- $evt \in \textsc{Evt}$ if the event being searched for was found.

Each of these can be extended and combined so that a query need not test whether a single object exists in the profile – instead, the query can combine multiple substatements along these lines. There are two sets which we believe will be of particular interest for querying: R and $\textsc{Evt}$.

Querying a profile to see whether a sequence of events occured within the system's history is relatively straightforward. We use the *happened-before* relation as described by Lamport [13]. A sequence of events with wildcard timestamps or bounded potential times instead of timestamps (e.g. after 6:27pm but before 8:30pm) is provided by the invesigator as a statement for evaluation. Then we test that each timestamp $t_i \in T$ in the sequence of events provided in the investigator's query *happened-before* the timestamp t_{i+1}, i.e., that:

$$t_i \rightarrow t_{i+1}.$$

Relationships provide the richest source for querying. An investigator might posit any relationship between any two objects. The statement would be sustained if such a relationship already existed in AR. Otherwise, a statement like aRb may still be sustained depending on the properties of the relationship type (see Sect. 3). For example, if the posited relationship type is *transitive* and the relation includes $((a, c), (c, d))$ then we can also say that aRb and the statement is therefore sustained.

5 Use in Preliminary Forensic Examination

Profiling exists as an activity in the preliminary examination stage to be deployed primarily when an investigator has suspicions or witness statements (or other statements of evidence), expressed as formal hypotheses, which may involve the computer system to be examined. These hypotheses may arise from the digital forensic triage stage, or from interviews with suspects, or from other leads. As formalized observations, the profiles produced at each iteration may be of some use to an investigator with no prior knowledge of a computer system, however, as the process continues through multiple iterations, we anticipate that the profile will grow too large to be manually digestible. This anticipation is based on the numbers of objects reported in profiles constructed according to an early version of the model described by Marrington et al., where for an average office desktop PC circa 2007, the profile constructed by prototype software included nearly 4600 objects and over 44,000 events [14]. The profiling implementation described in that paper was not iterative, so the profile constructed would be analogous to the profile cp_n in this work. If the computer profile is ever useful as a set of human-digestible observations about an unknown computer system, then, we suggest that it will only be so during the early iterations, as the numbers of objects, relationships and events are likely to be too large to be practical for manual interpretation as the profiling process nears iteration n. The real utility of profiling for preliminary forensic examination, then, is to test suspicions or evidentiary statements to support investigators as they form leads and as they interview suspects.

Profiling could be employed as a preliminary activity either on a live system, a write-blocked hard disk removed from a suspect system, or on an already acquired disk image. As we discussed in Sect. 3, one of our objectives is to reduce access to the disk by constructing the profile progressively in iterations, rather than all at once in a single step. Obviously, if executed on a live system, then profiling as a triage activity may produce a result without the necessity of imaging an entire disk. However, live forensics of all types have serious drawbacks [15], and any profiling software which was run on a live system would likely cause modifications to the system's hard disk. For this reason, a more forensically sound approach to preliminary examination using profiling may involve acquiring an image of the suspect's hard disk, and then subsequently analyzing the image, or, as we have said in Sect. 3, removing the suspect's hard disk, connecting it to a write blocker, and conducting a preliminary examination on a forensic workstation plugged into the write protected disk. Both of these approaches will ensure that the profiling tool can recover objects from unallocated space, and thus produce more complete results, while protecting the suspect's hard disk from modification. The latter approach will avoid the need to completely acquire the media.

Whether executed on a "live" system or an already acquired disk image, we envision that the iterative profiling process described in this work would be executed after triage tools have been employed to identify the most relevant media, but where a preliminary forensic examination is still necessary. In some cases, the triage tools alone may be adequate and avoid the need for a preliminary forensic examination altogether – for example, a child pornography investigation would be well served by a triage tool which simply displayed all the image files found on the suspect computer system to the screen, as discussed in Subsect. 2.1. Rogers et al. discuss the different sorts of data files which investigators should focus on retrieving first in the triage stage of an investigation into a wide variety of case types [8]. Beyond case-specific triage tools, when an investigator is searching for the presence of particular files, hash values pre-computed for known files can be computed for hash values of files found on the suspect's hard disk. Similarity hashing is an advanced hashing technique which aims to still compute a match when only minor differences exist between known files and the files on the suspect's hard disk [16]. Recent work into sector hashing also means that target files may even be identified on disks where the file system has been destroyed or otherwise cannot be referenced [17]. Computer profiling exists to supplement these triage techniques – not to replace them.

There is an inherent inefficiency in the approach described in Sect. 3, concerning the case of computer systems whose disks contain no evidence of interest to the forensic investigator. The inefficiency is that while the profiling process may be ended at an iteration $i < n$ in the case where a query returns a result of interest, for an uninteresting computer system (i.e. one containing no evidence), the profiling process will continue until iteration n, by which point the entirety of the computer's disk/s (or image/s of the disk/s) will have been read. This is unsatisfactory, but does not mean that iterative profiling is unsuited to triage, as for an uninteresting system, even though all n iterations of the profiling process will be executed, they will constitute the entirety of the examination of the system, bypassing the need to manually examine the file system in the traditional fashion.

One point to bear in mind is that forensic science is a search for both *inculpatory* and *exculpatory* evidence. Since the profile generated by the process we discuss in Sect. 3 generates as minimal a profile as is necessary, the querying process we discuss in Sect. 4 will only likely discover inculpatory evidence (i.e. evidence of a suspect's guilt), since a query will only be answered in the positive (or not at all). A complete examination of the computer system being profiled subsequent to the triage stage of the investigation may, however, uncover exculpatory evidence which explains away the results of this query. It is important to note that we propose profiling as a preliminary examination activity only – we do not suggest that it replace manual in-depth forensic examination, only that it supplements it.

6 Future Work

We are in the process of implementing a practical computer profiling preliminary examination tool which applies the approach described in this work to live computer systems, physical disks addressed through a write blocker, and also to images acquired before the profiling process. We are particularly interested in building a profiler tool which reads in a DFXML file (introduced in [18]), in order to enhance the inoperability between our profiling tool and existing open source forensic tools.

Once a practical profiling triage tool has been implemented, we would like to use it to generate computer profiles for data mining puposes. Data mining computer profiles to produce patterns which we associate with particular illicit usage scenarios. These patterns could be included with future versions of the profiling triage tool, so that similar patterns in the target hard disk's computer profile could be identified. Once a likely pattern match is identified, the profiling tool could characterize the target computer's profile according to the matching pattern. For instance, the tool might alert the investigator to the similarity between the system they are currently examining and a system they examined before.

Finally, we wish to expand upon the basic approach of testing for a particular sequence of events. We are interested in allowing investigators to specify large hypotheses which are chained together, and even including multiple slightly divergent timelines. This would allow for the expression of still more powerful queries than those discussed in Sect. 4.

7 Conclusion

In this work, we have proposed the use of computer profiling for the problem of preliminary digital forensic examinations. We have designed an iterative profiling process which gradually expands the computer profile as it progresses. Due to this iterative approach, the computer profile is immediately useful for some querying early on, before the examination is complete. This makes the computer profile functional straight away, and thus minimizes the chance that the entire disk will need to be read/written to before any useful results are produced by the profiling tool.

The approach is not without limitations. Like most triage and preliminary examination applications, some knowledge of the target computer sytem is required. In the case of this work, enough knowledge of the target system is required to compose formal evidentiary statements about the system for testing via computer profiling. Counter-intuitively, our approach takes longer (at least, takes more iterations) for the computers on which no interesting evidence is found than it does on systems which are interesting, as discussed in Sect. 5.

References

1. Carrier, B.: Defining digital forensic examination and analysis tools using abstraction layers. Int. J. Digital Evid. **1** (2003)
2. Casey, E., Ferraro, M., Nguyen, L.: Investigation delayed is justice denied: proposals for expediting forensic examinations of digital evidence. J. Forensic Sci. **54**, 1353–1364 (2009)
3. The American Heritage Dictionary of the English Language. Houghton Mifflin, Boston (2000)
4. Rogers, M.: The role of criminal profiling in the computer forensics process. Comput. Secur. **22**, 292–298 (2003)
5. Abraham, T., de Vel, O.: Investigative profiling with computer forensic log data and association rules. In: Proceedings of 2002 IEEE International Conference on Data Mining, ICDM 2002, pp. 11–18 (2002)
6. Marrington, A., Mohay, G., Morarji, H., Clark, A.: A model for computer profiling. In: Third International Workshop on Digital Forensics at the International Conference on Availability, Reliability and Security, Krakow, IEEE, pp. 635–640 (2010)
7. Batten, L.M., Pan, L.: Using relationship-building in event profiling for digital forensic investigations. In: Lai, X., Gu, D., Jin, B., Wang, Y., Li, H. (eds.) Forensics in Telecommunications, Information, and Multimedia. LNICST, vol. 56, pp. 40–52. Springer, Heidelberg (2011)
8. Rogers, M.K., Goldman, J., Mislan, R., Wedge, T., Debrota, S.: Computer forensics field triage process model. In: Proceeding of the Conference on Digital Forensics Security and Law, pp. 27–40 (2006)
9. Garfinkel, S.: Digital media triage with bulk data analysis and bulk-extractor. Comput. Secur. **32**, 56–72 (2013)
10. Gladyshev, P., Patel, A.: Finite state machine approach to digital event reconstruction. Digital Invest. **1**, 130–149 (2004)

11. Carrier, B., Spafford, E.: Categories of digital investigation analysis techniques based on the computer history model. Proc. Sixth Ann. Digital Forensic Res. Workshop (DFRWS '06) **3**, 121–130 (2006)
12. Buchholz, F., Spafford, E.: On the role of file system metadata in digital forensics. Digital Invest. **1**, 298–309 (2004)
13. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**, 558–565 (1978)
14. Marrington, A., Mohay, G., Clark, A., Morarji, H.: Event-based computer profiling for the forensic reconstruction of computer activity. In: Clark, A., McPherson, M., Mohay, G. (eds.) AusCERT Asia Pacific Information Technology Security Conference 2007 Refereed R&D Stream, Gold Coast, pp. 71–87 (2007)
15. Carrier, B.D.: Risks of live digital forensic analysis. Commun. ACM **49**, 56–61 (2006)
16. Roussev, V., Richard III, G., Marziale, L.: Multi-resolution similarity hashing. Digital Invest. **4**, 105–113 (2007)
17. Young, J., Foster, K., Garfinkel, S., Fairbanks, K.: Distinct sector hashes for target file detection. Computer **45**, 28–35 (2012)
18. Garfinkel, S.: Digital forensics XML and the DFXML toolset. Digital Invest. **8**, 161–174 (2012)