International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

# Weight-Based-Approach for Searching File Using File Attributes in Forensic

MAHENDRA SHIVAJI PANDIT , IRFAN SIDDAVATAM

K.J. SOMAIYA COLLEGE OF ENGINEERING, MUMBAI, EMAIL ID - MECOMP2012@GMAIL. COM
K.J. Somaiya College of engineering, Mumbai, email id - irfansiddavatam@somaiya.edu

**Abstract**— The main goal of digital forensics is the extraction of suspected files from the target devices that can be defined as digital evidence. The digital world is developing at a very fast pace. The size of hard disks made available to the users is also increasing rapidly. The volume of data or the number of files that can be stored is also increasing. To find a particular file the investigators rely on filters and traces. The traceability process has become a key or an important element of the digital investigation process, as it is capable to map the events of an incident from difference sources in obtaining evidence of an incident to be used for other auxiliary investigation aspects. Filter are used by the investigator to remove unwanted data and get the required files. But the traceability and filter have not been explored to its limits. Because of these, little manipulation in the data on the digital device makes the use of specific trace or filter useless. These work like a loophole, which can be used by criminals to divert the investigation away from evidence. The loophole can be made less harmful by creating a priority based investigation using traces and filters. Priority is given to files which may be or may not be evidence, by assigning them the weight on the basis of the results of traces and filters. By assigning the weight, all files will be taken into consideration and files can be arranged based on the weight. So with weight-based-priority in use, the little or more manipulation to data will be taken into consideration.

**Keywords**— Digital Forensic, traceability, filters, weight , priority, data analysis, file search, attributes.

## INTRODUCTION

Computers and other digital devices are becoming ubiquitous in our modern society. It was inevitable that they would begin to feature as heavily in crime and law. Since the late 1970s the amount of crime involving computers has been growing very quickly, creating a need for constantly developing forensic tools and practices. Almost 99 percent of criminals leave evidence which could be captured and analyzed through proper computer forensic procedure. At one end as the technology is getting advanced, the space of the digital devices are increasing rapidly. The global data supply reached 2.8 zettabytes (ZB) in 2012 - or 2.8 trillion GB - but just 0.5% of this is used for analysis, according to the Digital Universe Study. Volumes of data are projected to reach 40ZB by 2020, or 5,247 GB per person, with emerging economies accounting for an increasingly large proportion of the world's total[1]. Thus the data to be analyzed becomes huge and a challenge for the forensic investigator to perform the forensic investigation in time. One of the important factor in analyzing the data is traceability.

Whenever any operation is done on a device, it makes a lot of entries for security and auditing purpose. These entries are often called as traces. With the help of these traces, it will be known what should be traced and what location and what data. This process is known as traceability. Traceability is the means to identify and follow real or imaginary objects through a process chain. It gives the opportunity to back-track a chain of events, or to predict process outcomes given in the origin of an object. In digital forensic investigation process, tracing is described as a process of finding or discovering the origin or cause of certain scenario. The tracing activities are able to discover the traces left in digital devices. In the computer crime perspective, trace can be found in any digital devices. These traces consist of activities such as login and logout of the system, visit of pages, accesses documents, create items and affiliation groups found in records of data [2].

## BACKGROUND

Digital forensic have solved many crimes committed with the help of computers where evidence may reside on a computer. From its start in 1970 till today, the field of digital forensic have came a long way and have made many developments. Digital forensic started in early 1970. At that time forensic techniques were developed primarily for data recovery. By the late 1980s utilities were being widely advertised that could perform a variety of data recovering, including "Unformat, Undelete, Diagnose & Remedy. In these early days forensics was largely performed by computer professionals who worked with law enforcement. The years from 1999

to 2007 were a kind of "Golden Age" for digital forensics. During this time digital forensics became a kind of magic window that could see into the past (through the recovery of residual data that was thought to have been deleted) and into the criminal mind (through the recovery of email and instant messages). The Golden Age was also marked by a rapid growth in digital forensics research and professionalization. Universities around the world started offering courses in digital forensic[3]. On the other hand many companies developed software's specialized for forensic investigation. Open source platform also contributed towards the development of digital forensic.

*A. Size of digital devices - A problem*

Today much of the last decade's progress is quickly becoming irrelevant. Digital Forensics is facing a crisis. Hard-won capabilities are in jeopardy of being diminished or even lost as the result of advances and fundamental changes in the computer industry:[4]

- The growing size of storage devices means that there is frequently insufficient time to create a forensic image of a subject device, or to process all of the data once it is found.
- The increasing prevalence of embedded flash storage and the proliferation of hardware interfaces means that storage devices can no longer be readily removed or imaged.
- The proliferation of operating systems and file formats is dramatically increasing the requirements and complexity of data exploitation tools and the cost of tool development.
- Whereas cases were previously limited to the analysis of a single device, increasingly cases require the analysis of multiple devices followed by the correlation of the found evidence.

The vast size of today's storage devices means that time honored and court-approved techniques for conducting investigations are becoming slower and more expensive. External hard disks of any size starting from 1tb are easily available in market at reasonable prices. This rapid increase in size is becoming a challenge for forensic investigators.

*B. Working of search filter*

Filters have always been of great help to the investigators in getting the particular file from many files on device. The filter works in very simple way. The following figure explains the use of filter.

In the diagram, the rectangle represents the digital device space and the circle represents the result of filter. It can be clearly seen how some specified data is separated from all the data using the filter. More than one filter can be used to get more specific files[5].


Figure - Use of filter

The figure below, explains the working of use of more than one filter. In the Diagram, 3 filters are being used. Lets represent the result of filter 1 as 'A', result of filter 2 as 'B' and result of filter 3 as 'C'. Let the final output of all 3 filters be 'O'. So 'O' can be written as-

$$O = A \cap \mathbf{B} \cap C$$

The black color represents the final output.


Figure - Use of more than one filter(3)

## TRACES AND FILTERS

Most important factor in retrieving the evidence are the traces or filters. So traces and filters play very important role in getting the evidence or the hint for evidence

A trace is any entry that the operating system makes on the device when a certain operation is executed. The operating system maintains many such traces when working. Some of the trace points are listed below.

1. Recent files
2. Prefetch files
3. Jumplist
4. Lnk files
5. Event log
6. System log
7. MFT
8. Memory dump
9. Registry
10. Previous version

Filters are the nothing but the user specified conditions. The filters can be the metadata or contents of the file. The investigator uses the filters to filter out the excess data and get the data that may be the evidence. For example - Type Filter, if we give .pdf as type filter then outcome will be only .pdf file and all other file types will be not considered.

## EXPLORING FILTERS

With the current working of filters, result only contains the files that satisfy all filters. If any changes is made to the data, then the files affected by the change may or may not be the be the part of the results. In such case, the evidence itself may not be taken into consideration.

Implementing weight-based-priority can handle such changes effectively. Let's see how the weight-based-priority helps. Let's use simple weight system given below -

Weight(File) = No_Of_Filters_Satisfied(File)

According to above equation, weight of a file will be equal to number of filters satisfied by the file. So, files in red colored area will have weight 3, files in blue colored area will have weight 2, files in orange colored area will have weight 1 and files in white colored area will have weight 0. Files can be then giving priority based on this weights. There are two cases :

Some investigator would like to have the traditional way of working with filters. For them , the files of importance will be the files which satisfy the condition below -

Weight(File) = No_Of_Filters_Satisfied(File)

= Total_No_Of_Filters

This files would have the highest weight and based on priority will be placed at the top of all files. So the traditional way of working with filters have changed but still the investigator can get the result of filters as if it were working in traditional way.

Other files with less weight will follow high priority files. The question is why this files are important. consider files whose weight will be equal to -

Weight(File) = Total_No_Of_Filters - 1

Example - Name, Author, Type, CreationTime, AccessTime, ModifyTime are the filters used by investigator for analysis. Suppose the suspect have renamed the file . So with traditional way of filters, all the filters will not be satisfied and the file will not be shown in results. With weight based priority, the file will be placed immediately following the top priority files(if any). The file which was changed by the suspect is also shown in results. Similarly if suspect changes the extension of file, the file will still be shown in results based on its priority.

Taking into consideration traditional search, the result of applying filters would be -

$O = A \cap B \cap C$

If any changes are made to the data, then the evidence may or may not be shown in the results. This would make things more difficult.

While on the other hand, the weight-based- priority approach will take whole space into consideration. The files can be arranged by weight to get most important files on top.

$O = A \cup B \cup C \cup U$

[Note: U represents whole device space.]

## EXPLORING TRACES FOR SEARCHING

A trace is nothing but a entry of certain operation being executed on the OS. The trace when followed leads to certain file on the digital device. There are two possibilities - either the file may be present or file may not be present there. In current scenario, the trace that leads to a file that is not present in its location is ignored.

There are many possibilities of why the file may not be present at its location. For example - file may have been deleted, file may have been renamed, file may have been moved to some other location and so on. Instead of Ignoring such traces, it is possible to use the result of the traces in further exploring the devices. The next important point to consider is , they are many types of traces written on the devices. Each trace leads to certain file or location. Each type of trace is evaluated independently of other type of trace. Instead of evaluating each trace independently, the results of one trace can be used further with the results of other traces. This process if continued will build up a relationship among the traces. This in turn will lead to pointing out to the evidence with higher possibilities. The following figure shows how the traces can be used in exploring the data:

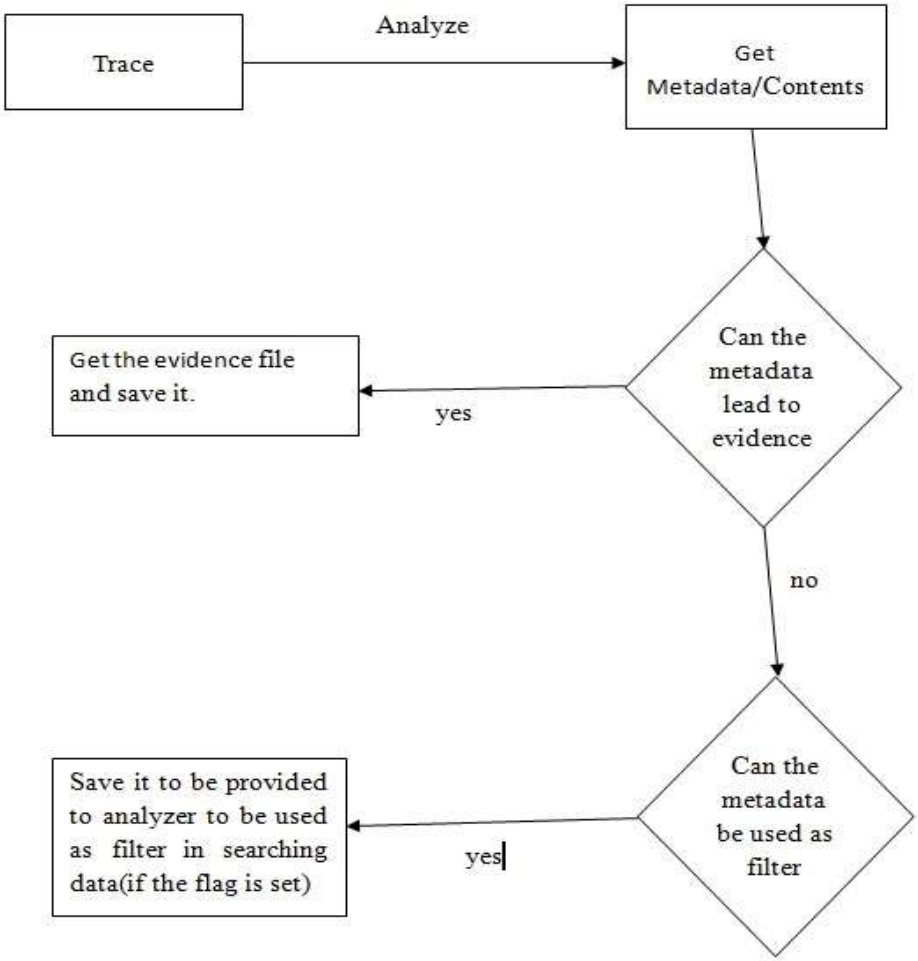


Figure - Use of traces to get metadata

**Steps:**

1. Get the trace.
2. Analyze the trace.
3. If it's associated file/location is present save the file as evidence.
4. If the file is not present, the metadata recovered from the trace will be provided to search filter.
5. Are there any more traces left. If yes, get the next trace and go to step 2.
6. Follow this process until all traces have been analyzed.

## IMPLEMENTATION DETAILS

*A. System details*

- System Type - 64 bits
- OS - Windows 7 Ultimate (Service pack 1)
- Hard disk size - 500 GB
- Ram - 6 GB
- Processor - Intel(R) Core(TM) i3-2348M @ 2.30 GHz 2.30 GHz

*B. File System Details*

- File System type - NTFS
- Number of drives - 7

Table - File System Information

| Volume | Letter | Total size | Files | Dir |
|---|---|---|---|---|
| 1 | H: | 97.3 | 15755 | 2252 |
| 2 | I: | 48.8 | 21713 | 4764 |
| 3 | C: | 48.8GB | 182993 | 81872 |
| 4 | D: | 71.6 | 19952 | 2681 |
| 5 | E: | 25.9 | 17855 | 18066 |
| 6 | F: | 73.9 | 62033 | 20996 |
| 7 | G: | 98.8 | 7097 | 2694 |

*C. Program implementation*

3 Modules were written to implement the Weight-Based-Approach for file search. One third party software was also used to assist in collecting the information. The software's used is - Log Parser 2.2 provide by Microsoft. The three module programmed for the implementation of Weight-Based-Approach for file search are -

1. FS_collector.bat - This module is designed to collect all file system information on hard disk. This module uses the Log Parser 2.2 tool to collect information.
2. Analyze.py - This module is designed to analyze the information collected by FS_collector.bat . This module is a command line script. It provides different options to take input from user.

Table - Options for analyze module

| | | |
|---|---|---|
| -h | | Show this help message and exit |
| -f | FILE_NAME | File name to be searched |
| -e | EXTENSION | The Extension of file to be searched |
| -m | MODIFICATION_TIME | Provide the range of time of file modification |
| -a | ACCESS_TIME | Provide the range of time of file access time |
| -c | CREATION_TIME | Provide the Range of time of file creation (from-to) |
| -s | FILE_SIZE | Size of file |
| -p | FILE_PATH | Path of file |
| -w | MINIMUM_WEIGHT | Minimum weight of file to be satisfied |
| -t | | Use the traditional way of search filter |

3. Get_file.py - This module is used to get the results or search within the results created by Analyze.py module.

The 3 module were run on the system(details are provide above) and the results were as follows. First the FS_collector was run. The FS_collector gathered file name, Size, Path, creation time, last access time, modification time of all the files present on the disk. This module created 7 files corresponding to 7 drives. Each file contained all the files information and its attributes contained in that drive. Secondly the Analyze.py was executed. If analyze.py is executed without any arguments then it will analyze all the seven files created by the FS_collector and give weight zero to all files and store the result in on file. This scenario is never used. It is used only when its needed to check the maximum amount of time that the program will take to analyze the files. The graph below represents the amount of time needed to collect and analyze the files.

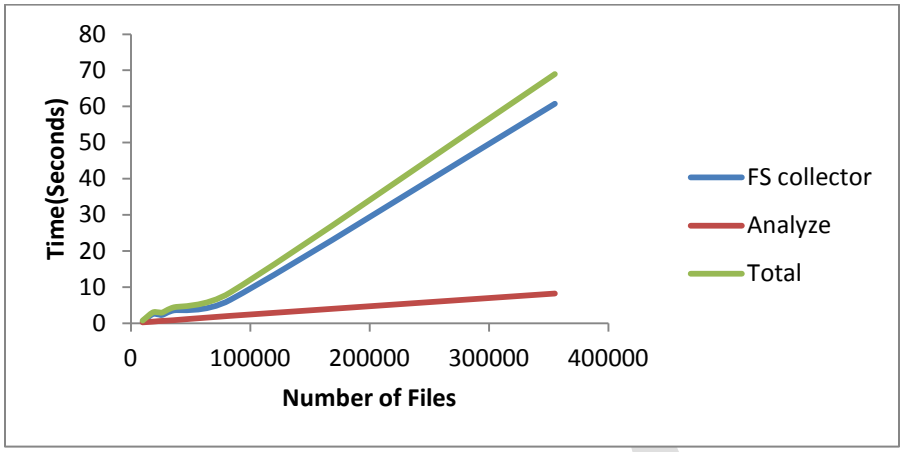


Figure - Time required to collect and analyze

The main module to understand weight-based-system is Analyze module. This module actually analyzes the data collected by FS_collector to perform weight-based searching. This searching can be done in 3 ways.

1. The Traditional way - In this case, the files are searched in traditional way i.e. the result of search will include only those file that satisfies the filters given. That means -

weight(file) = Number of Filters.

2. Weight-Based-Approach where weight is not given - In this case analysis to find a file is done on the basis weight-based-approach but weight is not assigned. So automatically weight is assigned a value of zero. The result will show all the files and their associated weight. That means -

weight(file) = Number of Filters Satisfied >= 0

3. Weight-Based-Approach where weight is given(W) - In this case, weight is given and it is some positive integer. The file are analyzed and only those results are stored where weight of file is greater than the weight given(W). That means -

weight(file) = Number of Filters satisfied >= W

Few commands were run randomly to check the results. In all cases the files to be searched were found(that existed on disk). The only difference was seen in the execution time taken by them. The graph below represents the time taken by them .

Table - Time taken when different number of filters are provided

| No. of filters | Time -t | Time Weight Based |
|---|---|---|
| **1** | 8.9466 | 14.4066 |
| **2** | 9.2586 | 15.6936 |
| **3** | 9.6408 | 14.235 |
| **4** | 10.1556 | 15.3406 |
| **5** | 10.8186 | 15.7638 |
| **6** | 11.817 | 19.4064 |
| **7** | 12.1602 | 31.8162 |

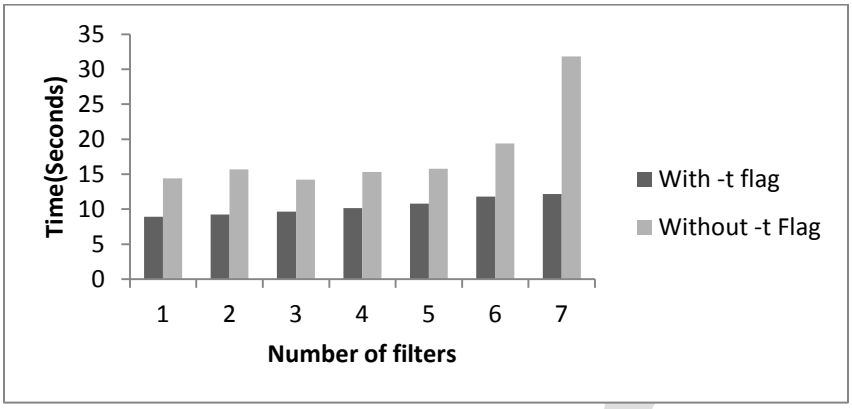


Figure - Time taken by different number of filter

With -t flag means traditional way and without -t flat means weight-based-approach where weight was not provided.

*D. How weight based approach is different*

In traditional search, if the file is not found then it becomes little more difficult to find the file. Sometimes the investigators have to apply different approaches(hash calculation and matching , content matching etc) to all the files. This process is time consuming and requires resources. In most of cases if the metadata of file is changed, then it becomes a hectic job to find the file with traditional search.

With weight based approach, it will be little easy for searching the file even if the metadata of file is changed. The following command was executed to find a file named gui.py.

analyze.py -e py -s 10-100000 -a 2015-201508 -m 2015-20150825 -c 2015-201508 -f gui -p d:\

The results of this command(given by get_file.py) were as follows -

- Number of files satisfying 0 filters - 127096
- Number of files satisfying 1 filters - 253756
- Number of files satisfying 2 filters - 57238
- Number of files satisfying 3 filters - 22802
- Number of files satisfying 4 filters - 4846
- Number of files satisfying 5 filters - 1662
- Number of files satisfying 6 filters - 1
- Number of files satisfying 7 filters - 0

The following graph represents the results -

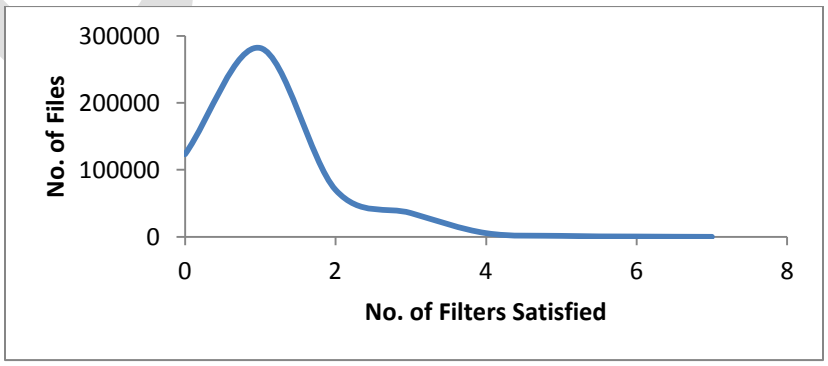


Figure - Number of files VS Number of filters satisfied

From the above graph and reading, it can be concluded that for all non-zero positive number, more the number of filters satisfied less the number files. As the number of filters being satisfied decreases, the number of files increases. In the example the file was present and it satisfied all the filters and so was easily found. Suppose one of the property of gui.py was changed. So if the same command was executed this time, it's obvious this time the file satisfying 7 filters will be zero and files satisfying 6 filters will increase by 1. Now instead of doing different function(hash matching, content matching etc) on all files(nearly half million on this system), this functions can be done on the 505 files instead (i.e. files satisfying 6 filters).Same will be the case when 2 or more properties are changed. In this way the probability of finding the file as soon as possible increases. This would reduce time and make less use of resources.

*E. Use of traces*

The traces can be further used to get more accurate data about the files. This information can be further used to provide filters to analyze.py module more accurately and precisely. In our project, prefetch files were used. From prefetch files , information like when was the particular application last time opened. Analyze.py takes the MAC time as a range rather than a specific date and time. So the timestamp recovered from prefetch will be used as a upper end limit while providing the date and time range for analyze.py module.

When information recovered from traces are used as filters to search a file, the number of files satisfying higher number of filters will decrease and the number of files satisfying less number of filters will increase. This also increases the probability of getting the file faster. For example - if from .lnk files we get the file path. But if the file is renamed or moved to other folder or deleted then the .lnk file will not be able to trace the file and so the trace becomes useless. But the .lnk files have ceratin metadata like the path. So if we use the path found in .lnk file to provide the filter then it may or may not help us to get more accurate results.

Let's take the same example executed above in section D. But this time the path will be set to the path extracted from .lnk file

analyze.py -e py -s 10-100000 -a 2015-201508 -m 2015-20150825 -c 2015-201508 -f gui -p E:\PROJECT\implementation

The results of the command(given by get_file.py) are as follows -

- Number of files satisfying 0 filters - 127014
- Number of files satisfying 1 filters - 241227
- Number of files satisfying 2 filters - 56340
- Number of files satisfying 3 filters - 35725
- Number of files satisfying 4 filters - 5393
- Number of files satisfying 5 filters - 1700
- Number of files satisfying 6 filters - 1
- Number of files satisfying 7 filters - 1

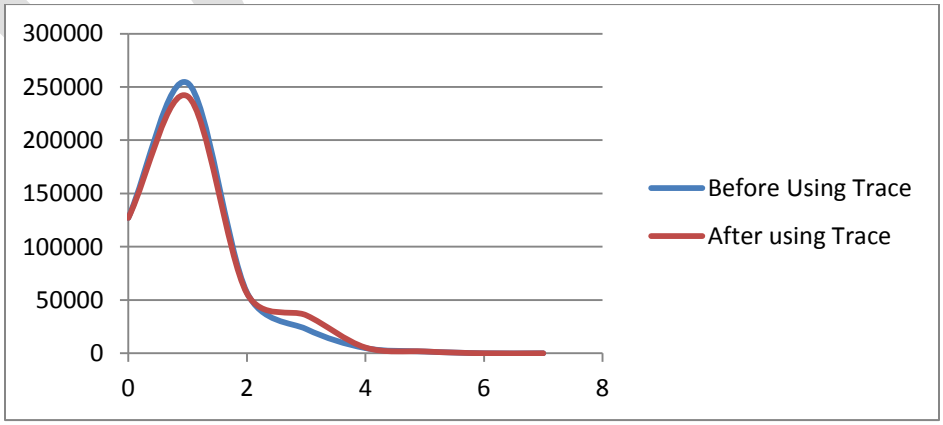


Figure - Difference in result after using .lnk trace

This is the general behaviour of how the number of files and number of filters satisfied will be affected when filter are used from data extracted from traces.

There are other traces that can be used to extract more information . This information can then be used to more precisely provide filters.[6]

1. Registry - List of executed commands, search keywords, last accessed folder, recently executed files and application usage
2. Web Browser File - Visited URL/time, downloaded files, search keywords
3. Specific Document File - Encrypted files, file name and files with modified extension
4. $MFT file : file and directory names, file extension, creation time, modification time, access time, file size, and file location and its possessor.

More the number of traces used as filters, more precise will be results.

## CONCLUSION

The use of traces and filters in forensic investigation for analysis and getting evidence have always been very important. Little changes to data associated with traces and filters can get the evidence away from investigation prompting the investigator to use more complex investigation process. But using the weight based approach to get the evidence by use of traces and filter can help the investigator in getting the all the data on their priority basis. Even if the evidence file is changed, weight-based-approach will help in placing it among the results(which would not be placed among results using traditional working) based on changes made to file and the weight assigned to it during process of analysis.

**REFERENCES:**

1. TheGuarding Newspaper [Online] Available : http://www.theguardian.com/news/datablog/2012/dec/19/bigdatastudydigitaluniverseglobalvolume
2. Siti Rahayu Selamat, Robiah Yusof, Shahrin Sahib, Nor Hafeizah Hassan, Mohd Faizal Abdollah, Zaheera Zainal Abidin "Traceability in Digital Forensic Investigation Process" published in 2011.
3. Shelton Donald E. " The 'CSI Effect': does it really exist? " March 2008; http://www.ojp.usdoj.gov/nij/journals/259/csieffect.htm.
4. Simson L. Garfinkel "Digital forensics research: The next 10 years" published in 2010.
5. Mahendra S. Pandit " Weight-Based-Priority Approach for Analyzing Data Using Traces and Filters " - (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 2537-2540
6. SeungBong Lee, Jewan Bang, KyungSoo Lim_, Jongsung Kim, and Sangjin Lee "A Stepwise Methodology for Tracing Computer Usage" in 2009.
7. Joshua I. James and Pavel Gladyshev, "Challenges with Automation in Digital Forensic Investigations" published in 2013.
8. Brian Carrier "Open Source Digital Forensics Tools: The Legal Argument" published in October 2002
9. Brian D. Carrier Eugene H. Spafford, "Automated Digital Evidence Target Definition Using Outlier Analysis and Existing Evidence" published in 2005.
10. Xiyan Zhao ; Key Lab. of Intell. Comput. & Novel Software Technol., Tianjin, China ; Yukun Li ; Jingyu Liu ; Yingyuan Xiao "Searching Desktop Files Based on Synonym Relationship" , Published in: Web Information System and Application Conference (WISA), 2013 10th.
11. Pavel Dmitriev , Pavel Serdyukov ,Sergey Chernov "Enterprise and Desktop Search" published in http://www.wwwconference.org/ in 2010.
12. Chang-Tien Lu ; Virginia Polytech. Inst. & State Univ., Blacksburg ; Shukla, M. ; Subramanya, S.H. ; Yamin Wu "Performance Evaluation of Desktop Search Engines" Published in Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference
13. Cole, B. "Search engines tackle the desktop" Published in Computer (Volume:38 , Issue: 3 ) IEEE Paper.
14. Gaugaz J. ;Costache, S. ; Chirita, P.-A. ; Firan, C.S. ; Nejdl, W."Activity Based Links as a Ranking Factor in Semantic Desktop Search" Published in Web Conference, 2008. LA-WEB '08., Latin American