High Speed Search Using Tarari Content Processor in Digital Forensics

By

Jooyoung Lee, Sungkyong Un, Dowon Hong

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2008 USA

Baltimore, MD (Aug 11th - 13th)

available at www.sciencedirect.com

ScienceDirect

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

High-speed search using Tarari content processor in digital forensics

Jooyoung Lee[*], *Sungkyong Un*, *Dowon Hong*

Information Security Research Division, Electronics and Telecommunications Research Institute, Gajeong-dong, Yuseong-gu, Daejeon, Republic of Korea

Keywords:
Digital forensics
High-speed searching tool
Hardware-based approach
Forensic analysis
Text string search
Tarari content processor

A B S T R A C T

Recently, ‘‘Speed’’ is one of the hot issues in digital forensics. Thanks to a recent advanced technology, today we can get bigger hard drive disks at a lower price than previously. But unfortunately, it means for forensic investigators that they need tremendous time and effort in the sequence of process of creating forensic images, searching into them and analyzing them. In order to solve this problem, some methods have been proposed to improve performance of forensic tools. One of them getting attention is a hardware-based approach. However, such a way is limited in the field of evidence cloning or password cracking while it is rarely used in searching and analysis of the digital evidence. In this paper, we design and implement a high-speed search engine using a Tarari content processor. Furthermore, we show feasibility of our approach by comparing its performance and features to those of a popular forensic tool currently on the market.

1. Introduction

Recently, ‘‘Speed’’ is one of the hot issues in digital forensics. Advances in HDD technologies make the price of HDD to decrease to its lowest level while increasing its capacity to terabytes. Now a 500 GB hard drive is popular costing about \$0.45/GB and you can get 1 TB hard drive with about \$ 400 manufactured by major companies, such as Seagate, Samsung, Hitachi, and so on. Recently, Hitachi Global Storage Technologies announced that they have secured technology implementing areal density of 1 Tb/in^2 and have a plan to commercialize 4 TB HDD for desktop PC by 2011 (Hitachi, 2007; eetasia). Thanks to such a recent advanced technology, today we get bigger hard drive disks at a lower price than previously. As a result, it will not be long before general personal computers are equipped terabytes of hard drive and most enterprise systems have even petabytes of RAID storage.

However, this trend means for forensic investigators that they need tremendous time and effort in the sequence of process of creating forensic images, searching into them and analyzing them. Currently searching tools in digital forensics perform searching at the speed of about 20 MB/s. In this case, it takes for investigators 14 h to search 1 TB of data. When considering general legal process, they do not have enough time to investigate evidences ranging from thousands of person computers to e-mail servers and financial databases of companies. In this point, we can know that ‘‘Size’’ is a serious matter.

In order to overcome this ‘‘Size’’ problem, some methods have been proposed to improve speed in the sequence of each forensic process (Beebe and Clark, 2005; Sommer, 2004). One way in the highlight is a hardware-based approach (Wood, 2007). It has been mainly used in the field of evidence cloning or password cracking aiming acceleration. However, such an approach is limited in the field above mentioned while it is rarely used in searching and analysis of the digital evidence. In this paper, we propose a method using hardware to speed up searching procedure in digital forensics and

[*] *Corresponding author.*
E-mail addresses: joolee@etri.re.kr (J. Lee), skun@etri.re.kr (S. Un), dwhong@etri.re.kr (D. Hong).

1742-2876/\$ – see front matter © 2008 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2008.05.006

design a high-speed search engine with a Tarari content processor. Finally in order to discuss feasibility of our hardware-based approach, we compare its performance and advantages to those of a popular forensic tool on the market – EnCase.

2. Related works

Currently, there have been commercialized hardware forensic tools. Such tools are manufactured and sold in the form of write blockers, disk copiers, data recoverers, external storage systems, forensic systems integrated with various forensic took kits, and field kits such as laptops, PDA seizure toolkits and mobile phone seizure toolkits (http://www.digitalintelligence.com/forensichardware.php).

Some of the most popular tools currently on the market and their features are listed in Table 1. HardCopy II and Shadow 2 of Voom technology are hardware-based imaging tools which are connected with a SATA adapter to a hard disk and they embed a write blocker. HardCopy II makes a copy of an ATA drive at speeds up to 5.5 GB/min (Voom technology). Tableau says that working in conjunction with PRTK (Password Recovery Toolkit) software from AccessData, its TACC1441 delivers accelerated password attacks for algorithms of WinRar, PGP and Winzip by a factor of 6–30 times compared to the unaccelerated processors (Tableau). OmniClone and Sonix of AnyTech LLC make a clone of a SATA drive at peek rate of 3.5 GB/min (Logicube). Besides, several kinds of hardware tools have been developed though they aim for creating a forensic image or accelerating password recovery.

For forensic search tools, it is one of the important features to present all the matching results without missing when an investigator gives a query. To meet this requirement, the forensic searching needs more time than traditional searching because it has to perform bitwise operations on the whole disk in the physical level. However, it is known that there are few products for accelerating forensic searching and analysis. Roussev and Richard (2004) proposed a design based on distributed processing and an open protocol. Their results show that the approach using multiple machines performs faster than tool of FTK. But, their approach is little practical although it introduces possibilities to use distributed processing in digital forensics.

To accelerate search time, an index-based search method is provided by commercial forensic tools but it also takes long time to construct an index database before performing the search though it returns a result in a short time. Such points make stream-based disk forensics as a hard problem (http://www.forensicswiki.org/index.php?title=Open_Research_Topics). But we are assured that the hardware-based searching presented in this paper is an attractive approach to overcome this problem.

3. Programming model

In this section , we determine a programming model to utilize a Tarari board most effectively and design a search engine. The Tarari board allows a user to develop applications that exploit Tarari RegEx agent which provides an arbitrary content identification and characterization (Tarari, 2006). It enables applications to analyze fixed or variable patterns in the streams or volumes at speeds up to 1 Gb/s. Therefore, it is mainly used in the field of Intrusion detection and prevention, Anti-SPAM, Content filtering, MIME parsing, XML parsing, Anti-virus, Real-time message routing, Protocol emulation/

Table 1 – Hardware forensic tools on the market and their main features

	Forensic tool	Manufacture	Feature
	HardCopyII Shadow 2	Voom technology	H/W based imaging tool with writing protect
	Instant Recall		Second generation instant recovery tool
	TACC1441	Tableau	Accelerating password recovery
	T35e		Write blocker
	OmniClone Sonix	AnyTech LLC	Hard drive duplication system

modeling, and so on. For all that, sophisticated design for each application is necessary for getting best performance of the Tarari board.

In this paper, we apply it to performing a digital forensic search with a bitstream of evidence image as a Tarari bitstream and with query keywords as Tarari regular expressions. The searching process summarizes initialization, compilation and loading of a regular expression, and scan. In the initialization step, a content processor initializes internal variables and communication parameters between an application and Tarari RegEx agents and tunes values for optimized performance. We'll tune parameters enabling multiple agents in this step.

By decoupling compilation of regular expressions from scanning, Tarari content processor allows the application to work with very large set of regular expressions which can take several minutes to compile while avoiding having to compile them every time they are used. This feature is useful for the forensic searching. For example, we can write regular expression sets for mobile phone numbers in Korea as 01 [0–9] *– *[0–9] [0–9] [0–9] [0–9]* *– *[0–9] [0–9] [0–9] [0–9] and for personal registration numbers as [0–9] [0–9][0–1] [0–9][0–3] [0–9] *– *[1–4] [0–9] [0–9] [0–9] [0–9] [0–9] [0–9]. They have several asterisks and it means that it may take more time to compile those expression sets to binary images due to their complexity. So in order to save time, we're able to precompile those expressions and load them to the agents before scanning.

When loading binary images to the agents, Tarari provides two methods. One is an automatic load balancing model controlled by the content processor and the other is an agent addressing model. In the agent addressing model, it may be expected improved performance by implementing a priority load balancing scheme since the application can load each agent individually with the same or different sets of expressions.

After scanning process, we can get an output which is a binary file containing only the numerical values in a hexadecimal format. The values are the starting pointer and end pointer of a pattern, indicating relative addresses to the beginning of an input stream. Tarari board supports a single threaded model and a multi-threaded model for the scanning process. It give us another room for speeding up this process since the scan operations are overlapped with job I/O and the Tarari agents can be kept fully utilized by spawning multiple threads.

Using Tarari board, we take several tuning factors into consideration. First of all, we select a programming model considering by load balancing methods and the number of threads. Besides, we take account of the number of agents, size of input data to be processed per a round, a point of time to read in the data and so on. In this paper, we design three algorithms for high-speed searching and choose a best case through some tests. Fig. 1 shows three pseudo algorithms we designed.

All algorithms have a similar flow. First step is to initialize the Tarari RegEx agent and tuning parameter values. In this step, we set the parameter to support the agent addressing mode to control the agents directly. Next, a file which contains keywords or regular expressions is read in and compiled into a Tarari image. The keywords and regular expressions in the file are what the investigators want to search for and are obtained from users through a graphic user interface module. The compiled image is loaded on the agents. Scanning sequences by Tarari agents begin with thread initialization to obtain a thread handle that is needed by the scanning functions. To prepare a document for scanning, an evidence image is read into buffers. Scanning function submits the job to the Tarari RegEx agent hardware, and returns a unique job ID to be used later as a handle to reference the job and its results. To get results, the application iteratively retrieves the document state for each completed job, then accesses the scan

(a) Case 1

```
Initialize
getHWConfiguration
read in rexFile
compileAndSave
loadImageToAgent
initializeThread
for # of agent per board {
    read in dataFile
    scanNonBlock
}
do {
    if (jobCompleted) {
        if (available agent) {
            scanNonBlockToAgent
        }
    }
    else {
        read in DataFile to buf
    }
} while(there is activeJob)
for total job finished {
    getResults
    printResults
    freeJob
}
freeJobList
deinitializeThread
shutdown
```

(b) Case 2

```
Initialize
getHWConfiguration
read in rexFile
compileAndSave
loadImageToAgent
initializeThread
for # of agent per board {
    read in dataFile
    scanNonBlock
}
do {
    if (jobCompleted) {
        if (available agent) {
            scanNonBlockToAgent
        }
        getResults
        printResults
        freeJob
    }
    else {
        read in DataFile to buf
    }
} while(there is activeJob)
freeJobList
deinitializeThread
Shutdown
```

(c) Case 3

```
Initialize
getHWConfiguration
read in rexFile
compileAndSave
loadImageToAgent
initializeThread
for total job {
    read in dataFile
    scanNonBlock
}
while(!JobListCompleted) {
    If (jobCompleted) {
        getResults
        printResults
        freeJob
    }
}
freeJobList
deinitializeThread
shutdown
```

Fig. 1 – Three different pseudo algorithms proposed for the search using Tarari RegEx agents.

results by calling a *completescan* function with the job ID. After that, the application releases resources consumed by the job.

The main difference among three algorithms is a point of time to read documents into buffers and to get scanning results. In the first algorithm, an application executes file I/O while polling if the agents finished their works. With similarity to the first algorithm, the second algorithm has difference in a postprocessing routine which gets results and prints them. Therefore, it can omit a "*for*" statement routine since the postprocessing routine comes into a "do-while" statement. In the final algorithm, the application reads in all data to buffers and submits total jobs to the agents. After that, the application executes the postprocess when the agents complete their jobs. In Section 4, we make a test on performance of three algorithms and implement a search engine with the best case. And then, we discuss feasibility of our proposal by presenting a performance measurement of the search engine.

4. Performance measurement

The first experiment in this section is to measure the performance of three algorithms presented in Section 3. Platform requirements are listed in Table 2. In this test, we use a Tarari 3113 model on Linux Fedora Core 6.

This test measures how long it takes for each algorithm to search a pattern stored in the evidence of 150 MB. As changing the number of agents, we take a measure of time. In this test, we use a Korean term "홍길동" as the pattern. The experimental results are shown in Table 3 and Fig. 2. These are the average values of 100 trials. Among three algorithms, the third case with 4 agents has best performance, 138 MB/s and the first algorithm with 1 agent is worst case, 22 MB/s.

When increasing the number of agents from 1 to 2, we make improvements of 35% in the case 1. But we find that it takes more time of 2–4 ms for the search using more than 2 agents though it has little practical meaning. In case 1 and 2, whenever increasing the number of agent, we make improvements of 80, 50, and 20%, respectively.

As a result of this test, we implement a search engine using the case 3 algorithm with 4 agents. The search engine is integrated with a graphical user interface on Windows system for convenience in forensic investigation. When an investigator queries a keyword, the GUI module forwards it to the search engine on Linux over TCP/IP and then, the search engine executes the search process and returns results to the GUI module.

We install the GUI module on a system which has Microsoft Windows XP Professional Service Pack 2(SP2) OS and Intel Core™2, 2.4 GHz, 3 GB DDR2 DRAM. Also, for comparison of performance, EnCase Forensic Version 6 which is one of the popular forensic tools is installed on the same system. The second experiment is to find all occurrences of the simple phrase "홍길동" in Korean for a single keyword, "forensics", and "암호기술연구팀" in Korean for multiple keywords, and a more complicated regular expression:

$$[0-9][0-9][0-9][0-9][0-9][0-9]^{*}-^{*}[0-9][0-9][0-9][0-9][0-9][0-9][0-9]$$

Table 2 – System setup for experiment

Platform	Description
CPU	Intel Xeon 5149 2.33 GHz
Memory	1 GB DDR2 667 MHz ECC
Disk	500 GB 7.2K rpm SATA
Interface	PCI-X slot
Pattern matching board	Tarari Grand Prix 3113
OS	Linux Fedora Core 6
Compiler	gcc

which matches all strings containing Korean resident registration number, with six digit sequences and seven digit sequences which have a dash (–) between them. We use a 1 GB forensic image which is made with *dd* command of Linux. The test results are shown in Table 4.

The number in the parenthesis indicates the hit number of keywords. In case of multiple keywords and regular expression, EnCase finds fewer patterns than we do. We assume that it is caused by the fact that EnCase could not extract texts in a structured format by a domestic word processor, Hangul. The structure of Hangul document is so unique that we use an individual approach to handle Hangul document. We find all Hangul documents in a target evidence image and use a special filter to extract plain texts from the documents before scanning. We achieve 100.84 MB/s for a singles keyword using the hardware-based search engine and it is 5 times faster than the speed of a forensic tool on the market. Especially, the search based on Hardware has similar performance for a regular expression with the case of single keyword because it does not require additional time to process the regular expression.

As to digital forensics, it is one of the important requisites of the search tool to present all matched results from the given query keyword. The bitstream search using the Tarari board enables the investigator to find patterns in deleted files and hidden files, for example data in an alternative data stream in NTFS. Considering these kinds of advantages, we assure that our proposed method is very useful in forensic analysis of a massive volume of data.

5. Conclusions and future works

Requirement for high tech tools against high tech crimes has been increasing steadily. In order to meet such a requirement,

Table 3 – Experimental measurements of three cases

Agents	Case1	Case2	Case3
1	6.898	6.643	1.460
2	3.814	3.570	1.080
3	2.602	2.368	1.084
4	2.158	1.929	1.086

These are the average search time (s) for a string in 150 MB stream data.

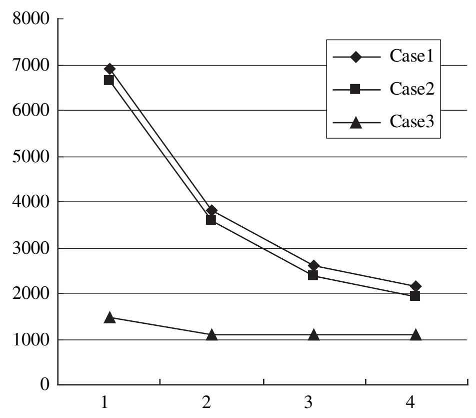

Fig. 2 – Performance graph of three algorithm cases. The case 3 shows the best performance and the case 1 and 2 show similar shapes.

a lot of forensic tools based on hardware have been developed and commercialized. But unfortunately, applying this approach to the field of the forensic search and analysis is rarely to be found. In this paper, we have proposed a forensic searching method using hardware as a solution to those trends and requirements. Additionally, as making a measurement of the performance and comparing it to that of a commercialized forensic tool, we proved the feasibility of this approach. Our results showed that search using a Tarari board can be performed over 5 times faster than tools currently on the market. We had same results with even a set of regular expression. Furthermore, it is a practical approach and we have used it as a search component of HSFS (High Speed Forensic System) which has been developed to accelerate speed in digital forensics investigation.

However, the bitstream-based disk forensic including our method brings about misanalysis or over-analysis since it searches whole disks including not only data area for files, but also system area and unallocated area. It is possible for a general purpose search engine to increase precision ratio while lowering recall rate but forensic investigators want all matched results. Therefore, for forensic analysis, it would be better to present relatively fittest information to the investigator's intention in the front parts of the result list. It is expected to minimize the time for the investigator to perform filtering unnecessary data and it can contribute to improve the task efficiency. Beebe and Clark (2007) and Lee (2008) previously have shown fundamental research results on this issue.

Table 4 – Test results for a single keyword, multiple keywords, and a regular expression

	Single keyword	Multiple keywords	Regular expression
Proposed	100.84 (18)	97.03 (823)	102.58 (70)
EnCase	20.14 (18)	17.41 (711)	17.12 (0)

These show that the speed (MB/s (Hit)) of proposed method is faster over 5 times than that of EnCase.

In order to solve this problem, we have a plan to study on effective presentation methods and integrate them into our search engine. But measuring the satisfaction degree of the investigators is considerably difficult since it is very subjective. Therefore, the continuous research on an evaluation method for effectiveness and satisfaction is also planning to be proceeded.

REFERENCES

Beebe N, Clark J. Dealing with terabyte data sets in digital investigations. In: IFIP international conference on digital forensics. Advances in digital forensics; Feb 13–16, 2005.

Beebe NL, Clark JG. Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. Digital Investigation 2007;4S:S49–54.

eetasia. HDD makers explore ways to up areal density. Available from: http://www.eetasia.com/ART_8800470011_499486_NT_702e0281.HTM.

Hitachi. Available from: http://www.hitachi.com/New/cnews/071015a.html; 2007.

Lee Jooyoung. Proposal for the efficient searching and presentation of search result in digital forensics. In: Proceedings of workshop on digital forensics; Mar, 2008.

Logicube. Available from: http://www.logicube.com/products/hd_duplication/omniclone10xi.asp.

Roussev V, Richard G III. Breaking the performance wall: the cases for distributed digital forensics. In: Proceedings of the digital forensics research workshop; 2004. p. 1–16.

Sommer P. The challenges of large computer evidence cases. Digital Investigation 2004;1:16–7.

Tableau. Available from: http://www.tableau.com/index.php?pageid=products&;model=TACC1441.

Tarari. Regular expression agent content processor API guide; 2006.

Voom technology. Available from: http://www.voomtech.com/hc2.html.

Wood S. Computer forensics: high-tech tools for a high-tech problem. In: Conference on IT-incident management & IT-forensics; Sep 11–13, 2007.

Jooyoung Lee is a Senior Member of Engineering Staff in Electronics and Telecommunications Research Institute. She received M.Sc. degree in computer science from Yonsei University, Korea. She joined ETRI in 2000 and participated in a secure Web Service platform development project and an RFID security technology development project. Her research interests are forensic analysis, artificial intelligence, and data mining.

SungKyong Un is a Principal Member of Engineering Staff in Electronics and Telecommunications Research Institute. He holds M.Sc. degree in computer science from Pohang Institute of Science and Technology, Korea. He joined ETRI since 1993 and participated in Satellite Communication, Operating System Security, Conditional Access System and Security Router development projects.

Dowon Hong received the BS, MS, and PhD degree in mathematics from Korea University in 1994, 1996, and 2000. He joined ETRI in 2000 and has worked as a senior engineer in the Cryptography Research Team. His research interests are in digital forensics analysis, computer security and cryptography.