SlackStick: *Signature-Based File Identification for Live Digital Forensics Examinations*

Rob Hegarty
School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Chester Street, Manchester, M1 5GD, UK
R.Hegarty@mmu.ac.uk

John Haggerty
School of Science and Technology, Nottingham Trent University, Clifton Campus, Clifton Lane, Nottingham, NG11 8NS, UK
john.haggerty@ntu.ac.uk

***Abstract*—A digital forensics investigation may involve procedures for both live forensics and for gathering evidence from a device in a forensics laboratory. Due to the focus on capturing volatile data during a live forensics investigation, tools have been developed that are aimed at capturing specific data surrounding state information. However, there may be circumstances whereby non-volatile data analysis, such as the identification of files of interest, is also required. In such an investigation, the ability to use file-wise, or hash, signatures is precluded due to pre-processing requirements by the forensics tools. Therefore, this paper presents *SlackStick*, a novel automated approach run from a USB memory device for the identification of files of interest or non-volatile evidence triage using an alternative signature scheme. Moreover, the approach may be used by inexpert users during a first-response phase of an investigation. The results of the case study presented in this paper demonstrate the applicability of the approach.**

Keywords—Digital forensics, file signatures, live investigations

I. INTRODUCTION

Society's reliance on technology has brought many economic and cultural benefits, but it also harbors many technical and social challenges. One major benefit is our access to data due to information sharing between multitudes of devices. However, these various technologies can lead to a greater access to the means by which illegal activities can be perpetrated. If illegal activities are suspected, a resulting digital forensics investigation may be required to provide evidence of any wrongdoing.

A digital forensics investigation may involve procedures for both live forensics and for gathering evidence from a device in a forensics laboratory. Live forensics takes place at the location at which the computing device is seized prior to the computer being turned off and taken to the forensics laboratory. The aim of such an investigation is to capture volatile data pertinent to the case that may be lost once the computer is powered down, such as existing network connections, open ports, and running software. Persistent or non-volatile data written to the hard drive, such as user files, system logs, and operating system configurations, may be retrieved using forensics tools such as EnCase [1] and the Forensics Toolkit (FTK) [2] even when the computing device is powered down. In this way evidence may be retrieved by an examiner in the forensics laboratory at a later date.

Due to the focus on capturing volatile data during a live forensics investigation, tools have been developed that are aimed at capturing specific data surrounding state information. However, there may be circumstances whereby non-volatile data analysis is also required at this stage. For example, probation services wishing to ensure an offender's compliance with a court order regarding computer usage, or triaging a number of devices owned by a suspect to identify the location of files of interest to the examiner prior to analysis in the forensics laboratory. Forensics techniques for file analysis used in the laboratory cannot be applied in live forensics investigations due to the preparation of the evidence for analysis by the forensics software. For example, the widely used technique of using file hashes as a signature scheme to identify data of interest requires that the evidence is pre-processed, a time consuming procedure that precludes this approach for live forensics. However, such evidence may be pertinent to the examiner at this stage of the investigation and if collected here, may save time to reduce costs and provide focus during the overall investigatory process.

This paper presents *SlackStick*, a novel approach for the identification of files of interest or non-volatile evidence triage during a live forensics investigation. As will be discussed in the next section, the common practice of using file hashes for file identification is not without its issues, often precluding this activity from such investigations. Therefore, rather than use file hashes, the approach posited in this paper uses an alternative method based on identifying features of the underlying file rather than performing an operation on the whole file or data blocks to create the search signature. The automated *SlackStick* software runs from an external device, such as flash memory, that is able to access persistent data stores in read-only mode so as not to tamper with evidence, a key requirement in any forensics investigation. This has the advantage that inexpert people can conduct file searches rather than having to learn complicated forensics applications. Moreover, the proposed scheme allows for multiple signatures for a single file of interest thus reducing the scheme's susceptibility to attack and increased reliability. The case study and results presented in this paper demonstrate the applicability of the approach for live forensics investigations.

This paper is organized as follows. Section 2 presents related work in computer forensics and the use of file

signatures. Section 3 posits the proposed *SlackStick* approach to identify files of interest to a computer forensics examiner. Section 4 presents a case study of using the approach to identify files of interest within test data sets. Finally, we make our conclusions and discuss future work.

II. RELATED WORK

During a digital investigation it is common for forensics examiners to search the evidence for files of interest, i.e. those that are pertinent to the investigation and provide evidence of the activities under analysis. Most commonly, an image of the hard drive is taken to replicate the original evidence source. This ensures that the integrity of the evidence is maintained as all analysis is carried out on the image of the original hard drive. A forensics tool is used to recreate the logical structure of the underlying file system. A computer forensic analyst views the files, both extant and deleted, and files of interest are reported with supporting evidence, such as time of investigation, analyst's name, the logical and actual location of the file, etc. As the investigation of the hard drive relies on the analyst viewing files as if part of the file system, this process is laborious and could be prone to human error.

Therefore, practitioners have attempted to improve the speed of the search within the constraints of the tools at their disposal. A commonly used technique is to compare MD5 and/or SHA-1 file hashes created from the files on the hard drive under investigation to hashes of known files recorded from previous investigations using tools such as EnCase [1] and FTK [2]. These tools reconstruct the underlying file system and create hashes of files to verify the validity of the data analyzed by the examiner. These hashes are used to form the signatures for automated evidence searches. However, three main issues remain when using this approach. First, this scheme can be defeated by changing just one byte in a file as this will result in a new hash value. Second, this approach works well for files on traditional magnetic media but is problematic in emerging environments, such as cloud storage, where file fragments are located across multiple devices and brought together to form the file that a user sees. Finally, the file system has to be recreated to generate the signatures to enable file searches, which is a time consuming process and therefore precludes this technique from live forensics investigations.

Due to these challenges, a number of other schemes have been proposed. For example, file carving is a technique that searches hard drives for files by comparing file headers and footers to a database of known parameters, such as that used by *Scalpel* [3], now a popular forensics file carving tool. Pungila [4] posits a file carving approach for identification of files through data-parallel pattern matching. Alternative schemes focus on file signatures, data patterns of interest embedded in a much larger data file and are therefore applicable to a range of applications, such as network security countermeasures, medicine, and image processing [5]. For example, [6] proposes a file signature scheme for intrusion detection for the identification of malicious PDF files. Solis [7] posits an approach for identifying the existence of sensitive information in a database utilizing individual bits.

The advantage of signatures over hashes has led to a number of approaches to identify files of interest in dynamic environments, such as networks and cloud computing. For example, [8] discusses the potential of metadata tags as a potential approach to tracking original file information in the cloud. Chawathe [9] proposes the use of block-wise rather than file hashes for file fingerprinting whilst [10] posits a scheme to perform file block classification by using Support Vector Machines for the identification of the relative file blocks. Gan and Chen [11] propose a scheme for data verification based on the encryption of file data blocks. da Cruz Nassif and Hruschka [12] propose clustering as an approach to identify similarities between files and retrieve files of interest. Pierris and Vidali [13] propose an approach based on block hash values combined with Hierarchical Self-Organizing Map algorithms in order to classify broken chains of previously unknown files. Schaefer et al [14] and Edmundson and Schaefer [15] propose an approach for the retrieval of multimedia files from Huffman tables located in the header of JPEG files. These schemes have in common two issues that restrict their use in the identification of files in live forensics investigations. First, they require information used by the file system, such as header metadata information, which may not provide a unique, robust signature to form the basis of file analysis. Second, many of the schemes assume complete file or block data to be complete and unchanged, which may not be the case. If a single bit is changed in the original file by a malicious user, the detection scheme may be circumvented.

III. THE *SLACKSTICK* APPROACH

As discussed in the previous section, the common practice of using file-wise signatures (or hashes) as the basis for file identification is not without its limitations, especially for investigations involving live forensics. Therefore, the *SlackStick* approach posited in this paper utilizes the advantages of alternative schemes, in particular, the ability to generate multiple signatures from a single file, not requiring file system information to be present or recreated, and not relying on block-wise information. The way in which this is achieved is discussed below.

The number of electronic devices that a suspect may own or may provide evidence of significance to a digital forensics investigation has grown over recent years. In addition, the range of technology and the available storage memory of electronic devices have driven the scale of such investigations. In the past, a forensics examiner may have only investigated a small number of devices, such as desktops and connected servers; today, an investigation will incorporate these and devices such as mobile phones, laptops, tablets, and Internet access points. This has led to the situation whereby potentially Terabytes of data may have to be searched for evidence relevant to the investigation.

The scale of digital investigations has led to the frequent involvement of both first- and second-response teams during the forensics process. First-response teams comprise those that may be at the scene of an incident, such as system or network administrators, and therefore focus on live forensics. Second-response teams will comprise forensics specialists that may be called to an incident and will work on the available data to

collect evidence both at the scene and in specialist laboratories. The training afforded to first responders is often less than that of digital forensic specialists, prohibiting the application of specialist tools. Furthermore, the time required to analyze Terabyte-scale storage using conventional techniques is becoming untenable.

As device diversity increases there is a demand for operating system agnostic triage tools that take into account the skill sets of first-response teams. This paper therefore posits an approach that may be used to overcome the challenges posed by storage scale, operating system diversity, investigation time, and first responder training. The principle design goals and system requirements for the *SlackStick* approach are as follows:

- **High-speed analysis of large volume data:** As discussed above, the volume of data that may be searched during an investigation may involve Terabytes of files located on available storage media. Therefore, triaging of this data is required to identify data that may provide relevant evidence to the investigation. As noted in [16], triage can extract evidence quickly at the crime scene to provide vital intelligence in time critical investigations.
- **Operating system and resource agnostics:** The diversity of devices encountered means that the approach must be capable of high speed analysis using limited computation and memory resources. In addition, the examiner may not have access to specialist forensics software, resources, or knowledge of the specifics of an operating system. Therefore, the approach must provide an automated means by which a variety of devices and operating systems may be analyzed.
- **Minimal false positives and false negatives:** False positives are undesirable at the triage stage of digital investigations as they will impact on the amount of data that must be analyzed during the second response phase. This will incur resource and cost implications. A high rate of false negatives will ensure that evidence of significance will be missed during the ongoing investigation.
- **Minimal user interaction:** As discussed above, first responders are often not equipped with the knowledge required to ensure the forensic soundness of the triage or the full investigation process. Limiting interaction with the underlying software ensures that the scope for human error is reduced. A simple indication that signatures of files of interest have or have not been detected will suffice for the triage process as their significance will be explored during the second response phase.
- **Partial file detection:** It is often the case that deleted files are only partially overwritten; slack space and blocks marked by the file system as empty often contain fragments of files. It is important that these areas are analyzed to identify deleted or partial fragments of files.

The above goals are achieved by booting a minimal Linux environment from flash memory, such as a USB memory drive attached to the device to be investigated. As will be discussed in the case study, a variety of Linux environments are suitable, such as the Slax [17] operating system (hence *SlackStick*). This environment loads a signature library into memory and enumerates the drives on the device. These drives are then mounted in read-only mode to prevent tampering with potential evidence that may be relied on during subsequent investigatory processes, a requirement of any triage activity. *SlackStick* reads the memory blocks on the target machine sequentially to generate block fingerprints to compare with the signature library. Once the search has taken place, a report is generated for the user to indicate whether and where signatures of files of interest are discovered. A benefit of this approach is that it provides the operating system agnostics discussed above by searching the underlying memory blocks with low memory and system resource requirements compared to forensics tools such as EnCase and FTK which require file system analysis. In this way, search times can be improved versus file-based approaches. Moreover, the automated approach reduces the level of expertise required to perform a signature search of a device as the user merely has to be able to insert *SlackStick* into the computer under investigation.

Figure 1 provides an overview of the *SlackStick* approach. The dashed line represents the boundary between the *SlackStick* application and the external components. As described below, signatures are generated from files of interest to form a signature library that is included in the memory stick mountable operating system. Each block from the evidence source is loaded into memory by *SlackStick* from which a block signature is generated and created to be compared with the signature library. If a match is found the user is informed via the terminal interface.

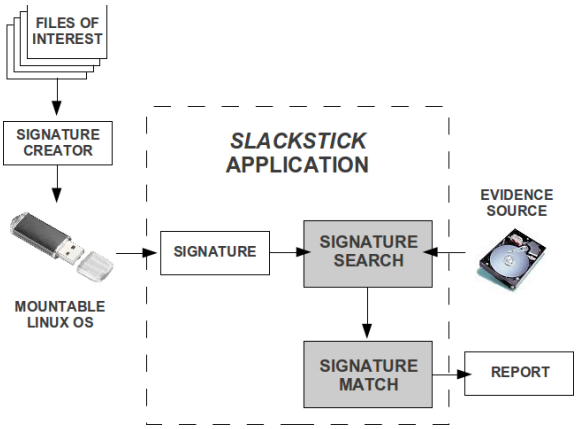

Fig. 1. Overview of the *SlackStick* approach.

The mountable Linux operating system is compiled from a Linux operating system. Slax is suitable as it provides a minimal operating system, approximately 200 MB in size, which is augmented via the addition of modules. This utilization of a minimal operating system reduces resource overheads and complexity, providing only the functionality required for a specific application, in this case *SlackStick*. Python is used to provide the signature search functionality and as such, a Python library is included on the bootable USB

drive. As the mountable Linux operating system is booted, the USB drive loads the Python implementation of *SlackStick*, which is executed with root privileges. This is achieved through the modified *rc.local* script included on the *SlackStick* USB drive. Root privileges are required to gain block level access to the storage device undergoing triage. As discussed above, the storage device is accessed in read-only mode to avoid contamination of potential evidence that may be relied upon later in the forensics investigation.

In order to provide functionality across a wide range of triage activities, the signature library and codebase included on the *SlackStick* USB device can be updated or modified by the investigator using any Windows/Mac/Linux computer. A Signature Creator application on such a computer creates signatures from known files of interest. Ideally, these files will be entropic, i.e. files that have a wide distribution of byte values, such as image or multimedia data, to provide robust signatures. This forms the signature library that is loaded into memory by the Python implementation to provide the comparison with the signatures generated from the blocks on the storage device.

The signatures on which we search will have an impact on the effectiveness of the approach. If we were to search for a specific file, there are three factors that will affect the search. First, digital images may be in the region of Megabytes, thereby requiring a large signature. Second, only one byte in the original data need be altered to give rise to a large number of false negatives, as is the case with current practice of using file hashes. Third, memory blocks allocated to a file are not necessarily sequentially ordered on the underlying storage media. As the data is not sequentially stored, searches using large signatures may be defeated due to fragmentation of the file across memory locations on the storage medium.

To create a signature for the *SlackStick* application, a known file of interest is passed to the Signature Creator where a block within the original file is chosen, which may be from anywhere within a file. With many files being upwards of hundreds of kilobytes in size, the file itself will use many blocks. The first and last blocks of a file are not provided for the signature search. The first block may hold generic but redundant data, such as setting up fonts, and therefore is less robust for analysis. The last block will include slack space data, and therefore cannot provide a reliable signature.

SlackStick accesses the device to search for signatures of files of interest to the investigation. This is conducted a memory block at a time to ensure that the entire storage medium is searched. Each memory block of the storage medium is compared to the signature block(s). If none of these are found within the file, *SlackStick* reports that no signatures are matched, and therefore, no files of interest are resident on the device. If, however, there is a match, a report is generated detailing the signature matched and the physical location of the file in the storage media. The application continues to search the data until the end of the storage medium is reached.

Figure 2 illustrates the digital signature comparison process. The evidence file and signatures are loaded into the application as illustrated in figure 1. A number of points within the signature block are compared to the data in the block. If all points are identical, a signature match is reported.

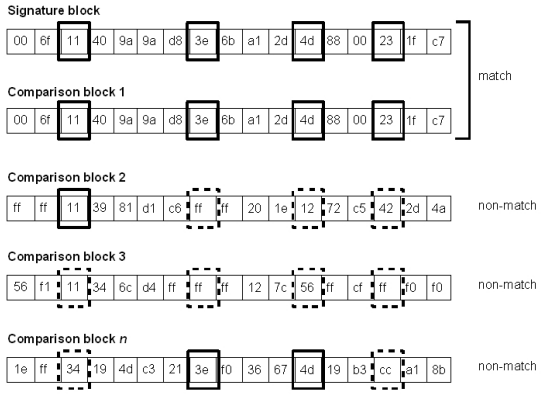

Fig. 2. *SlackStick* signature matching.

Once a signature block and evidence block have been loaded for comparison, various points of reference (i.e. individual bytes at pre-determined positions within the blocks) are compared to see if they are the same. The actual number of points can be chosen depending on the desired balance between the potential for false positives and false negatives; as the number of points of reference increases, so the likelihood of false positive decreases. However, there are also benefits to be had by reducing the level of precision. For example, one of the useful characteristics of the proposed system is that even if some of the evidence data is altered, it may nonetheless still be possible for a signature to be used to discover the existence of a file. Such alteration of data might occur if a malicious person attempts to hide the existence of some data, but without wishing to destroy the file completely, by altering just a few bytes associated with the file. Although this would have the effect of preventing identification using a hash-based signature approach, the flexibility of the *SlackStick* signature process means that it may still be able to detect the file.

Table 1 provides an indication of how the potential for false positives and false negatives will change depending on the size of the signature (i.e. the number of points of reference) used. This data is based on detection in a 4 GB storage medium containing 500 files. The probabilities are calculated based on each byte having a potential value of 1/256. Therefore, using a single byte as a signature and assuming an even spread of byte values across the data set, we would expect the probability for 819248 matches to the signature byte. This value significantly decreases when the probability of specific byte values in specific locations is calculated, as demonstrated in table 1. Since the partition contains 500 files, any value over 500 indicates that false positives have been found. The table and figure highlight how low the probability of false positives is given a signature size of just a few points.

TABLE I. COMPARISON OF SIGNATURE SIZES.

Signature size (no. of bytes)	*Probability of false positive*	*Expected matches*	*Actual matches*
1	3.91×10^{-3}	8192498.04	7769873
2	1.53×10^{-5}	32499.99	30652
3	5.96×10^{-8}	625	608
4	2.33×10^{-10}	500.49	500
5	9.09×10^{-13}	500	500
6	3.55×10^{-15}	500	500
7	1.39×10^{-17}	500	500
8	5.42×10^{-20}	500	500

This section has presented an overview of the novel *SlackStick* scheme for file analysis in live forensics examinations. This approach analyses storage media for evidence of files of interest to the investigation and due to its automated approach can be used by inexpert first responders for forensic triage of devices. The application is able to conduct very fast searches of large volumes of data as it does not rely on reconstructing the underlying logical file structure. The following section provides experiment results to demonstrate the applicability of the proposed approach.

IV. CASE STUDY AND RESULTS

A case study including two experiments was conducted to evaluate the performance, accuracy and feasibility of *SlackStick*. In order to evaluate the feasibility and accuracy of using block based signatures to identify known files of interest, a feasibility study of the accuracy of block based signature detection under real world conditions was carried out. A subsequent experiment was carried out to evaluate the performance and feasibility of a live USB implementation. This type of deployment provides the benefit of portability for triage situations. However, it often introduces a performance reduction. The aim of the experiment was to determine to what extent the performance reduction impacts on the feasibility of live deployment.

In order to compare the performance and accuracy of both experiments a common dataset and target environment was used in both experiments. The target environment was a 1GB FAT32 partition on an internal hard drive, containing 2,095,104 blocks. The drive was sanitized prior to the experiment by securely erasing the existing data on the drive. It should be noted that the approach is file system agnostic. FAT32 was selected as it is compatible with most major operating systems. The standardized dataset stored on the target environment comprised 2194 JPEG images from the British library all of which were between 4 KB and 341 KB in size (totaling 23 MB). As *SlackStick* is a block-based approach, 1 GB of data is undergoing analysis as every block on the device is analyzed.

The signature libraries used in both experiments were also standardized to allow comparisons to be drawn. Signatures were generated from varying numbers of the target files from the standardized dataset. Twenty signature libraries were generated containing between 100 – 2000 signatures at 100 signature increments. Signatures were generated by selecting 11 bytes from fixed offsets within the second block of each of the target files. Signatures are generated from the second block as this allows files containing as few as three blocks to be detected. As discussed above, the first block of a file is not a suitable candidate for the proposed approach as it contains low entropy data (Huffman tables, etc.) and may result in false positives. The final block of a file is also unsuitable due to the presence of slack space that may introduce false negatives based on previously overwritten files. By generating signatures containing 11 bytes from the target files we create a collision resistance signature with 288 (3.0948501e+26) possible permutations, vastly reducing the probability of a false positive occurring during the search.

During each experiment the *SlackStick* script was deployed and executed to analyze the 1 GB partition. Each of the signature libraries was used in turn to analyze the partition. The script generated a signature from each block on the partition and compared it with the signature library being used. Each time a match occurred the signature match counter was incremented and upon completion of each search iteration, the number of matches was recorded. If there were more matches than signatures in the signature library being used, the additional matches were deemed to be false positives. If there were fewer matches than signatures in the signature library being used the difference between the expected number of matches and actual number of matches were deemed to be false negatives (zero false positives or negatives were recorded during the experiment). In addition to recording the number of matches and false positives and negatives the time required to complete the search using the each signature library was recorded by capturing the end and start time of the system clock.

The time required to carry out analysis using the *SlackStick* approach deployed on a native Linux system and as part of a Live USB deployment is illustrated in figure 3.

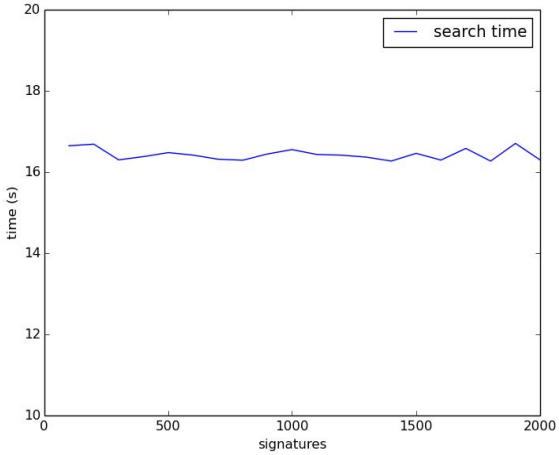

Fig. 3. Search results and comparisons.

The results of the experiment illustrate that the *SlackStick* approach is both performant and highly accurate. As the number of signatures increased no measurable impact on performance was observed. The assumption here is that the application spends a large proportion of time carrying out I/O

with the time spent on comparison with the signature library only taking a small proportion of the execution time. During the experiment zero false positives or negatives occurred. The performance difference (44MBps versus 78MBps) between the live USB (SLAX) implementation and the native implementation exist due to operating system optimization, the native implementation has more finely tuned device support and thus higher I/O than the Slax implementation that is designed to run on any machine, using generic device drivers with the entire OS loaded in to main memory (reducing the memory available for applications).

V. Conclusions and Future Work

A digital forensics investigation may involve procedures for both live forensics and for gathering evidence from a device in a forensics laboratory. Due to the focus on capturing volatile data during a live forensics investigation, tools have been developed that are aimed at capturing specific data surrounding state information. However, there may be circumstances whereby non-volatile data analysis, such as the identification of files of interest, is also required. In addition, this first-response stage of a forensics investigation may involve users with little or no understanding of a digital investigation or have limited forensics tools at their disposal. First responders require tools to triage devices to quickly rule them in or out of an investigation. If too much evidence is gathered the back log of cases is increased; if too little evidence is gathered the investigation may be put in jeopardy.

This paper has therefore presented *SlackStick*, a novel approach for the identification of files of interest or non-volatile evidence triage during a live forensics investigation. As discussed in this paper, file-wise signatures, such as hashes, are not without their issues. The approach posited in this paper uses an alternative method based on identifying features of the underlying file rather than performing an operation on the whole file or data blocks to create the search signature. The automated *SlackStick* software runs from an external device, such as flash memory, that is able to access persistent data stores in read-only mode, so as not to tamper with evidence, and can be used by inexpert users to perform data triage during live forensics investigations. The results of the case study demonstrate that while there is some reduction in performance when carrying out analysis using a live USB device when compared with a native deployment of *SlackStick*. The temporal requirements are reasonable and the accuracy of the approach is preserved.

Further work aims to quantify the relationship between signature length and a variety of other factors to ensure the robustness of the scheme. Encryption will also be introduced to prevent signature tampering by malicious users. Finally, further analysis of the application of a variety of data structures will be carried out to tune the performance of the approach.

References

[1] Guidance Software, "EnCase Forensic v7", https://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx, accessed 1 May 2015.

[2] Access Data, "Forensic Toolkit (FTK)", http://accessdata.com/solutions/digital-forensics/forensic-toolkit-ftk, accessed 1 May 2015.

[3] G.G. Richard III and V. Roussev, V., "Scalpel: A frugal, high performance file carver", Proc. of Digital Forensic Research Workshop, New Orleans, USA, 17-19 August, 2005.

[4] C. Pungila, "Improved file-carving through data-parallel pattern matching for data forensics", Proc. of the 7th IEEE International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, May 24-26, 2012, pp. 197-202.

[5] Z. Ying and T.G. Robertazzi, T.G., "Signature searching in a networked collection of files", IEEE Trans. on Parallel and Distributed Systems, vol. 25 no. 5, pp. 1339-1348, 2014.

[6] A. Bazzi and Y. Onozato, "IDS for detecting malicious non-executable files using dynamic analysis", Proc. of the 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), 25-27 September, 2013, Hiroshima, Japan, pp. 1-3.

[7] J. Solis, "Private searching for sensitive file signatures", Proc. of Security and Cryptography, Seville, Spain, 18-21 July, 2011, pp. 341-344.

[8] R. Crossley, E. Asimakopoulou, S. Sotiriadis and N. Bessis, "A study on metadata tagging for tracking original file information within the cloud", Proc. of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 28-30 October, 2013, pp. 453-456.

[9] S.S. Chawathe, "Fast fingerprinting for file-system forensics", Proc. of the Conference on Technologies for Homeland Security (HST), Waltham, MA, USA, 13-15 November, 2012, pp. 591-596.

[10] L. Sportiello and S. Zanero, "File block classification by Support Vector Machines", Proc. of the 6th International Conference on Availability, Reliability and Security, Vienna, Austria, 22-26 August, 2011, pp. 307-312.

[11] H. Gan and L. Chen, "An efficient data integrity verification and fault-tolerant scheme", Proc. of the 4th International Conference on Communication Systems and Network Technologies, Bhopal, India, 7-9 April, 2014, pp. 1157-1160.

[12] L.F. da Cruz Nassif and E.R. Hruschka, E.R., "Document clustering for forensic analysis: an approach for improving computer inspection", IEEE Trans. on Information Forensics and Security, vol. 8 no. 1, pp. 46-54, January 2013.

[13] G. Pierris and S. Vidali, "Forensically classifying files using HSOM algorithms", Proc. of the 3rd International Conference on Emerging Intelligent Data and Web Technologies, Bucharest, Romania, 19-21 September, 2012, pp. 225-230.

[14] G. Schaefer, D. Edmundson, K. Takada, S. Tsuruta and Y. Sakurai, "Effective and efficient filtering of retrieved images based on JPEG header information", Proc. of the 8th International Conference on Signal Image Technology and Internet Based Systems, Naples, Italy, 25-29 November, 2012, pp. 644-649.

[15] D. Edmundson and G. Schaefer, "Fast mobile image retrieval", Proc. of the International Conference on Multimedia and Expo Workshops, San Jose, CA, USA, 15-19 July, 2013, pp. 1-6.

[16] D. McClelland and F. Marturana, F. , "A digital forensics triage methodology based on feature manipulation techniques", Proc. of the International Confer-ence on Communications Workshops, Sydney, Australia, 10-14 June, 2014, pp. 676-681.

[17] T. Matejicek, "Slax Linux", https://www.slax.org, accessed 1 May 2015.