Finite State Machine Analysis of a Blackmail Investigation

Pavel Gladyshev [1,2]
Technical and Security Risk Services
Ernst & Young Ireland

Abstract

This paper gives informal introduction into the finite state machine approach to analysis of digital evidence and explores its use as a defence tool – for finding weaknesses in the forensic analysis performed by the opposing party. The key concepts of the finite state machine approach are reviewed, and an example analysis of a published case study is performed. It is shown how the described approach can be used to generate alternative scenarios of the incident.

Introduction

Digital evidence is encountered in many types of criminal and civil investigations. It has been argued, however, that currently widespread *ad-hoc* analysis of digital evidence is inappropriate in complex investigations, because it is error-prone and its findings are hard to defend in court. More scientific methods of analysis have been called for [2].

One such method is based on the application of finite state machine theory [3,4]. The idea of the method can be summarised as follows. Many digital systems, such as digital circuits, computer programs, and communication protocols can be described mathematically as finite state machines. A finite state machine can be viewed as a graph, whose nodes represent possible system states, and whose arrows represent possible transitions from state to state (see Figure 1).

Suppose that after the incident the system is found to be in some state. All possible scenarios of the incident can be determined by backtracing transitions leading to that state. In principle, investigator could perform investigation as follows:

1. obtain a finite state machine model of a system under investigation;
2. determine all possible scenarios of the incident by backtracing transitions[3];
3. discard scenarios that disagree with the available evidence.

To turn this idea into an algorithm, the relationship between the evidence and the finite state machine model of the system must be clarified. This has been done in

[1] E-mail for correspondence: pavel@gladyshev.info

[2] Disclaimer: this paper reflects the views of its author and not the views of Ernst & Young.

[3] If the final state is not fully specified (i.e. only certain properties of the final state are known), backtracing should be performed for all states that possess all known properties of the final state.

[3,4]. A formal notation for describing evidence has been defined, and the event reconstruction algorithm based on that definition has been presented.

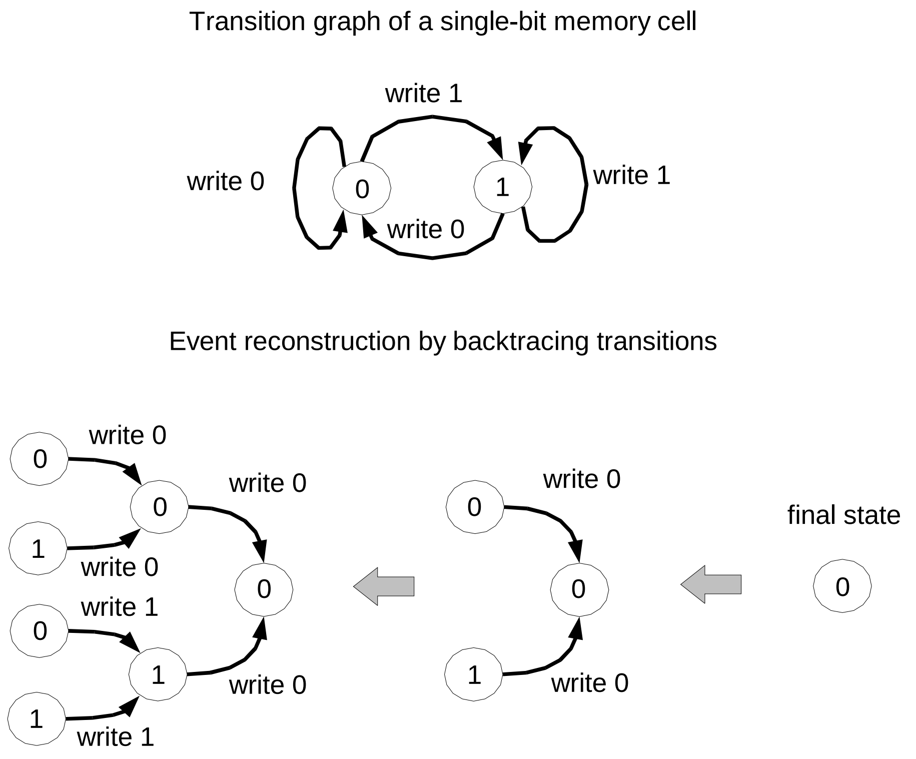

Figure 1. Event reconstruction by backtracing transitions

Contribution of this paper

Most publications on formal analysis of digital evidence, consider formality as a way to improve investigative reasoning. This paper, in contrast, explores the use of formal analysis to improve chances of legal defence. The aim of such analysis can be stated *as finding weaknesses in an informal analysis of digital evidence, so that that informal analysis can be attacked in legal proceedings*. Note that this "defensive" type of analysis does not have to be exhaustive as long as sufficient weaknesses are found. This allows considerable simplification of the formal model of the incident and, in turn, reduction of computational power required for analysis.

This paper demonstrates how this type of analysis can be performed using the finite state machine approach. This paper takes a published case study of a blackmail investigation [1], builds a state machine model of the incident, and explores it using the event reconstruction technique described in [3,4]. As a result of this exploration, two alternative scenarios of the incident are found that support innocence of the accused person, while agreeing with the evidence described in [1]. Potentially, these new scenarios could have been used by the defence to rebut the investigative theory.

Organisation of the paper

The rest of the paper is organised into four sections. The first section reviews the key concepts of the finite state machine approach to analysis of digital evidence. After that, section two describes a case study of blackmail investigation. The third section verifies the informal forensic analysis from the case study using the finite state machine approach. Finally, section four concludes the paper by highlighting advantages and disadvantages of the described method of analysis.

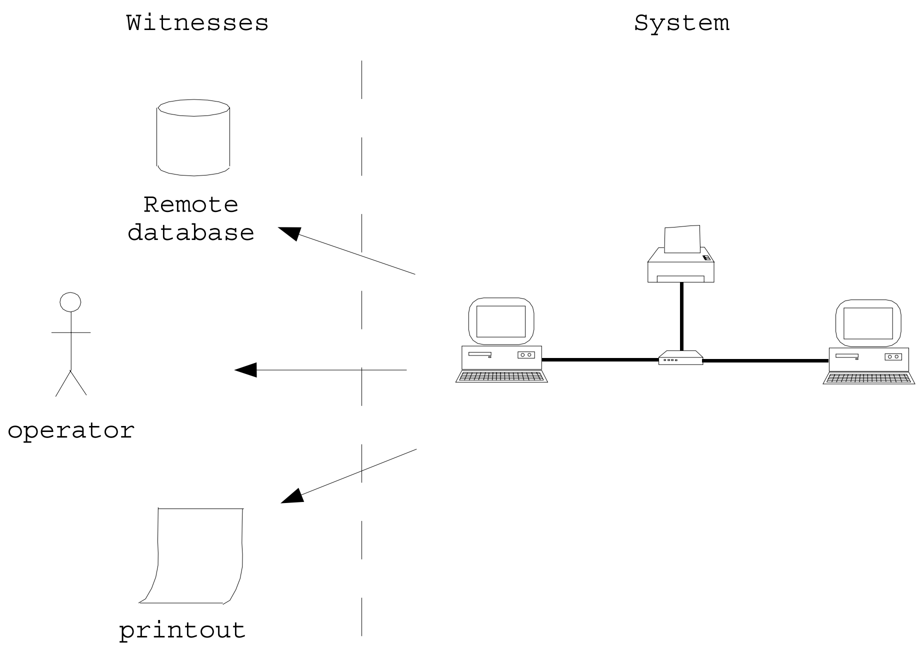

Figure 2. A generic view of the incident

Finite state machine approach to analysis of digital evidence

The finite state machine approach to digital investigation is based on mathematical modeling of the incident. It is an iterative process that consists of two distinct activities – creation of the incident model and analysis of the created model. Each of these activities is reviewed below[4].

Modeling of the incident

A generic finite state machine model of the incident (see Figure 2) consists of two parts:

1. a system under investigation, which is modeled as a finite state machine, and
2. witness observations, which are used for determining the set of possible computations of the system during the incident. The term witness is used

[4] The overview given in this section is largely informal. Please refer to [3,4] for the rigorous definition of concepts introduced in this section.

in a broad sense that includes human witnesses as well as natural phenomena and technical devices that bear traces of the system activity during the incident.

To create such a model, the investigator formalises his knowledge of system functionality as a finite state machine and uses specially designed evidential statement notation to describe the witness observations.

Finite state machine model of the system

Formally, a *finite state machine* is defined as a triple $T = (Q, I, \phi)$, where

- I is a finite set of all possible events,
- Q is a finite set of all possible states,
- $\phi : I \times Q \to Q$ is a transition function that determines the next state for every possible combination of event and state.

To create a finite state machine model of the system, the investigator must define events and system states that could have happened in the system during the incident and specify a transition function. Creation of the system model is an important step that determines the scope, the level of detail, and the complexity of the subsequent analysis.

Evidential statements

Evidential statements formalise evidence as observations about the behavior of the state machine during the incident. The system is viewed as a “translucent box,” whose state and behavior is *partially observable* to a number of witnesses. The knowledge of each witness is encoded as a “story” (observation sequence) about the state and change of observable properties of the system during the incident. Syntactically, evidential statements consist of observation sequences, which, in turn, consist of observations.

Observation

An *observation* is a statement that system behavior exhibited some property p *continuously for some time*. Syntactically, observation is a triple $o = (P, min, opt)$, where P characterises the set of all computations of T that possess observed property, and min and opt are non-negative integers that restrict length l of these computations: $min \le l < (min + opt)$.

Several special types of observation can be defined:

- *Fixed length observation* is observation of the form $(P, x, 0)$. Any computation satisfying it has length x.
- *Zero-observation* is observation of the form $(P, 0, 0)$. The only computation explaining it is empty computation denoted ε.

- *No-observation* is observation $\$ = (C_T, 0, infinitum)$ that puts no restrictions on computations that could have happened during it. C_T denotes the set of all possible computations of the state machine T and $infinitum$ is an integer constant that is bigger than the longest possible computation that could have happened during the incident.

Observation sequence

An *observation sequence* is a non-empty sequence of observations listed in chronological order:

$$os = (observation_a, observation_b, observation_c, ...)$$

Informally, an observation sequence represents an uninterrupted eyewitness story. The next observation in the sequence begins immediately when the previous observation finishes. Gaps in the story are represented by no-observations.

From the state machine point of view, observation sequence characterises the set of all computations that can be partitioned into sections where each section satisfies a corresponding observation within the observation sequence.

Evidential statement

An *evidential statement* is a list of observation sequences that refer to the same incident.

$$es = (os_x, os_y, os_z, ...)$$

An evidential statement combines restrictions imposed by all of its observation sequences. So, a computation satisfying one observation sequence must also satisfy all other observation sequences in the evidential statement.

From the state machine point of view, the task of the investigator can be defined as finding all possible computations of the given finite state machine T that simultaneously explain all observation sequences in the evidential statement es. A method for computing explanations of evidential statements is given in [3,4]. It is based on backtracing transitions using the inverse of the transition function of the state machine. Please refer to [3,4] for more discussion of evidential statements and related concepts.

Analysis of the incident model

Once the model of the incident is defined, the algorithm described in [3,4] can be used to compute possible sequence of events that could have happened during the incident. Note, however, that the determination of possible sequences of events is not the final goal of forensic analysis. The ultimate purpose of forensic analysis is, usually, to prove or disprove some claim about the incident. Clearly, this can be achieved by examining possible sequences of events:

- To *disprove* a claim the investigator has to show that there are no explanations of evidence that agree with the claim.
- To *prove* the claim the investigator has to show that all explanations of evidence agree with the claim[5].
- If there are some explanations of evidence that agree with the claim, and some explanations of evidence that disagree with the claim, the claim is neither proved nor disproved. Additional evidence is required to eliminate the explanations that cause the uncertainty.

A straightforward approach to proving or disproving some claim is to compute all possible explanations for the given evidential statement and check them all manually. However, this approach is impractical if the number of explanations is large. An alternative approach is to formulate the claim as an observation sequence, include it into the evidential statement, and try to find explanations that agree with both the evidence and the claim. If some explanations are found, they must agree with both the evidence and the claim, which means that the claim may be true. If no explanations are found the claim must be contradicting the evidence. This automatic hypothesis testing is used in the rest of this paper.

The next two sections show how the described approach to analysis of digital evidence can be used for finding weaknesses in the informal investigative reasoning. A blackmail investigation case study from [1] is used for that purpose.

Case study of a blackmail investigation

Presented below is a summary of the blackmail investigation described in [1].

A managing director of some company, Mr. C, was blackmailed. He contacted the police and handed them evidence in the form of a floppy disk that contained a letter with a number of allegations, threats and demands. The floppy was known to have come from his friend Mr. A. The police officers went to interview Mr. A and found that he was on holiday abroad. They seized the computer of Mr. A and interviewed him as soon as he returned into the country. Mr. A admitted that he wrote the letter, but denied making threats and demands. He explained that, while he was on holiday, Mr. C. had access to his computer. So it was possible that Mr. C. added the threats and demands into the letter himself to discredit Mr. A.

The contents of Mr. A computer's hard drive were examined. A total of 17 recognisable fragments of the letter located in various areas of the disk space were found. One of the fragments was a "clean" letter, without threats, stored in an active file. Other fragments contained threats and were found in unallocated disk space. It was concluded by the investigators that the fragments found in unallocated space were deleted versions of the letter. The conclusion follows from the fact that, when a file is deleted, FAT-based file systems do not erase the content of clusters previously used by the deleted file.

[5] Note that this is equivalent to disproving the negation of the claim.

The textual contents of the fragments were compared and placed by the investigators into a sequence that showed how the blackmail letter was created and modified through a series of revisions. The timestamps available in the file system indicated that all modifications happened before Mr. A went on holiday. The timestamps, however, were considered to be inconclusive. To fix the editing sequence in time, a form of event time bounding was used instead.

The time bounding relied on the properties of so-called *slack space*, which is unused space at the end of the last cluster of an active file. The formation of slack space is illustrated in Figure 3. One of the blackmail fragments was found in the slack space of another letter unconnected with the incident. When the police interviewed the person to whom that letter was addressed, he confirmed that he had received the letter on the day that Mr. A had gone abroad on holiday. It was concluded that Mr. A must have added the threats and demands into the letter before going on holiday, and that Mr. C could not have been involved.

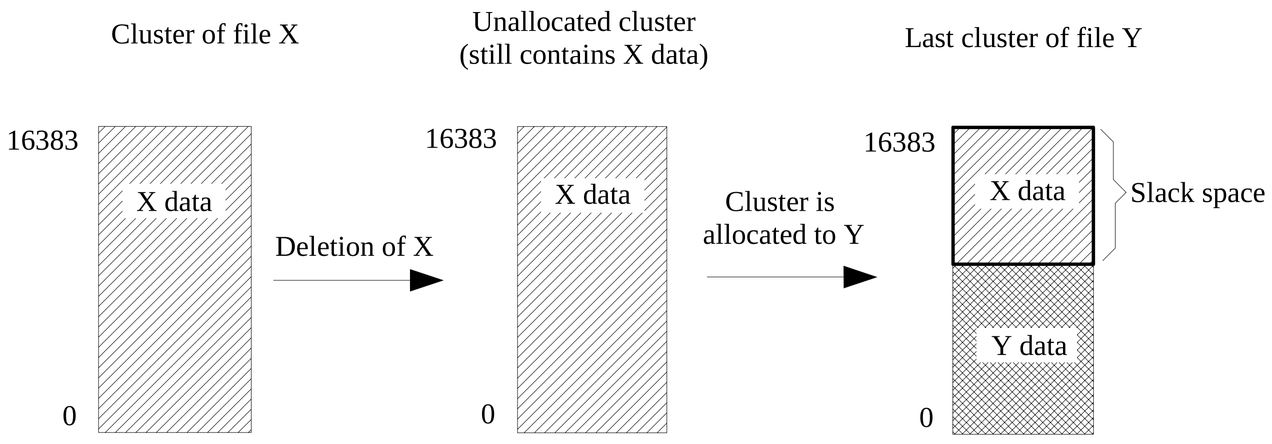

Figure 3. Formation of the slack space

The piece of reasoning that disproves Mr. A's theory seems to be as follows.

1. The letter unconnected with the incident must have been written after the letter with threats and demands, because of the way the slack space is formed.
2. Since the letter unconnected with the incident was received on the day the Mr. A had gone on holiday, it must have been written and posted at least two days before (because of the way the postal service works).
3. Based on 1 and 2, the letter with threats and demands must have been written before Mr. A went on holiday.

Although this reasoning seems plausible, there are other explanations of the available evidence that support Mr. A's alibi. These scenarios may have been used by Mr. A's legal team to rebut the investigators' theory. The next section shows how some of these scenarios can be identified using the finite state machine approach.

Finite state machine analysis of the blackmail investigation

The analysis presented in this section looks for weaknesses in the investigative reasoning from the blackmail example. The adopted approach is as follows.

1. Define a model of the incident.
2. Use automatic event reconstruction to find possible scenarios of the incident that agree with the theory that Mr. A has been framed.

Defining the model of the incident

The first step was to define a finite state machine that adequately describes the system under investigation. In the blackmail example, the functionality of the last cluster of a file was used to determine the sequence of events and, hence, to disprove Mr. A's alibi. Thus, the scope of the model can be restricted to the functionality of the last cluster in the unrelated file. The last cluster in a file can be modeled as an array of bits augmented with a length (see Figure 4). The array of bits represents cluster data. The length specifies how many bits from the beginning of the cluster are actually used by the file. Although real clusters do not have any length field, the number of data bits in the file's last cluster can be calculated from the file length and the known size of cluster in the file system. Zero length in the model would represent an unallocated cluster.

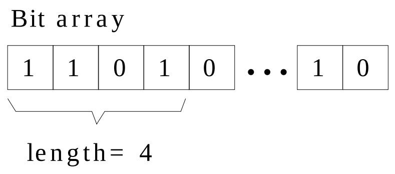

Figure 4. The initial model of the last cluster in a file

Simplifying the model

In the blackmail case study [1] each cluster is said to contain sixteen kilobytes of data. As a result, the number of distinct possible states of the cluster model shown in Figure 4 is $2^{(16384\times 8)} = 2^{131072}$. The exploration of this number of possibilities is clearly beyond the ability of modern computers. To make the analysis possible, the model had to be simplified.

One way to simplify the model is to artificially restrict the set of possible data objects that can be written into the cluster. Clearly, the resulting model will not be able to represent the full behavior of the cluster. However, *as long as every possible transition in the simplified model corresponds to a possible operation with the cluster*, any sequence of events that is possible in the model can also happen to the actual cluster[6]. This kind of simplification is acceptable for the purposes of the "defensive" analysis, because as long as the analysis finds *some* sufficiently

[6] Note that the inverse is not necessarily true. There may be sequences of operations with the cluster that cannot be represented in the simplified model.

credible explanations of Mr. A's alibi, it does not really matter if more of such explanations may have been missed.

States of the simplified model

Simplification of the lengths field

The reduced cluster model can store data objects of only three possible lengths:

$$LENGTH = \{0,1,2\}$$

where zero length means that the cluster is unallocated, the length 1 means that the cluster contains the object of the size of the unrelated letter tip, and the length 2 means that the cluster contains the object of the size of the data block with threats. All other sizes are disallowed. This, essentially, divides the cluster into two parts as shown in Figure 5; the left part that in the final state contains the unrelated letter tip, and the right part that in the final state contains the piece of the letter with threats.

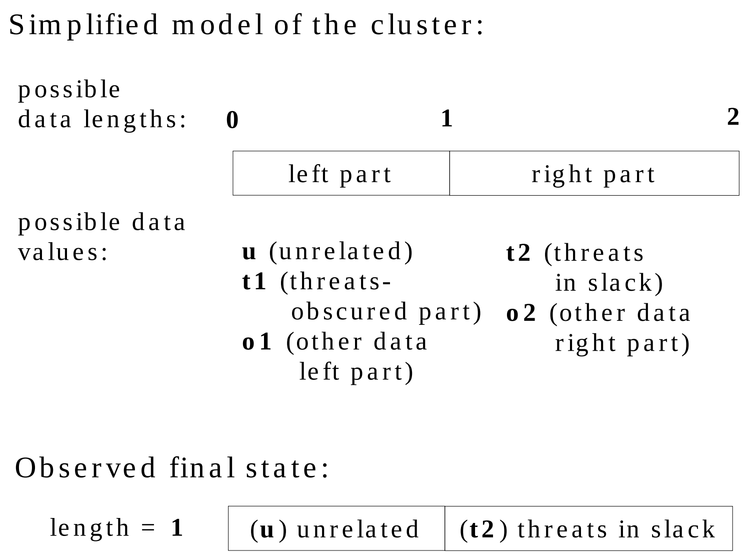

Figure 5. Simplified model of the cluster and its final state

Simplification of the cluster contents

Since the contents of the actual letters were not used to disprove Mr. A's alibi, these contents can be encoded by symbol sequences (u) and $(t1,t2)$ respectively, where u and $t1$ are stored in the left part of the model, and $t2$ is stored in the right part of the model. Special values $o1$ and $o2$ are used for representing some *other* datum that could have been written into the cluster before or in between the writes of the two letters. No other values are allowed in the model:

$$LEFT\_PART = \{u, t1, o1\}$$
$$RIGHT\_PART = \{t2, o2\}$$

It is assumed that the data values represented by symbols u, $t1$, $o1$, $t2$, and $o2$ are different from each other. The set of all possible states of the model is defined as

$$Q = LENGTH \times LEFT\_PART \times RIGHT\_PART$$

where operation $\times$ denotes Cartesian product of two sets.

Events of the simplified model

In FAT-based file systems, the state of the last cluster can be changed by three types of events: (a) direct writes into the cluster bypassing the file system, (b) writes into the file to which the cluster is allocated, and (c) deletion of the file. Each of these events is considered separately below.

Ordinary writes into the file

When cluster is modified as part of the file, the new data is written into consecutive locations starting from the beginning of the cluster. In the simplified cluster model, there are nine possible sequences that can be "ordinarily" written into the cluster:

$$WRITE = \{(u), (t1), (o1), (u,t2), (u,o2), (t1,t2), (t1,o2), (o1,t2), (o1,o2)\}$$

Apart from replacing one (left) or both (left and right) parts of the cluster model, every such event also sets the length of active data in the cluster to either 1 or 2 respectively.

Direct writes into the cluster

Cluster content can also be modified directly, for example, using a low-level disk editor. A direct write into the cluster is modeled as a complete replacement of the cluster content. The following events represent all possible direct writes into the cluster in the simplified cluster model:

$$DIRECT\_WRITE = \{d(u,t2), d(u,o2), d(t1,t2), d(t1,o2), d(o1,t2), d(o1,o2)\}$$

Since direct write into the last cluster of the file does not change the length of the file, a direct write into the cluster does not modify the length field of the model.

Deletion of the file

After a file is deleted, the information about the length of the file sooner or later becomes unavailable. This happens when the deleted file's directory entry is reused by another file, or when the FAT chain of the deleted file is broken. To model this eventual loss of length, the deletion event del is introduced. It sets the length of the cluster to zero.

The set of all events in the simplified cluster model is defined by

$$I = WRITE \cup DIRECT\_WRITE \cup \{del\}$$

where operation $\cup$ denotes the union of two sets.

Formalisation of evidence

In the above defined model of the cluster, the state observed by the investigators is represented by the triple $(1, u, t2)$. Let O_{final} denote the observation of this state by the investigators. Then the entire sequence of observations made by investigators in the blackmail investigation is formalised by the observation sequence os_{final}:

$$os_{final} = (\$, O_{final})$$

It states that nothing was observed about the cluster's content until forensic examination, which found that the cluster was in the state $(1, u, t2)$.

To complete formalisation of evidence one more observation sequence needs to be created. The observation sequence $os_{unrelated}$ says that the unrelated letter was created at some time in the past, and that it was received by the person to whom it was addressed:

$$os_{unrelated} = (\$, O_{unrelated}, \$, (C_T, 0, 0), \$)$$

where $O_{unrelated}$ denotes the observation that the "unrelated" letter tip (u) is being written into the cluster. The zero-observation $(C_T, 0, 0)$ represents the reception of the letter by the addressee.

The evidential statement for the blackmail example combines os_{final}, and $os_{unrelated}$:

$$es_{blackmail} = (os_{final}, os_{unrelated})$$

Finding explanations of Mr. A's theory

Mr. A's theory was that Mr. C could have added threats to the blackmail letter *after* Mr. A had written the unrelated letter and went on holiday. In terms of the cluster model it means that (1) when Mr. A wrote the unrelated letter into the cluster, there was no blackmail fragment in the cluster, and (2) after a while the blackmail fragment somehow appeared in the slack space of the cluster.

This claim was formalised as an observation sequence:

$$os_{Mr.A} = (\$, O_{unrelated-clean}, \$, O_{blackmail}, \$)$$

where $O_{unrelated-clean}$ denotes the observation that the "unrelated" letter (u) is being written into the cluster and, at the same time, the cluster does not contain the blackmail fragment; $O_{blackmail}$ denotes the observation that the right part of the model now contains the blackmail fragment ($t2$).

This observation sequence was added to the evidential statement $es_{blackmail}$, and possible explanations of the resulting evidential statement were computed using [5]. Several possible explanations of Mr. A's theory were found. One explanation is provided by the following sequence of events:

$$\ldots \xrightarrow{(u)} (1, u, o2) \xrightarrow{(u,t2)} (2, u, t2) \xrightarrow{(u)} (1, u, t2)$$

This sequence of events suggests that someone could have framed Mr. A by:

1. finding the unrelated letter, which was written by Mr. A earlier;
2. adding threats into the last cluster of that letter by editing it "in-place" with a suitable text editor (such as ViM [6]);
3. restoring the unrelated letter to its original content by editing it "in-place" again.

To understand this sequence of events, observe that certain text editors (e.g. ViM) can be configured to edit texts "in-place." In this mode of operation, the modified file is written back into the same disk blocks that were allocated to the original file[7]. As a result, the user can forge the content of the file's slack space by (1) appending the desired slack space content to the end of the file, (2) saving it, (3) reverting the file back to its original content, (4) saving it again.

Another explanation of Mr. A's claim is provided by variations of the following sequence of events:

$$\ldots \xrightarrow{(u)} (1, u, o2) \xrightarrow{d(u,t2)} (1, u, t2)$$

In this scenario, the threats are added into the slack space of the unrelated letter by writing directly into the last cluster using, for example, a low-level disk editor.

Both of the above scenarios allow the possibility that the threats were added while Mr. A was on holiday. Potentially, these scenarios could have been used by Mr. A's lawyers to rebut the investigators' theory. However, to convince the finder of fact in plausibility of these scenarios, additional evidence would probably be required to demonstrate that Mr. C had a capacity to perform the slack space forgery either himself or through the services of a skilled associate.

[7] When the modified file is bigger than the original file, additional disk blocks are appended to the end of the file by the operating system as necessary.

Concluding remarks

This paper described the finite state machine approach to analysis of digital evidence and explored its application for finding weaknesses in informal forensic reasoning. The described approach offers rigorous a mathematical framework for modeling the incident and provides automatic event reconstruction.

Despite potential advantages, such as analysis rigor and automation, the described approach requires additional effort to develop and simplify the model of the incident. The cost of this additional effort needs to be weighed against the potential benefits that may be obtained.

The techniques described in this paper are still at an early stage of development. Additional research is necessary to develop good practices for applying finite state machine analysis to "mainstream" investigations and to develop efficient and user-friendly software tools supporting these practices.

About the Author

Pavel Gladyshev is a Senior at the Technology and Security Risk Services division of Ernst & Young Ireland. Prior to joining Ernst & Young he conducted research in computer security and forensics at University College Dublin, Ireland. Pavel holds a PhD and MSc in computer science and a primary degree in computer engineering.

References

1. Bates J. Blackmail: case study. *International Journal of Forensic Computing* 1997; 1(2)

2. Casey E. *Digital Evidence and Computer Crime*. Academic Press; 2 edition; 2004

3. Gladyshev P. *Formalising event reconstruction in digital investigations*. PhD Dissertation; University College Dublin; 2004

4. Gladyshev P., Patel A. Finite state machine approach to digital event reconstruction. *Digital Investigation Journal* 2004; 1(2)

5. Gladyshev, P. *Event Analysis and Reconstruction in Lisp (EARL)*, 2005 http://www.gladyshev.info/smforensics/earl

6. Oualline S. *Vi iMproved (ViM)*. Sams publishers; 1st edition; 2001