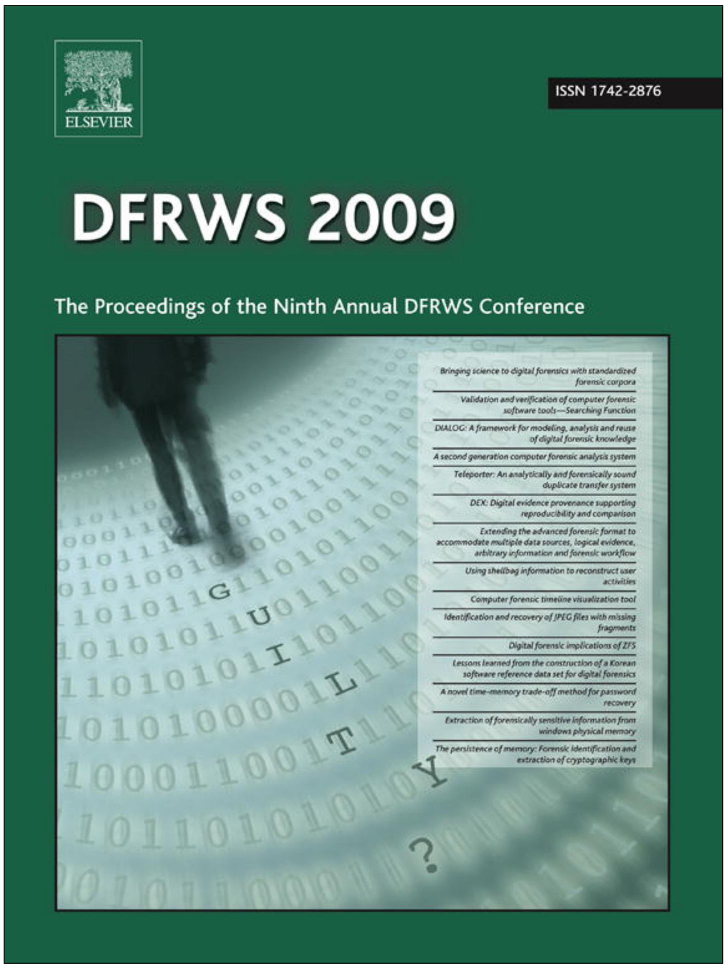


ELSEVIER

ISSN 1742-2876

DFRWS 2009

The Proceedings of the Ninth Annual DFRWS Conference

Bringing science to digital forensics with standardized forensic corpora
Validation and verification of computer forensic software tools—Searching Function
DIALOG: A framework for modeling, analysis and reuse of digital forensic knowledge
A second generation computer forensic analysis system
Teleporter: An analytically and forensically sound duplicate transfer system
DEX: Digital evidence provenance supporting reproducibility and comparison
Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow
Using shellbag information to reconstruct user activities
Computer forensic timeline visualization tool
Identification and recovery of JPEG files with missing fragments
Digital forensic implications of ZFS
Lessons learned from the construction of a Korean software reference data set for digital forensics
A novel time-memory trade-off method for password recovery
Extraction of forensically sensitive information from windows physical memory
The persistence of memory: Forensic identification and extraction of cryptographic keys

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Computer forensic timeline visualization tool

Jens Olsson[*], *Martin Boldt*

Blekinge Institute of Technology, School of Computing, Box 520, SE-372 25, Ronneby, Sweden

Keywords:
Computer forensic timeline
Event visualization
E-fraud
Timestamp
Chronological evidence
Time Variable
Time determination

A B S T R A C T

Computer Forensics is mainly about investigating crime where computers have been involved. There are many tools available to aid the investigator with this task. We have created a prototype of a new type of tool called CyberForensic TimeLab where all evidence is indexed by their time variables and plotted on a timeline. We believed that this way of visualizing the evidence allows the investigators to find coherent evidence faster and more intuitively. We have performed a user test where a group of people has evaluated our prototype tool against a modern commercial computer forensic tool and the results of this preliminary test are very promising. The results show that users completed the task in shorter time, with greater accuracy and with less errors using CyberForensic TimeLab. The subjects also experienced that the prototype were more intuitive to use and that it allowed them to easier locate evidence that was coherent in time.

1. Introduction

Computer forensic software available today rely on different methods when displaying identified evidence. Most of these applications have different views for different kinds of information. For example Forensic Toolkit has a special view for browsing E-mail, a special view for images displayed as thumbnails, a special view for file browsing etc. The increase in storage space on hard drives obviously impacts both the performance utilization and the time when carrying out such tasks. Both evidence visualization and browsing needs to be intuitive for the examiner to allow relevant evidence to be found in a timely manner. We think that this will be one of the most important aspects of a computer forensic tools in the future, and we also believe that the actual software has the potential to automate a great deal of the tasks currently carried out manually by the investigator.

The time parameter of evidence is similar to a least common denominator. For example, E-mail messages and chat conversations (Windows Messenger, ICQ and Trillian etc) all contain a timestamp for each message sent and received. All files on most modern computers contain timestamps when the file was first created, but not in Unix systems (Carrier, 2005; Casey, 2004). Considering that different files are used for different things, this is probably a great source of useful information about what tasks computers have been used for, and at what time. It is also possible to get timestamps for currently opened documents, executed software, visited web pages, all settings and data saved to the Windows registry etc. This is just to name a few examples and is a small subset of all evidence types, containing a timestamp that could be gathered.

By combining all these evidence types in a database and sorting it by time, we could create a new effective way of organizing and browsing computer evidence. This will also enable the examiner to choose a time span of interest based on previous investigation findings. The examiner is then able to see all events that happened on the particular computer, and possibly other computers if indexed in the same database, on and around that time. The examiner can browse back and forth from this time to find what happened right before and right after. It is also possible to zoom in and out to see exactly what happened at a certain time. Using such an approach we believe it is possible to find evidence faster, and to spot evidence patterns that would not even have been found otherwise.

[*] *Corresponding author.*
E-mail addresses: jens.olsson@rby.se (J. Olsson), martin.boldt@bth.se (M. Boldt).

1742-2876/$ – see front matter © 2009 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2009.06.008

We have created a tool that extracts all timestamps from a disk image and lists them on a timeline for the examiner. This could for example make it possible for the examiner to see E-mail messages, IM conversations and web browser history on the same timeline. The challenge here is to find all timestamps from different evidence types and to implement readers for them in the software. Another difficulty is to create an intuitive user interface that allows the investigator to see the relation between the evidence gathered. Of course the tool also needs to offer fast response times for large data sets.

To test this idea we created a prototype tool called CyberForensic TimeLab (CFTL) that scans a disk image for timestamps and then visualizes them on a graphical timeline, allowing the examiner to see what evidence of different types occur within the same time frame. Furthermore, we also evaluated our tool against a current state-of-the-art computer forensic product to see whether our prototype had any advantages over the state-of-the-art product. The actual evaluation was carried out using test subjects trying to solve a fictitious case, where half of the subject used the state-of-the-art product and the rest used our prototype tool.

The time variable of evidence is very important for the examiner regardless of the tool used. The criminal might therefore use a tool like Timestomp to clear all timestamps in the file system. Our tool however handles many kinds of file types and does not only rely on the file system timestamps to be useful to the examiner.

In the next section we discuss the related work available in the community, and we then move on to describe the structure and design of the prototype tool. In Section 4 we describe the empirical study carried out in an attempt to evaluate our prototype tool against another state-of-the-art tool. Section 5 presents the results from the evaluation, and we then discuss these results in Section 6. Finally we state our conclusions in Section 7, and present suggestions for future work in Section 8.

2. Related work

The amount of previous work focusing on forensic timelining tools is sparse and all of the major computer forensic tools lack the ability of presenting a timeline overview to the investigator.

EnCase, developed by Guidance Software, is considered to be the largest digital investigation software on the market. It can access a large variety of file systems and it also has the ability to find timestamps. However, without any feature visualize all timestamps on a graphical timeline.

Forensic ToolKit (FTK), developed by AccessData is also able to analyze different file systems, and its strength lies in the ability to search vast amounts of evidence in a timely manner, due to pre-indexation of the evidence. FTK includes special features to carve files from free space, and it includes specialized views for browsing particular evidence types such as images and E-mails. The tool allows the investigator to view timestamps in different file formats through any of the built-in viewers and also through plug-ins. However, none of these allow the investigator to get an overview of all timestamp data.

ILook Investigator, developed by IRS Criminal Investigation Division Electronic Crimes Program, handles raw disk images as well as some widely used commercial formats. It enables the examiner to browse files in various categories. It also has the advantage of enabling image creation of complicated systems such as servers with advanced RAID configurations etc. The software is not available to the public, but we have not found any information showing that this tool includes any graphical overview of the timestamp data.

Sleuth kit/Autopsy is a collection of Unix tools that are bundled together with a common graphical front end. By using these tools it is possible to analyze numerous file systems and for instance to recover deleted files. It is also possible to list all file accesses chronologically using the FLS tool, but without any timeline overview.

ProDiscover, developed by Pathways, is also able to analyze a number of file systems, but lacks a feature that present the user with a combined timeline view of all evidence.

In fact, the only computer forensic timeline tool that we managed to find was a tool called Zeitline, which is a so-called timeline editor developed by Florian Buchholz at Purdue University in 2004 (Buchholz and Falk, 2005). It can collect events from either syslog files or from file systems itself in the form of MAC times (Modified, Accessed, Changed). Zeitline require that the investigator manually add, sort and group the timestamps before they can be analyzed. Unfortunately the Zeitline tool lacks functionality to automatically add events, and to visualize events in a way so that investigators can find evidence that are coherent in time.

To summarize, current forensic tools are highly advanced applications that offer many different features for browsing specific kinds of information within evidence, and our prototype lacks most of these features. However, the goal of our prototype is in no way to cover all these features but rather to show the potential of a graphical overview of the timestamp data.

Due to the lack of this overview in current forensic tools the investigators are instead forced to do a lot of switching between alternative views in an attempt to manually compare evidence with each other. This is where we believe that an overview perspective based on the time variables of all evidence would be a highly valued feature by the forensic investigators. The closest tool we could find was Zeitline, which on the negative side requires quite a manual effort from the investigator when it comes to adding the individual timestamps to the tool. We therefore used an alternative approach in our prototype by having the tool automatically including all evidence when started. Therefore the investigator can begin analyzing the timeline, and zoom in on clusters of interesting events, without the need of any manual and labor intensive work.

3. CyberForensic TimeLab

In this chapter we present an overview of the design of the prototype tool. We discuss the internal structure and we also touch upon the interface that is used to interconnect the different parts of the prototype.

3.1. *Design decisions*

The prototype is divided into two separate parts called the Scanner and the Viewer. The purpose of the scanner is to scan a hard drive or an arbitrary amount of evidence recursively and identifying and storing the time stamps found. The result from the scanner is stored in XML format, which the viewer then reads, indexes and displays as a graphical timeline. We chose to implement the scanner using Perl, which is a script language well suited for text and data extraction. Perl also allows the scanner to run on a vast amount of different platforms, i.e. it is possible to keep the scanner while replacing the graphical viewer. The viewer is built in C# and .NET because Windows is considered to be the largest operating system in use for the audience of the application. It is also a language that will give a short time to market because of its large amount of ready-made functionality.

3.2. *The scanner*

The prototype scanner was implemented in Perl since it has proven to be a very suitable language for the problem at hand. Perl is additionally platform independent which makes it possible to connect the scanner to a native viewer for each operating system supported. During development the scanner has been developed and tested on both Mac OSX Leopard and Windows XP. The scanner is composed of four parts: the main executable, a data controller, a number of handlers for different file formats and utilities. At execution the scanner is supplied with a list of evidence files.

The main executable of the scanner traverses through this list and tries to identify the content of each file. It does this by querying the handlers one by one asking them if they can handle the specific content.

Each handler has a function called Validate and another function called Extract. The validate function is used by the main executable to query the handler whether it can handle a certain file. The extract function is called when the main executable know that the handler is suited for handling the specific format found, and will then go through the data structure searching for timestamps. It will also search for more data objects that it should send back to the main executable for inclusion in the evidence list. This enables the scanner to go through a whole hard drive, by first identifying the MBR, then the individual partitions, then the files on the partition and so on.

The handlers are closely coupled to source objects, i.e. the sources where the data is stored. All evidence is transformed into source objects to maintain information about the location from where the evidence originates. The source objects contain a chain of byte offsets, byte lengths as well as the origin. The origin can be either a file on the host system or another source object, which is one of the key concepts for allowing the recursive searching. This makes it possible to begin with a large disk image of a hard drive, and then recursively traverse through each one of the partitions, e.g. an NTFS file system. In this file system the scanner might identify another disk image, which in turn is containing a FAT32 file system that the scanner starts to analyze recursively.

Our prototype includes the following handlers for different data objects:

FAT – Handles both FAT16 and FAT32 volumes. Extracting all files together with their Read, Access and Write timestamps.

JPEG – Handles image files in JPEG format. Extracts the dates contained in the EXIF part. This often contains the time of photo shooting and the time it was transferred to a computer.

MBOXArchive – Handles e-mail messages in MBOX format. This format is used in Unix machines as well as in Mozilla Thunderbird and Eudora to name a few graphical E-mail clients. The MBOX handler extract timestamps of E-mails both sent and received from all E-mail servers on the way between the sender and recipient. These in-transit timestamps can be used to for example verify against the server logs that an e-mail message has really been sent. For obvious reasons it is much more difficult for computer criminals to alter their traces on many servers than on just one single computer.

MBR – Handles the Master Boot Record, but since the MBR does not contain any timestamps none could be extracted. However, the handler will go through the partitions and extract these as source objects for other handlers to extract.

NTFS – Handles NTFS partitions by extracting all files together with their Read, Access, Write and MFTChange timestamps.

WindowsEventLog – Handles windows system event logs. Each log entry in these log files contain a time stamp and details about the event.

WindowsIndexDat – Handles the index.dat history files created by Microsoft Internet Explorer. It will extract timestamps together with the URL for each visited webpage. It can even extract some deleted entries since they sometimes can be left in the file by the browser.

WindowsLiveMessengerHistory – Handles the conversation history logs created by Windows Live Messenger. Each message has a timestamp, the message content as well as a sender and receiver address.

WindowsLNK – Handles shortcuts in Windows. Each shortcut file contains a copy of the Read, Access and Write timestamps of the file pointed to by the shortcut file.

WindowsRegistryHive – Handles registry hives (Carvey, 2005; Mee et al., 2006). Each registry key is extracted together with its LastWriteTime telling when it was last changed. These hive files can be found in the windows folder and in each user's folder, but they are also commonly found in the system restore points often created by windows computers giving the examiner access to many historical timestamps and values for each key.

UnixLog – Unix has a large number of log files in text format that begins with a date/time and ends with a description of the event. This handler is generic for parsing those kinds of files, such as HTTP logs (Log Files).

In Fig. 1 the Document Type Definition (DTD) for the format used as interface between the scanner and viewer is presented, and Fig. 2 shows a small sample of how the evidence collection is encoded using this format. The interface between the scanner and the viewer is an XML file created by the scanner. The XML file is very straightforward and contains a list of all ⟨Evidence⟩ tags. Each one contains a number of ⟨Timestamp⟩ tags describing all timestamps found in the evidence file. It also optionally contains a number of ⟨Data⟩ tags, which add special extracted information about the evidence. Furthermore, a number of ⟨Chunk⟩ tags specifying where exactly in the parent evidence that the current evidence was found. The

```
<!ELEMENT EvidenceCollection (Evidence*)>

<!ELEMENT Evidence
(Chunk+,Timestamp*,Data*)>
<!ATTLIST Evidence title CDATA #REQUIRED>
<!ATTLIST Evidence type CDATA #REQUIRED>
<!ATTLIST Evidence id CDATA #REQUIRED>
<!ATTLIST Evidence parent CDATA "">

<!ELEMENT Chunk EMPTY>
<!ATTLIST Chunk from CDATA #REQUIRED>
<!ATTLIST Chunk to CDATA #REQUIRED>

<!ELEMENT Timestamp EMPTY>
<!ATTLIST Timestamp type CDATA #REQUIRED>
<!ATTLIST Timestamp value CDATA #REQUIRED>

<!ELEMENT Data EMPTY>
<!ATTLIST Data name CDATA #REQUIRED>
<!ATTLIST Data value CDATA #REQUIRED>
```

Fig. 1 – Document Type Description (DTD) for interface between the scanner and the viewer.

```
<xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
href="CyberForensicsTimeLab.xsl"?>
<!DOCTYPE EvidenceCollection SYSTEM
"CyberForensicsTimeLab.dtd">
<!--
Created by CyberForensics TimeLab.
Copyright(C) 2008 Jens Olsson.
-->
<EvidenceCollection>
  <Evidence title="Volumes/NTFS.img"
type="MBR" id="1" parent="0">
    <Chunk from="0" to="5368709120" />
  </Evidence>
  <Evidence title="Partition?0?"
type="NTFS" id="2" parent="1">
    <Chunk from="32256" to="5354657280" />
  </Evidence>
  <!-- More evidence is located here -->
</EvidenceCollection>
```

Fig. 2 – Sample XML code from the scanner.

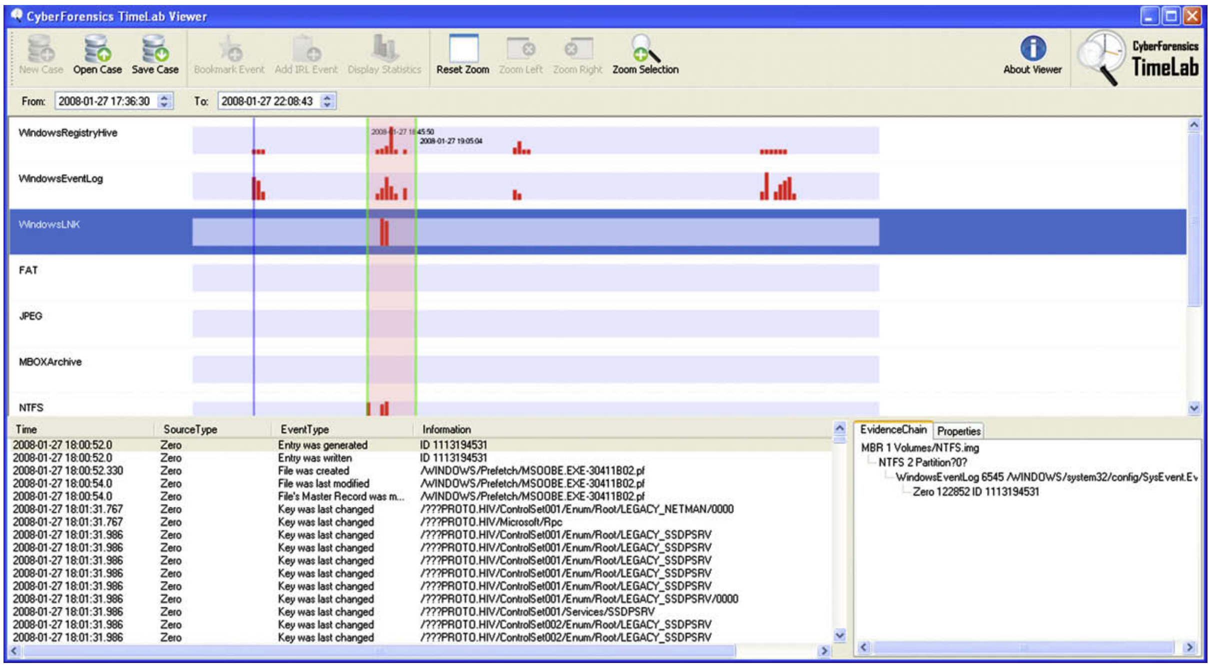
sequence of ⟨Chunk⟩ tags specifies this with offsets and lengths together forming the evidence data.

Evidence added manually have a parent of "0" while evidence found by the scanner have the parent set to the evidence it was automatically found in. This makes it possible for anyone that has the XML file and the image file to extract the data part for a particular evidence without having to embed the full evidence data into the XML file.

3.3. *The viewer*

The prototype viewer was implemented using C# and .NET and has the ability to load the XML files created by the scanner. It consists of a main user interface with a graphical timeline together with a panel for more detailed information. The timeline is implemented as a histogram using bars to represent the number of evidence at a specific time. In the detail viewer panel all timestamps are listed in a chronological list together with a short text describing each. The viewer also maintains the relationship between the timestamps and its source object so that the actual data of the evidence can be easily found for further analysis. The viewer also maintains a tree of evidence in memory to assist the investigator in getting an overview of the situation. The tree view can also be used for identifying which object that contain a certain file, and in which object that object has been found and so on.

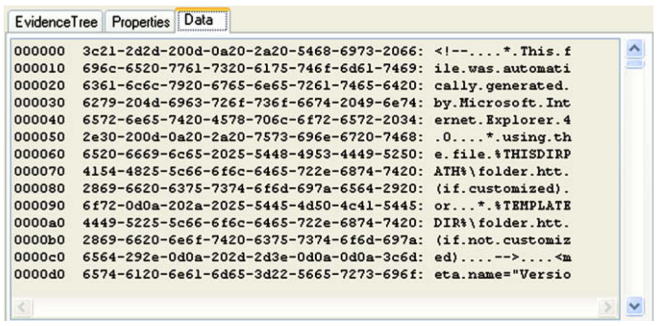
The investigator is initially presented with an overview timeline showing all evidence found, as shown in Fig. 4. It is possible for the investigator to highlight interesting parts of the timeline and zoom in to get greater detail of that particular time span. Fig. 3 is zoomed out to its maximum where it is



Time	SourceType	EventType	Information
2008-01-27 18:00:52.0	Zero	Entry was generated	ID 1113194531
2008-01-27 18:00:52.0	Zero	Entry was written	ID 1113194531
2008-01-27 18:00:52.330	Zero	File was created	/WINDOWS/Prefetch/MSOOBE.EXE-30411B02.pf
2008-01-27 18:00:54.0	Zero	File was last modified	/WINDOWS/Prefetch/MSOOBE.EXE-30411B02.pf
2008-01-27 18:00:54.0	Zero	File's Master Record was m...	/WINDOWS/Prefetch/MSOOBE.EXE-30411B02.pf
2008-01-27 18:01:31.767	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_NETMAN/0000
2008-01-27 18:01:31.767	Zero	Key was last changed	/???PROTO.HIV/Microsoft/Rpc
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Enum/Root/LEGACY_SSDPSRV/0000
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet001/Services/SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet002/Enum/Root/LEGACY_SSDPSRV
2008-01-27 18:01:31.986	Zero	Key was last changed	/???PROTO.HIV/ControlSet002/Enum/Root/LEGACY_SSDPSRV

```
MBR 1 Volumes/NTFS.img
 └ NTFS 2 Partition?0?
    └ WindowsEventLog 6545 /WINDOWS/system32/config/SysEvent.Ev
       └ Zero 122852 ID 1113194531
```

Fig. 3 – The graphical timeline view presented to the investigator.



```
000000  3c21-2d2d-200d-0a20-2a20-5468-6973-2066:  <!--....*.This.f
000010  696c-6520-7761-7320-6175-746f-6d61-7469:  ile.was.automati
000020  6361-6c6c-7920-6765-6e65-7261-7465-6420:  cally.generated.
000030  6279-204d-6963-726f-736f-6674-2049-6e74:  by.Microsoft.Int
000040  6572-6e65-7420-4578-706c-6f72-6572-2034:  ernet.Explorer.4
000050  2e30-200d-0a20-2a20-7573-696e-6720-7468:  .0....*.using.th
000060  6520-6669-6c65-2025-5448-4953-4449-5250:  e.file.%THISDIRP
000070  4154-4825-5c66-6f6c-6465-722e-6874-7420:  ATH%\folder.htt.
000080  2869-6620-6375-7374-6f6d-697a-6564-2920:  (if.customized).
000090  6f72-0d0a-202a-2025-5445-4d50-4c41-5445:  or...*.%TEMPLATE
0000a0  4449-5225-5c66-6f6c-6465-722e-6874-7420:  DIR%\folder.htt.
0000b0  2869-6620-6e6f-7420-6375-7374-6f6d-697a:  (if.not.customiz
0000c0  6564-292e-0d0a-202d-2d3e-0d0a-0d0a-3c6d:  ed)....-->....<m
0000d0  6574-6120-6e61-6d65-3d22-5665-7273-696f:  eta.name="Versio
```

Fig. 4 – The details panel with the hexadecimal viewer.

possible to see the four distinct occasions when the computer that is being investigated has been used. Zooming in on one of these occasions would reveal thousands of events, but by zooming in further it is possible to see what the computer-user has been up to. The timeline view is actually divided into several sub-timelines placed on top of each other where each one represent a certain type of events, e.g. file modifications from the file system or instant messaging events. Furthermore, it is possible to remove any sub-timelines from the screen to make it less distractive for the investigator.

Below the sub-timelines is a text-based list of evidence that reflect what is currently visible on the timeline. When the investigator finds something interesting it is possible to use this list to find evidence that exist closely in time. As soon as a timestamp is selected, the tree view shows what other timestamps the current object contains, as well as other properties that has been extracted.

If the investigator needs to go into more detail there is a hexadecimal viewer available, showing the whole file in either hexadecimal or ASCII format, as shown in Fig. 4. The hexadecimal and ASCII viewer is able to find the data to display by traversing the evidence tree and taking all data positions into account. When finally arriving to the top evidence that has no parent, it knows exactly which byte ranges that should be joined and displayed. This work even in very complex evidence trees, for example if the evidence is contained in a NTFS disk which itself resides in an image inside a FAT32 disk image.

To increase performance the viewer maintains an ordered index of all timestamps. Each timestamp is then connected to an evidence object, which resides in another ordered list with the evidence id as index. Each evidence object has one parent evidence, building a tree in the memory. This way of storing the data makes it very efficient to create the timeline visible in Fig. 3. To shorten the response times even further a cache of the graphs is maintained.

The list of evidence is implemented using a ListView control in C#. To keep the speed and memory consumption at an acceptable level, the list of evidence could not be loaded right into the control. There is a special mode of the ListView control called virtual mode that is enabled. This mode does not keep any data inside the actual control, but instead uses a callback function where the control can ask for information regarding the evidence data. By using the callback functionality it was possible to link the ListView control right into our indexed data structure, which makes scrolling very fast.

A similar approach is used for the hexadecimal viewer. Instead of loading all data inside a file, the hexadecimal viewer retrieves the file size. So, when using the scrollbar, the textbox will load different fragments of the file on the fly, i.e. instead of loading all file data to begin with. If we did not use this approach it would be impossible to display a 10 GB file since it would require 10 GB of RAM memory and probably reach other limitations of what the control can handle.

3.4. *Extendibility with plug-ins*

The prototype tool is fully extendable due to its use of plug-ins in the form of handlers for different file types and different file systems. All these plug-ins are written as Perl modules that are automatically loaded by CyberForensic TimeLab (CFTL) when launched, as long as they are placed in the dedicated plug-in directory. Each plug-in has an interface based on two functions, which is used by the main program for interaction. The first of these two functions is *validate()* that takes an evidence object as parameter and returns whether the plug-in can parse the evidence object or not. The other function is *extract()* which also takes an evidence object as a parameter. The *extract()* function parse the supplied evidence object for timestamps, which then is sent back to CFTL. It is even possible for the *extract()* function to identify new evidence objects (in addition to the timestamps). These new objects are also returned to CFTL for future evaluation, which might be done by yet another plug-in. This approach makes CFTL fully extendable and allows the tool to find evidence in complex structures. One such example could be a timestamp that is found in a registry hive or server log file, located on a NTFS file system, which is located in a disk image inside a FAT32 file system on a physical hard drive.

4. Prototype evaluation

To investigate whether the use of an evidence timeline is advantageous or not compared to traditional methods we carried out an initial user test. The test was based on that a set of users should investigate a fictional crime case using *either* our CyberForensic TimeLab prototype or Forensic Tool Kit (FTK), which is a commercial and reputable state-of-the-art computer forensic software. Our pilot evaluation of the prototype tool includes a limited number of subjects, in this case 12. However, the results indicate whether or not the use of a forensic timeline is an interesting technique to investigate further.

The actual test subjects were selected using a convenience sampling strategy and the subjects were then randomly divided into the two groups called *TimeLab Investigators* and *Traditional Investigators*. Each group had six members and none of them knew anything about each other or that they were divided into groups.

The actual test consisted of a fictitious case and a set of questions that the subjects should investigate and answer. At the time of the test all subjects received a disk image from the fictional suspect's computer. At this time the subjects were also provided with the questions that they were asked to answer. We also instructed the test subjects that time was of the essence and that they only had 20 min to investigate the evidence and answer the questions. Even though we restricted the time to 20 min we let all test subjects complete the task even if they exceeded the 20 min time frame, due to the limited number of subjects.

4.1. *Fictional test scenario*

For the sake of the prototype evaluation we developed a simplified crime scenario, which of course is not as extensive as real-world scenarios. This simplified approach was necessary so we could carry out the test within the limited time frame and with limited resources. Despite the fact that the case is of limited complexity it will still give us an indication of the investigated tools' capabilities.

The fictitious case was based on that law enforcement officials tracked down a cyber criminal when she was transferring a large amount of money from a stolen credit card at an Internet café in New York. The credit card was previously reported stolen and the credit card number was monitored in major payment services and banks, which made it possible to get immediate notification when it was used. Unfortunately she was long gone when law enforcement arrived at the café, but the café personnel recognized her from a photo and they also remembered which of the public computers she had used. The investigators took the computer back for analysis. One of the café's staff could also tell the investigators that the perpetrator used the computer between approximately 06:00 PM (18:00) and 06:30 PM (18:30) on the 13th of August 2008. Initial investigations show that the criminal (called Evelyn Evans) created her own account on the computer. The bank/payment service associated with the card reported that she logged on to the bank at 2008-08-13 18:17:05, which could be used as a starting point to solve the task.

The test case provided has been prepared to contain evidence that are coherent in time, which could be seen as an advantage for CyberForensic TimeLab compared to for instance FTK. In our fictive case the 12 test subjects knew the approximate time when computer had been used for illegal purposes, which could be a relative advantage for the subjects relying on the CyberForensic TimeLab prototype. However, we believe this reflects reality and thus provide a realistic test case, i.e. it should not pose a significant problem for the evaluation.

4.2. *The disk image*

In addition to the previous description of the task, each test subject also got access to a VMWare system that represented the evidence computer from the Internet café. The disk image was prepared with VMWare server in the following way:

Windows XP was installed
Mozilla Firefox was installed
MSN Messenger was installed
Microsoft patches and updates were installed
System was restarted
System clock was set to 2008-08-13 18:00
Evelyn logged in
Evelyn checks her E-mail where she receives a credit card number and instructions from bd@freenet.foo (MBX file generated)
Evelyn hides the MBX file in a hard-to-find folder
Evelyn connects to MSN where she receives an account number.
Evelyn surfs to www.americasbigbank.com and transfers the money.
Some additional MBX archives were added to the disk image so that there would be some e-mail to scan through; unfortunately MBX archives generated from Apple Mail did not get recognized in Forensic Tool Kit, meaning that they only added to the complexity for the TimeLab Investigators.

4.3. *The task*

The actual task presented to the test subject consisted of six different questions. Each question addressed different parts of the fictitious case that the test subjects only could know about if they successfully managed to locate the evidence on disk image they received. All test subjects were asked to answer the questions as accurately as possible in a timely manner.

Which payment system/bank was used to transfer the money?

Which bank account did Evelyn Evans transfer the money to?

What was the stolen credit card number?

When exactly was the bank withdrawal done?

Are there any indications that the job was ordered by someone else, if so, who?

Could you identify any additional people involved in the crime, if so, who?

4.4. *User survey*

When the task was completed, the subjects that participated in the prototype evaluation were asked to participate in an online survey to provide some feedback about the strengths and weaknesses in each program. The survey also included a number of questions to determine the subjects' previous knowledge about computer forensic and their computer knowledge in general. The survey was completed with a 100% response rate.

4.5. *Population of subjects*

We used convenience sampling when selecting twelve subjects that were divided into two different groups to evaluate the FTK tool and out prototype. All subjects were university students or former university students. As mentioned earlier we used an online survey to learn more about the subjects included in the study. From this survey we could conclude that 11/12 subjects had used computers for more than 10 years, while the last one had used computers between 6 and 10 years. We asked the subjects about how many programming languages they had knowledge in, as a simple measure of their computer experience. We found out that 1/12 had knowledge in 2–3 programming languages, while 5 subjects had knowledge in 4–6 different languages. The rest,

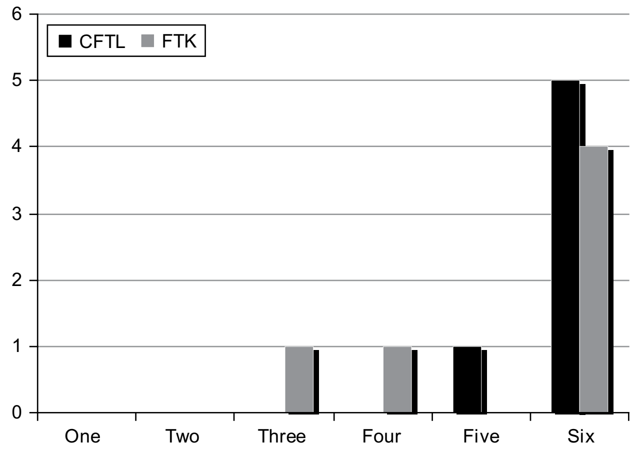


Fig. 5 – Number of correctly answered questions by the two groups using Forensic Tool Kit (FTK) and CyberForensic TimeLab (CFTL).

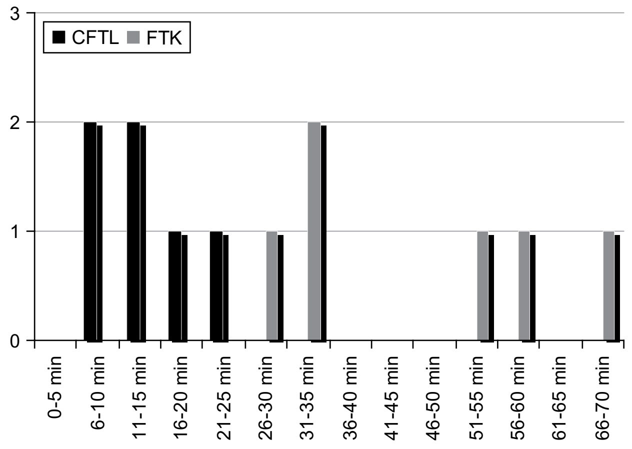


Fig. 6 – The amount of time that the subjects in the two groups needed to solve the fictive case.

6 subjects, are skilled in 7 or more programming languages. This indicates that the subjects have firm knowledge about computers and programming, which vouches for a fair comparison between FTK and CyberForensic TimeLab.

We also asked the test subjects about their knowledge in computer forensic and it turned out that 4/12 subjects (two in each group) had no prior experience about computer forensic, while 5 subjects had ‘‘little experience’’ (2 subjects in the prototype group and 3 in the FTK group). The remaining 3 subjects had ‘‘some experience’’ in computer forensic (2 from the prototype group and 1 from the FTK group). Even though none of the 12 subjects were much experienced in computer forensic we managed to get an even division between the two subject groups.

Finally, we asked the test subjects about their day-to-day usage of hex editors as an indication of how experienced the subjects are in computer forensics. These results show a slight advantage for the 6 subjects that evaluated the CyberForensic TimeLab prototype, since only 1 of them claimed to never use hex editors, compared to 4 subjects in the FTK group. Furthermore, 5 of subjects in the CFTL group use hex editors at least once a month compared to only 2 subjects in the other group. However, for the fictive case used in this study there was no need to have any knowledge about hex editors to solve the case.

5. Results

Based on comments from the test subjects the task did not seem to be very hard to solve and the majority answered all questions correctly. Two test subjects (one from each group) did confuse the buyer and the boss in the case. This has however been treated as correct since it is a mistake in the understanding of the case and not in the software usage. However, this didn't give any advantage for any of the tools and will therefore not impact the result of this comparison. Both test subjects still found the relevant information with the tool provided, which in this study is more important.

Some test subjects also confused the credit card number with the bank account number. This has also been accepted as correct for the same reason. This error was done only once and by the group that relied on the FTK tool. This has also been treated as correct. This might give an advantage to FTK but on the other hand, if we are right and this was a misunderstanding it would be unfair to FTK not to treat it as correct. Fig. 5 presents the number of correct answers for the two groups indicating quite equal result between the compared tools. A slightly better result can be seen for the TimeLab investigators that got a total of 35 correct answers compared to the Traditional investigators' 31 correct answers.

The test subjects were asked to enter their start and stop time in the answer sheet. The instructor also did a sanity check of the time specified. Fig. 6 shows some quite interesting results in the amount of time the two groups needed to solve the case. As the result shows the subjects that relied on the CyberForensic TimeLab solved the task faster than all subjects using the FTK tool, i.e. CyberForensic TimeLab is clearly faster than FTK when working with this type of case. The average time for solving the case in FTK is 45 min while the average time when using CFTL is 14 min, i.e. the case was solved in CFTL in approximately one third of the time compared to FTK. This difference is even statistically significant when using a two-sample t-test of 97.5%, despite the limited number of test subjects.

5.1. *Usability*

The survey presented to the test subjects after they completed the fictitious case included a number of questions on how they experienced the software they had been using. In Table 1 the test subjects have estimated how difficult they thought the task was to solve using the tool provided to them. To clarify we added weights to the answers so that ‘‘Very difficult’’ corresponds to -2, ‘‘Neutral’’ as 0, and ‘‘Very easy’’ as $+2$. We then calculated the mathematical average for each group, which lies within the range -2 to $+2$ (inclusive). The weighted result, $-1/6$ compared to $+1$, show that the subjects found it harder to solve the case using FTK than with CFTL, respectively. In other

Table 1 – Subjects opinions regarding the difficulty in solving the case, finding coherence between events, and the intuitiveness of FTK and CFTL.

	Yes indeed	Yes	Neutral	No	Not at all	Weighted average
Was it difficult to solve the task using FTK?	1	1	2	2	0	1/6
Was it difficult to solve the task using CFTL?	0	0	1	4	1	−1
Was it difficult to see coherence between events using FTK?	1	2	1	1	1	1/6
Was it difficult to see coherence between events using CFTL?	0	0	1	4	1	−1
Was the GUI intuitive in FTK?	0	2	2	2	0	0
Was the GUI intuitive in CFTL?	1	2	1	2	0	2/6

words, the subjects found CyberForensic TimeLab to be more easy to use than Forensic Toolkit in this particular case.

The subjects were also asked to estimate how easy it was to see coherence between different evidence using the tool they were assigned, see Table 1. Again, when analyzing the weighted result we see that the subjects using FTK found it somewhat difficult based on the score +1/6, compared to the CFTL subjects score of −1. In other words, it is clearly easier to see coherence between evidence when using CyberForensic TimeLab compared to Forensic Tool Kit. This is probably because all events are placed on the same timeline view in CFTL, while FTK use several different views without any overview.

The subjects were furthermore asked to estimate how intuitive the GUI was in the tool they used. The results in Table 1 show that FTK received a weighted average of 0, while CFTL got +2/6. This means that CyberForensic TimeLab had a slight advantage over FTK, but there is plenty of room for improvements in both tools. It can also be read in the open text comments about CyberForensic TimeLab that the zoom function (using the left mouse button to mark and the right mouse button to make a selection) is quite unintuitive, and that the prototype would be more intuitive if this were handled by only using the left mouse button, i.e. click and drag.

5.2. *Improvement suggestions*

The online survey also included an open section where the subjects could provide additional feedback. When analyzing these comments it is clear that three of the CFTL test subjects thought the zoom function was quite unintuitive, since it used the left mouse button to mark the first selection position and the right mouse button for the selection length. This should probably be changed, or at least be configurable for the users, even though using both buttons is more flexible.

Three test subjects also mentioned that there were some issues with the updating of evidence properties when selecting evidence. There is currently, in special situations, a risk that information from the previously selected evidence is still displayed if the user does not update manually. This was however already a known issue and a fix is planned.

Two test subjects also requested a function for panning left and right and to undo zoom-ins, two functions that seems reasonable to implement and that would add good value to the user of the tool. Another test subject requested search functionality, which we believe would increase the speed of using the software significantly.

6. Discussion

We believe that the single most interesting finding in this study is that the test subjects solved the task significantly faster using CFTL than when using FTK. The actual difference was approximately a magnitude of three. If the same results would be achievable in real cases it could clearly decrease the amount of time that law enforcement needs to solve real and complex cases. This would in turn allow the examiners to work more efficiently and therefore complete more cases, than would otherwise fit within the budget.

We have also seen that more questions were answered correctly using CFTL, i.e. 35 correct answers compared to FTK's 31. The difference can seem small, but if the priority instead is to keep the number of incorrectly answered questions to a minimum the difference is 1 for CFTL and 5 for FTK, i.e. the subjects using CFTL committed five times more errors than the ones using CFTL. In other words, it seems like it is easier to find answers to questions about a case with CFTL than with FTK, i.e. investigators could find evidence using CFTL that they would not otherwise have found.

We have also seen that the test subjects felt it was easier to solve the task with CFTL than with FTK. Since there are people in the test groups both with and without previous experience of computer forensics this would probably in reality result in that it will be easier to find people who can perform computer forensic investigation with CFTL, and also that it is easier for trained computer forensic examiners to solve cases.

CFTL also makes it easier for examiners to see coherence between evidence. The evaluation test showed that it was much easier to see coherence between the evidence on the timeline in CFTL than it was in FTK, +1 compared to −1/6 respectively. When the law enforcement officials are investigating a case in traditional software they might search for files changed at the target time span, then they might search E-mail or data from instant messaging software. However, chances are that they simply would not have the time to make all the searches and that they therefore would miss for instance to check the registry for changed keys, which would have shown to them that the suspect connected an USB hard drive and transferred data important to the case. With CFTL all these events are displayed on the same time line so when a specific time is investigated, all relevant evidence are showed in the same place, giving an valuable overview of the case.

In total we have found five different aspects where CFTL is performing better than FTK in the user tests. For instance, CFTL made it possible to solve the case three times faster, and only 1 question was answered incorrectly compared to 5 for FTK. If CFTL would work this well on a real case after improving its stability and increasing the supported file formats these two aspects would result in decreased costs and increased accuracy. We do have to keep in mind that CFTL is only a prototype and that it still contains bugs that make it somewhat unreliable, but the work on improving CFTL continues.

6.1. *Evaluation of the scanner*

The performance of the scanner is acceptable although it is implemented using Perl, a script-based language. In our own benchmarks it took 5 min and 30 s to scan a 5 GB NTFS disk image, of which 2.7 GB were used space, with roughly 20,000 files. The scan resulted in 152,288 indexed and timestamped events. These benchmarks were executed on a 2.4 GHz Intel Core 2 Duo processor, and during the execution the CPU utilization on the cores that was used never went below 90%. Unfortunately it is hard to set this benchmark in relation to a tool such as FTK since these products scan for different things. FTK for instance handles more file types and also scans the free space available on the disk, which obviously takes more time.

The choice of making a plug-in based system was a good idea, for instance when we had some issues with handlers, for example NTFS; it was easy to test the other parts of the system and disabling those not working properly at the moment. It also enforces a good breakdown of the software where each module handles a very isolated task. This plug-in based system would also make it possible for the end users to choose which modules to enable and in that way make the scans faster. Furthermore, it is possible to publicly release an interface so that third parties can create add-ons, for example end users can write their own scanner modules, for custom data-types, that could be nicely plugged-in to CFTL. We have also made some experiments with trying to unit test the separate add-on modules by themselves and initially this seems to be a good idea. Unit tests ought to be an important part of a forensic tool since it is required to prove that the tool is operating in a correct way.

6.2. *Evaluation of the viewer*

Our chosen development language, C#, was a sound choice since it allowed for easily creating a customized timeline component, based on the MS Windows GUI. XML also was a well-suited format to use for data exchange between the scanner and viewer since it is non-proprietary and can easily be debugged. It is also completely transparent what is actually transferred, which allows for verification of the data in an investigation. The C# language itself also has built in support for XML, which made it a simple task importing this data into the viewer.

We managed to speed up the GUI when large amounts of information should to be presented at once on the timeline. This was successfully done with different caching techniques and by relieving the windows controls from holding all the actual data in memory at once, but rather handing this task to a specialized controller.

The user test has also showed that the viewer was somewhat intuitive and it was possible for all test subjects but one to correctly answer the questions included in the case. This promising evaluation results could probably be increased further after taking the subjects feedback into account and after properly testing the prototype.

7. Conclusions

We created a prototype tool that we call CyberForensic TimeLab, which is divided into a scanner and a viewer. The scanner scans hard drives for evidence and generates index files, which then are read and displayed to the investigator by the viewer. The scanner can handle file systems like NTFS, FAT32 and FAT16. In addition to listing files in the file system, the scanner can also find E-mail, instant messaging, browser history and a number of other data formats, which it will add to the index file together with all their timestamps. The viewer will then read the indexed data that contains all timestamps and produce a graphical timeline on which all evidence is displayed. The investigator can then use the timeline to zoom in and browse evidence at a certain timeframe. It is also possible for the examiner to filter out just one or a set of evidence types to make the information easier to grasp.

In an attempt to evaluate the tool we set up a pilot experiment including 12 subjects that were asked to compare our prototype with a well-known and reputable state-of-the-art forensic tool called Forensic Tool Kit (FTK). Even though the number of test subjects was limited to 12 subjects, our results indicate that the use of a timeline in digital forensic tools allows investigators to solve a case significantly faster and with higher accuracy. We therefore think it is interesting to further investigate this technique.

The test subjects that used CFTL solved the task at hand in approximately 1/3 of the time compared to the subjects relying on the state-of-the-art tool. Furthermore, the test subjects also gained higher accuracy in their understanding of the case when using CyberForensic TimeLab than compared to FTK. Through a survey we also identified a set of improvement suggestions that would further increase both the usability and the reliability of the prototype. The bottom line is that visualizing evidence on a forensic timeline increases the examiners' ability to see correlation between different events on a computer, which definitely could prove valuable in a digital crime investigation.

8. Future work

First of all we plan to extend the evaluation test by including more test subjects. When it comes to the prototype itself we need to develop unit tests for the complete functionality of the prototype. This is essential to guarantee its correctness and to sort out the bugs that are still left in for example the NTFS handler. Regarding the functionality of the prototype we have identified a number of interesting areas, which are listed below.

8.1. *Integrations with existing software*

An interesting possibility would be to integrate the functionality of CFTL into an existing computer forensic utility. In this way it might be possible to reuse evidence extraction from the tool and link the timeline to the evidence files in the tools native interface. The timeline could in this way be a very good complement working together with the current interface of the tool.

8.2. *Automatic pattern search*

All evidence is already stored in a uniform way, making it easy for a computer to handle. To add automatic pattern search to the viewer it would be possible for the investigator to automatically search for certain predefined patterns of suspicious activity, helping the investigator to spot interesting parts of the timeline more efficiently. Interesting techniques to investigate include subareas to Artificial Intelligence such as Data Mining and Machine Learning.

8.3. *Time zones and time deviance*

Currently the prototype considers all timestamps to be in the same time zone. This will of course become an issue when gathering evidence from several different computers that each have their own system clocks and are configured to various time zones, i.e. the time deviance needs to be handled in a controlled manner (Buchholz and Tjaden, 2007).

8.4. *Other timestamp sources*

Other sources where timestamps should be investigated and implemented, e.g. printers and other network equipment, mobile phones, and personal calendars.

8.5. *Release of CFTL*

We plan to release a first version of CFTL in late 2009. However, at the writing of this all details are not decided yet. For more information please visit: http://cftl.rby.se

Acknowledgments

We would like to thank Martin Hylerstedt at Blekinge Institute of Technology in Sweden for his assistance in proof-reading this work.

REFERENCES

Buchholz F, Falk C. Design and implementation of zeitline: a forensic timeline. In: Proceedings of the 2005 digital forensic research workshop. New Orleans LA, US: DFRWS; August 2005.

Buchholz F, Tjaden B. A brief study of time. Digital Investigation 2007;4(Suppl. 1):31–42.

Carrier B. File system forensic analysis. Crawfordsville: Addison-Wesley Professional; 2005.

Carvey H. The windows registry as a forensic resource. Digital Investigation 2005;2(3):201–5.

Casey E. Digital evidence and computer crime. 2nd ed. Boston: Elsevier Academic Press; 2004.

Log Files – Apache HTTP Server. Available from: http://httpd.apache.org/docs/1.3/logs.html.

Mee V, Tryfonas T, Sutherland I. The windows registry as a forensic artefact: illustrating evidence collection for internet usage. Digital Investigation 2006;3(3):166–73.

Timestomp–forensics wiki. Available from: http://www.forensicswiki.org/wiki/Timestomp.

Jens Olsson is working as a software developer with focus on mobile devices and frameworks. He received a M.Sc. in Security Engineering and a B.Sc. in Software Engineering from Blekinge Institute of Technology. His computer interest involve security, server management and programming.

Martin Boldt is a PhD student at the School of Computing, Blekinge Institute of Technology in Sweden. His research focus on collaborative countermeasures against malicious software, which involves reputation systems and automatic analysis of End-User License Agreement using machine learning techniques.