7th International Conference on Computer and Knowledge Engineering (ICCKE 2017), October 26-27 2017, Ferdowsi University of Mashhad

# A survey on digital evidence collection and analysis

Somayeh Soltani
Department of Computer Engineering,
Ferdowsi University of Mashhad,
Mashhad, Iran
somayeh.soltani@mail.um.ac.ir

Seyed Amin Hosseini Seno
Department of Computer Engineering,
Ferdowsi University of Mashhad,
Mashhad, Iran
hosseini@um.ac.ir

***Abstract*—The growth of digital technologies results in the growth of digital crimes. Digital forensics aims to collect crime-related evidence from various digital media and analyze it. This survey reviews several tools and methods in the literature which extract pieces of evidence from the system and analyze them. It also discusses the challenges during the collection and analysis of low level data from the compromised system.**

***Keywords—digital forensics; digital evidence; event reconstruction***

## I. INTRODUCTION

Digital forensics is the science of detection, extraction and analysis of the pieces of evidence from the digital media, and is one of the critical requirements in cyber space. One important goal of digital forensics is to prepare court accepted reports [1]. Three important components of digital forensics are hard disk, memory and network forensics which record and analyze some tracks from behaviors of cyber criminals.

Digital forensics has nearly three decades of history. While the first attempts in digital forensics were to collect pieces of evidence from the compromised system, recent researches in digital forensics mostly concentrate on the analysis of gathered evidence and preparation of verifiable, valid, and reproducible reports for the court. In a very general categorization, digital forensics attempts can be divided into two groups of 1) evidence collection and 2) evidence analysis.

Several methods and tools are designed to collect evidence from the system. Therefore, enormous volume of information is extracted. The big amount of the low level information extracted from the system is very challenging. Examining millions of pieces of low level information to extract high-level info is a time consuming and exhausting work. Therefore, some automatic methods are required to extract high-level information from raw data [2-4]. This survey aims to review and classify research works on digital evidence collection and analysis and indicate some challenges posed to future work on the topic.

## II. EVIDENCE COLLECTION

Most attempts in the first years of introduction of digital forensics have collected pieces of evidence from different parts of compromised systems. These attempts have resulted in design and implementation of various free and commercial digital forensics tools. Therefore, several tools have been designed which give information about various items such as running processes, network connections, system drivers and services, log files, registry, and logged on users. Some of these tools such as netstat, tasklist, ipconfig, and driverquery are native, meaning that they are provided as part of the operating system distribution.

### *A. Collecting the Content of Hard Disk and Physical Memory*

In newer attempts, digital investigators try to collect the total content of hard disk or some partitions of it. In this regard, some digital forensics tools such as dd, Encase [5], and FTK Imager [6] are introduced.

However, there are situations in which the widely accepted forensic methodology of turning off a computer and then acquiring a bit-stream image of hard disk is not a sufficient option. For example, there are critical servers which cannot be shut down. Moreover, some forensically valuable information cannot be found in hard disk. Information such as decoded passwords and unobfuscated malware are just found in physical memory. Therefore, several digital forensics tools such as FTK Imager, MDD [7], and Moonsols [8] are designed which collect physical memory content.

### *B. Extracting Data Structures from Hard Disk/Physical Memory Images*

Images of hard disk and memory contain valuable forensics information. Therefore, some methods and tools try to extract various data structures from these images. For example, Richard M. Stevens et al. [9] extract Windows command line details from memory image. Similarly, James Okolica et al. [10] extract Windows clipboard from memory image. The technique proposed by Robert Beverly et al. [11] carves network packets and associated data structure from memory of popular operating systems. The approach proposed by Andrew White et al. [12] focuses on user space portions of memory image and identifies all user space allocations.

978-1-5386-0804-3/17/$31.00 ©2017 IEEE

The Sleuth Kit (TSK) [13] parses the hard disk image and extracts various important information about hard disk and its partitions. It is used behind the scenes of many open source and commercial forensics tools. The Volatility [14, 15] toolset is an open source digital forensics tool which is used to analyze memory images. Pyflag [16] is another digital forensics tool which analyzes hard disk image, memory image and network traffic.

### C. Integrity Issues

During data collection, the integrity of the collected data is an important challenge. Several papers [17-20] address the problem of bypassing the integrity of the collected data using anti-forensics techniques. For example, Johannes Stuttgen et al. [20] state that many of memory collection techniques are subverted by a number of simple anti-forensics techniques. The reason is that these acquisition techniques require code to be run on the compromised system. Johannes Stuttgen et al. then present a new memory collection technique which is based on direct page table manipulation and PCI hardware introspection. The technique proposed by Johannes Stuttgen et al. doesn't rely on operating system facilities, so it is more resistant to anti-forensics tricks. Namheun Son et al. [19] state that Android data acquisition methods do not guarantee data integrity. They develop an Android data acquisition tool which ensures the integrity of acquired data.

### D. Scalability Issues

Another challenge regarding data collection is the scalability. Increasing the volume of digital media results in growing the volume and variety of digital evidence. Therefore, conventional manual examination of all pieces of evidence may require several days. Several papers [21-25] address this issue and propose some kind of triage of digital evidence.

Vassil Roussev et al. [21] use similarity digests for the purposes of forensics triage. Fabio Marturana et al. [22] propose a triage methodology which aims at automating the categorization of digital media on the basis of plausible connections between digital evidence and crimes under investigation. Timothy Vidas et al. [23] propose OpenLV which helps investigators to speed up the loading of digital forensics images. OpenLV facilitates the review of digital evidence during investigation process. Owen Brady et al. [24] develop DESO, Digital Evidence Semantic Ontology. DESO acts as a repository and classifier of digital evidence artefacts.

## III. Evidence Analysis

After extracting pieces of evidence from different parts of compromised system, it is time to analyze them. The extracted pieces of evidence are usually low level and are not in the same format. Moreover, the extracted data has a large volume. Therefore, the investigator encounters large volume of raw and low level information. Examining millions of pieces of evidence to reconstruct the occurred events on the compromised system is a cumbersome and time consuming task. In order to facilitate digital investigation, some processes on low level collected data are required.

Several recent researches on digital forensics have addressed the issue of reconstructing events. The extracted pieces of evidence from the system are very various, and researchers usually focus on some of them. Many researchers have reconstructed events using information extracted from hard disk [2-4, 26-42]. Some researchers have used information extracted from memory image [10, 12] and network traffic [43, 44] to reconstruct events. Moreover, some researchers work on other platforms such as Android [45] and distributed and cloud platforms [46-49].

It should be noted that the extracted evidence from hard disk is more reliable and is more referenced in the court than the extracted evidence from memory. The reasons are as follow: 1) while there are some hardware products for acquisition of hard disk in a pristine state, most of methods of acquisition of memory are software-based, so they cannot provide pristine memory image and they are subverted by anti-forensics techniques [20]. 2) While hard disk image contains some permanent information, memory image is just a snapshot of system state. One can even find traces of events took place months before in a hard disk image [50]. However, it doesn't mean that the analysis of memory image isn't valuable. For example, memory image analysis is useful for detection and analysis of malware.

Since reconstructing events using information from hard disk are more referenced in the literature, this survey focuses on event reconstruction methods which are based on extracted information from hard disk.

## IV. Event Reconstruction using Hard Disk

Hard disk based event reconstruction methods can be divided into two categories. 1) Methods which reconstruct events by finding pattern or signature of running applications using low level pieces of evidence [2-4, 26-31]. 2) Methods which reconstruct events by finding correlation among low level pieces of evidence [32-36, 41]. Moreover, some papers on event reconstruction use miscellaneous approaches [37-40, 42].

### A. Event Reconstruction using Signature of Application

Some papers have tried to reconstruct events using signature of applications or events. These papers usually have done some initial tasks to find signature of various applications. Then they have used these signatures to find which applications have been run on the compromised system or which events have taken place.

Papers in this category use different low level information to find the signature of an application. Some papers [3, 26, 27] use file system metadata to find signature of application and reconstruct events. Besides file system metadata, some papers [4, 28, 29] use some information from other parts of hard disk such as registry or log files. Some papers [2, 30, 31] augment reconstruction methods by using various timing information within some files such as browser histories, prefetch files, pdf files, and LNK files.

The technique proposed by Sven Kalber et al. [3] tries to find fingerprints of different actions and applications using MACB timestamps in NTFS metadata. Running each application or doing each action leave traces on system which can be used for reconstruction purposes. Some of these traces are the MACB timestamps in NTFS metadata for each file and folder. Running each application updates MACB timestamps of many files and folders which are accessed, modified or even created during the running of the application. These updates make the fingerprint of application.

The technique of Sven Kalber et al. compares file system metadata just before and after running the application and considers the difference as the fingerprint of application. It uses Fiwalk [27] digital forensics tool for this purpose. However, since Windows is a multi-process operating system, several programs may run simultaneously. These programs leave traces on file system metadata and have impact on the generated fingerprint of the application. Thus, to remove the impact of other programs on the generated fingerprint, the process of creating fingerprint for each application is repeated one hundred times.

Similarly, Joshua I. James et al. [28], extract file and registry timestamps of each application using Microsoft Process Monitor[1]. The signature of an application is created using four categories of timestamp updates: 1) always updated timestamps, 2) timestamps updated on the first run only, 3) irregular update of timestamps and 4) usage-based timestamp updates. Timestamps in category 1 are the core part of the signature of the application. Because of inconsistent nature of other categories, they are considered as supporting evidence.

Methods in [3, 28] try to find fingerprint or signature of an application. However, since these methods use a few timestamp updates to create signature of application, they are easily subverted by anti-forensics techniques. For example, an attacker can easily hide his traces on the system by changing related timestamps using some tools like BulkFileChanger[2].

In order to reconstruct events, the method proposed by Sven Kalber et al. [26] extracts file system timestamps and sorts them in ascending order. This method considers a new event has taken place if the gap between two consecutive timestamps is more than 20 seconds. Then it clusters the timestamps within each event according to their filenames and paths using DBSCAN [51] algorithm. Although the method of Sven Kalber et al. tries to present a quick and easy method for event reconstruction, it doesn't detect events properly. It just considers a 20-second gap as a new event which isn't correct.

CFTL [30] extracts FAT and NTFS timestamps and times within various types of files including EXIF file, Link files, MBOX archives and registry files. This computer forensics timeline visualization tool designs special extractors for each type of file. log2timeline [31] is another efficient digital forensics timeline tool which supports 26 different input files including Event log, Chrome browser history, EXIF, McAfee antivirus log file, Firefox bookmark file, Firefox browser history, Internet Explorer browser history, and Open XML.

Tools such as [30, 31] extract various timestamps from different parts of hard disk. They don't usually search signature of an application or an event. However, Plaso (incorporating log2timeline) uses tagging rules to support event reconstruction. The output of these tools are used by researchers for event reconstruction.

PyDFT [2] is a python digital forensics timeline tool. Similar to two previous tools, PyDFT is capable of extracting timing information from various types of files. PyDFT is also capable of reconstructing high level events from low level ones. It preserves provenance of high level events to show their constituent low level ones. PyDFT has two steps: step-1) low level event extraction, and step-2) high level event reconstruction. PyDFT for step-1 has implemented extractors for various type of files. For each type of file, its extractor extracts timing information from the file and adds it to the low level event timeline. PyDFT for step-2 has defined some rules for each high level event. An analyzer script searches for these rules in the low level event timeline. However, defining these rules for different events is certainly a time consuming and exhausting work.

A neural network method for event reconstruction is presented in [4]. Inputs of neural network are extracted pieces of information from file system metadata, registry, and log files. Different applications are run separately on virtual machine to generate train and test data. FileSystemWatcher library of .NET technology is used to capture file system activity patterns of various applications. Feed forward neural network and recurrent neural network are trained using nine inputs. These inputs are filename, type, size, path, activity modes (access, creation, modification, renaming and deletion), date, time, registry key, and registry value. Each neural network has one input layer, two hidden layers, and one output layer. The output layer has a neuron which can be either '1' or '0'. The output '1' means that the particular file is manipulated by the application and output '0' means that the particular file isn't manipulated by the application.

To reconstruct events, two neural network and Bayesian network are presented in [29]. Then the performance of these methods are compared. However, it seems that methods in [4, 29] don't solve any problem of digital forensics. In these methods, different neural networks are trained for different applications. Trained neural network can determine whether file $x$ was manipulated by application $A$, but it cannot determine whether application $A$ was run on the compromised system or not.

### *B. Event Reconstruction using Event Correlation*

Some researchers work on event correlation in the field of digital forensics [32-36, 41]. There is a difference between event correlation in digital forensics domain and other computer security application domains such as intrusion

[1] https://technet.microsoft.com/en-us/sysinternals/processmonitor.aspx
[2] http://www.nirsoft.net/utils/bulk_file_changer.html

detection and network management. While events in intrusion detection have similar formats, events in digital forensics are extracted from different parts of system and thus have different formats. Therefore, it is difficult to repurpose existing approaches of event correlation to digital forensics [32, 35]. In order to correlate events in digital forensics, data can be stored either in databases [32, 33] or in ontologies [35, 36, 41], and then some queries or processes are performed on them.

ECF [32], extracts pieces of evidence from heterogeneous resources. Thus, these pieces of evidence have different formats. However, in order to correlate events, some common attributes of these pieces of evidence such as Time, Subject, Action, and Object are stored in a Canonical table. Since there is other valuable information in extracted pieces of evidence, additional log-specific tables are included. ECF provides parsers for various types of files such as Apache Server Log, Windows 2000 Security Log, Door Log, Browser Cache Logs, *NIX SYSLOG, POP3, SSH, sendmail, and smap. These parsers insert the aforementioned common information into the Canonical table and insert additional valuable information into log-specific tables. ECF provides some querying facilities in a user friendly manner. Unfortunately, event correlation isn't automatic in ECF, and the investigator should perform some queries and try to find correlations among events by himself.

FACE [33] is a framework for automatic evidence discovery and correlation from various resources such as memory dumps, network traces, hard disk images, log files, and configuration files. FACE provides one correlation engine and some parsers, including ramparser, pcap parser, and configuration/log file parsers. The correlation engine uses an internal database to store and relate various conceptual untis (such as user identities, filenames, etc.) together. FACE provides five main data views: users, groups, processes, filesystem, and network captures.

The methods in [32, 33] use a database and a SQL query interface to represent data and correlate events. Unfortunately, databases do not explicitly represent semantic of data. Moreover, it is hard for users to query the database and find event correlations.

AssocGEN [34] extracts metadata from digital artifacts belonging to hard disk images, internet browser logs (i.e. history log and cache log), and network packet captures. It then uses these pieces of metadata to determine associations between these forensics artifacts. AssocGEN provides algorithms to group the related artifacts. These algorithms seek metadata matches between various digital artifacts and group them. Digital artifacts in a single source can be grouped based on different metadata types. For instance, for document files, grouping may be done based on various types of metadata such as author, title, and file size.

If a single metadata match is found between two or more digital artifacts within the same source of digital evidence, a set of digital artifacts named a *similarity pocket* is created. The similarity pocket is identified by the metadata tag name. There may be different similarity pockets for a single source of digital evidence for different metadata. Similarity pockets may also overlap partially. If there are two overlapping similarity pockets within a single source of digital evidence, these are merged into a new bigger set, which is named a *similarity group*. Overlapping similarity groups across multiple sources are merged into an *association group*.

The increase in volume and complexity of forensics data requires semantically strong representational models and automated methods of correlating such forensics data. In [35], a forensics ontology using the Web Ontology Language (OWL) is presented. FORE [35] proposes a correlation tool based on rules to identify causal relationships between events; The event A causes the event B if A happens before B. FORE ontology consists of two base classes, an Entity class that represents objects in the world, and an Event class that represents changes in state of objects over time.

Although representing forensics data using an ontology is an attractive idea in FORE, using a rule-based system isn't appropriate. Constructing such rule set is a hard and time-consuming task.

In order to represent forensics data in a structured and standard representation, [36] proposes an OWL2-based ontology, called ORD2I. ORD2I ontology consists of three layers: 1) the Common Knowledge Layer, 2) the Specialized Knowledge Layer, and 3) the Traceability Knowledge Layer. Each layer consists of classes and some properties linking these classes.

The Common Knowledge Layer stores common knowledge about events that occurred during an incident. In spite of variety of sources, events have a number of common characteristics. The Common Knowledge Layer contains these common characteristics such as time information, resources and the subjects involved in the event. This layer has a public class, Entity, which is the parent of three subclasses Event, Subject, and Object.

The Specialized Knowledge Layer stores specialized information of objects. This layer consists of various classes such as File, Account, Web, Communication, and RegistryKey to represent various objects. The Traceability Knowledge Layer stores information about how the investigation is done. Investigative activities, agents involved in investigation, and data used during investigation are some of these pieces of information. The aim of the Traceability Knowledge Layer is to satisfy some legal requirements, i.e. by storing all actions taken at each stage of the investigation, it ensures reproducibility of the results. The InvestigativeOperation class represents any task conducted during an investigation.

Benjamin Turnbull et al. [41] propose an ontology to make event reconstruction process easier. The proposed ontology represents information about people, computers, files, websites, and events. It facilitates the understanding of users and social networks. The technique proposed by Benjamin Turnbull et al. tries to reconstruct two types of events: user events such as web browsing, and sending email and system events such as logins, start-ups, and shutdowns.

It seems that the use of ontologies is a good way to represent events. The formal description of ontologies

automates the reconstruction of events. By defining appropriate layers including classes and properties linking the classes, it is possible to represent events properly.

### C. Event Reconstruction using Miscellaneous Approaches

Besides the two aforementioned categories of event reconstruction methods, there is some other miscellaneous methods of event reconstruction.

In [37], a rigorous method for reconstructing events is presented. The idea is that the compromised system is described as a Finite State Machine (FSM). Event reconstruction is performed as follows: 1) A finite state machine of the system is modeled. 2) All possible scenarios of the incident are determined by back-tracing transitions from the state in which the system was discovered. 3) Scenarios that disagree with the available evidence are discarded. While this method try to reconstruct events precisely, it cannot be used in complex real-world systems; Real-world systems usually have thousands (or more) states and transitions, and their state spaces grow exponentially.

To reconstruct events, [38] uses the idea of finite state machine presented in [37]. It represent the system as a deterministic finite automaton (DFA) that encodes the set of system computations as a set of strings. Then it tries to model witness statements. Witness statements are defined as restrictions on strings accepted by the DFA. Therefore, these witness statements restrict the state space of system. Unfortunately, the proposed method in [38] has the same downfall of [37]. Analysis of real systems results in a large state space.

The method in [39] reconstructs user activities using ShellBag information extracted from some registry keys. Some pieces of information about Windows default shell, i.e. Windows Explorer are stored in the registry. Some of users' window viewing preferences such as the window size, the position of the window on the desktop, view mode and sort method for items within the window are stored in the registry. In order to reconstruct user activities, the method in [39] compares registry snapshots. Then using nine detection rules, it detects the presence or absence of user actions between two snapshots.

In [40], a formalized knowledge representation model for digital forensics timeline analysis is proposed. The model reconstruct scenarios from suspect data and analyze them using experts' knowledge and semantic tools. Then a formal incident modelization and operators for timeline reconstruction is presented. This formal representation is designed to answer the challenge of correctness of the whole investigation process.

Ashley Brinson et al. [42] present a cyber forensics ontology for the purpose of specialization, certification, and education. The ontological model consists of a five layer hierarchical structure. The first layer consists of two concepts: technology and profession. At the second layer, technology is broken down into hardware and software, and profession is broken down into law, academia, military, and private sector. Each of these sublayers are broken down into various concepts. This ontology can be augmented to be used for event reconstruction purposes.

Figure 1 summarizes the aforementioned methods on digital evidence collection and analysis.

## V. CONCLUSION AND CHALLENGES

Digital forensics process can be divided into three phases: 1) evidence collection, 2) evidence analysis and event reconstruction, and 3) preparation of court-accepted

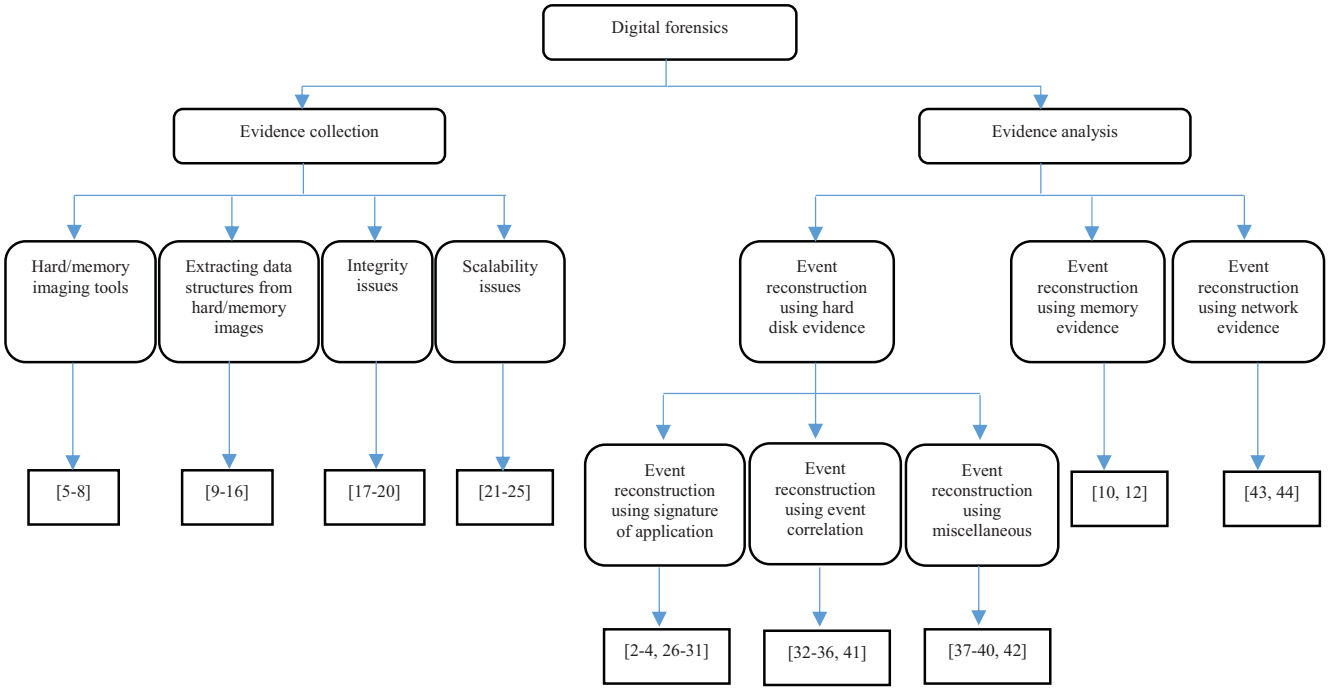


Fig. 1. Digital evidence collection and analysis methods

reports. Each of these phases have some challenges. In the phase of evidence collection, the integrity of collected data is a challenging issue. Hard disk or memory acquisition techniques must provide the integrity of the generated image. Moreover, they must be robust against anti-forensics tricks. Another challenge in the phase of evidence collection is the large volume of digital media. The digital forensics methods should be scalable.

The evidence analysis and reconstruction of events are very challenging too. Various types of low level information can be extracted from different parts of the system which makes the automatic reconstruction of events a challenging process. For example, raw data which are extracted from physical memory, network traffic, various log files, registry, file system metadata and other parts of the system have different formats. Integrating these large-volume multi-format data to generate meaningful events requires many works to be done. In the phase of preparation of reports, the challenge is the generation of reports which are verifiable, valid, and reproducible for the court.

There are several works in the literature which address some of these challenges. This survey tries to review these papers and discuss pros and cons of them.

## REFERENCES


1. Larry Daniel, Lars Daniel. ***Digital Forensics for Legal Professionals: Understanding Digital Evidence From The Warrant To The Courtroom***: Syngress Publishing; 2011.
2. Christopher Hargreaves, Jonathan Patterson. ***An automated timeline reconstruction approach for digital forensic investigations***. Digital Investigation. 2012;9:S69-S79.
3. Sven Kalber, Andreas Dewald, Felix C. Freiling. ***Forensic Application-Fingerprinting based on File System Metadata***. Proceedings of the 2013 Seventh International Conference on IT Security Incident Management and IT Forensics; 2013.
4. M.N.A. Khan, C.R. Chatwin, R.C.D. Young. ***A framework for post-event timeline reconstruction using neural networks***. Digital Investigation. 2007;4:146-57.
5. Guidance Software. Encase Forensic. Available from: https://www.guidancesoftware.com/.
6. AccessData Group. FTK Imager. Available from: http://accessdata.com/.
7. mdd. mdd. Available from: https://sourceforge.net/projects/mdd/.
8. Moonsols. Moonsols. Available from: http://www.moonsols.com/.
9. Richard M. Stevens, Eoghan Casey. ***Extracting Windows command line details from physical memory***. Digital Investigation. 2010;7:57-63.
10. James Okolica, Gilbert L. Peterson. ***Extracting the windows clipboard from physical memory***. Digital Investigation. 2011;8:118-24.
11. Robert Beverly, Simson Garfinkel, Greg Cardwell. ***Forensic carving of network packets and associated data structures***. Digital Investigation. 2011;8:78-89.
12. Andrew White, Bradley Schatz, Ernest Foo. ***Surveying the user space through user allocations***. Digital Investigation. 2012;9.
13. Brian Carrier. The Sleuth Kit 2003. Available from: http://www.sleuthkit.org/index.php.
14. Michael Hale Ligh, Andrew Case, Jamie Levy, AAron Walters. ***The art of memory forensics***: Wiley; 2014.
15. Volatility Foundation. Volatility. Available from: http://www.volatilityfoundation.org/.
16. M.I. Cohen. ***PyFlag – An advanced network forensic framework***. Digital Investigation. 2008;5:112-20.
17. Alessandro Distefano, Gianluigi Me, Francesco Pace. ***Android anti-forensics through a local paradigm***. Digital Investigation. 2010;7:83-94.
18. Slim Rekhis, Noureddine Boudriga. ***A System for Formal Digital Forensic Investigation Aware of Anti-Forensic Attacks***. IEEE Transactions on Information Forensics and Security. 2012;7(2):635-50.
19. Namheun Son, Yunho Lee, Dohyun Kim, Joshua I. James, Sangjin Lee, Kyungho Lee. ***A study of user data integrity during acquisition of Android devices***. Digital Investigation. 2013;10:3-11.
20. Johannes Stüttgen, Michael Cohen. ***Anti-forensic resilient memory acquisition***. Digital Investigation. 2013;10:105-15.
21. Vassil Roussev, Candice Quates. ***Content triage with similarity digests: The M57 case study***. Digital Investigation. 2012;9:60-8.
22. Fabio Marturana, Simone Tacconi. ***A Machine Learning-based Triage methodology for automated categorization of digital media***. Digital Investigation. 2013;10:193-204.
23. Timothy Vidas, Brian Kaplan, Matthew Geiger. ***OpenLV: Empowering investigators and first-responders in the digital forensics process***. Digital Investigation. 2014;11:45-53.
24. Owen Brady, Richard Overill, Jeroen Keppens. ***DESO: Addressing volume and variety in large-scale criminal cases***. Digital Investigation. 2015;15:72-82.
25. Ben Hitchcock, Nhien-An Le-Khac, Mark Scanlon. ***Tiered forensic methodology model for Digital Field Triage by non-digital evidence specialists***. Digital Investigation. 2016;16:75-85.
26. Sven Kalber, Andreas Dewald, Steffen Idler. ***Forensic Zero-Knowledge Event Reconstruction on Filesystem Metadata***. Lecture Notes in Informatics, 2014.
27. S. Garfinkel. ***Automating disk forensic processing with SleuthKit, XML and Python***. In Proceedings of the Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering; 2009.
28. Joshua Isaac James, Pavel Gladyshev, Yuandong Zhu. ***Signature Based Detection of User Events for Post-mortem Forensic Analysis***. Digital Forensics and Cyber Crime. 53 of Lecture Notes of the Institute for Computer

Sciences, Social Informatics and Telecommunications Engineering: Springer Berlin Heidelberg; 2011. p. 96–109.
29. Muhammad Naeem Ahmed Khan. ***Performance analysis of Bayesian networks and neural networks in classification of file system activities***. Computers & Security. 2012;31:391-401.
30. Jens Olsson, Martin Boldt. ***Computer forensic timeline visualization tool***. Digital Investigation. 2009;6:S78–S87.
31. Kristinn Guðjónsson. ***Mastering the Super Timeline with log2timeline***. SANS Institute, InfoSec Reading Room, 2010.
32. Kevin Chen, Andrew Clark, Olivier De Vel, George Mohay. ***ECF – EVENT CORRELATION FOR FORENSICS***. 1st Australian Computer, Network & Information Forensics Conference; Perth, Western Australia2003.
33. Andrew Case, Andrew Cristina, Lodovico Marziale, Golden G. Richard, Vassil Roussev. ***FACE: Automated digital evidence discovery and correlation***. Digital Investigation. 2008;5:65-75.
34. Sriram Raghavan, S V Raghavan. ***AssocGEN: Engine for Analyzing Metadata Based Associations in Digital Evidence***. Eighth International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE)2013. p. 1-8.
35. Bradley Schatz, George Mohay, Andrew Clark. ***Rich Event Representation for Computer Forensics***. Proceedings of the Fifth Asia-Pacific Industrial Engineering and Management Systems Conference (APIEMS 2004); 2004; Gold Coast, Queensland.
36. Yoan Chabot, Aurelie Bertaux, Christophe Nicolle, Tahar Kechadi. ***An ontology-based approach for the reconstruction and analysis of digital incidents timelines***. Digital Investigation. 2015;15:83-100.
37. Pavel Gladyshev, Ahmed Patel. ***Finite state machine approach to digital event reconstruction***. Digital Investigation. 2004;1:130-49.
38. Joshua James, Pavel Gladyshev, Mohd Taufik Abdullah, Yuandong Zhu. ***Analysis of evidence using formal event reconstruction***. Digital Forensics and Cyber Crime2010.
39. Yuandong Zhu, Pavel Gladyshev, Joshua James. ***Using shellbag information to reconstruct user activities***. Digital Investigation. 2009;6:S69-S77.
40. Yoan Chabot, Aurelie Bertaux, Christophe Nicolle, M-Tahar Kechadi. ***A complete formalized knowledge representation model for advanced digital forensics timeline analysis***. Digital Investigation. 2014;11(2):95-105.
41. Benjamin Turnbull, Suneel Randhawa. ***Automated event and social network extraction from digital evidence sources with ontological mapping***. Digital Investigation. 2015;13:94-106.
42. Ashley Brinson, Abigail Robinson, Marcus Rogers. ***A cyber forensics ontology: Creating a new approach to studying cyber forensics***. Digital Investigation. 2006;3:37-43.
43. Xie G, Iliofotou M, Karagiannis T, Faloutsos M, Jin Y. ***ReSurf: reconstructing web-surfing activity from network traffic***. IFIP networking conference2013.
44. David Gugelmann, Fabian Gasser, Bernhard Ager, Vincent Lenders. ***Hviz: HTTP(S) traffic aggregation and visualization for network forensics***. Digital Investigation. 2015;12(1):S1–S11.
45. Justin Grover. ***Android forensics: Automated data collection and reporting from a mobile device***. Digital Investigation. 2013;10:12-20.
46. Josiah Dykstra, Alan T. Sherman. ***Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques***. Digital Investigation. 2012;9:90-8.
47. Josiah Dykstra, Alan T. Sherman. ***Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform***. Digital Investigation. 2013;10:87-95.
48. M.I. Cohen, D. Bilby, G. Caronni. ***Distributed forensics and incident response in the enterprise***. Digital Investigation. 2010;8:101-10.
49. Nick Pringle, Mikhaila Burgess. ***Information assurance in a distributed forensic cluster***. Digital Investigation. 2014;11:36-44.
50. Jonathan Grier. ***Detecting data theft using stochastic forensics***. Digital Investigation. 2011;8:71-7.
51. Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu. ***A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise***. Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining; 1996.