CENTRALIZED DIGITAL FORENSIC ANALYTICAL SYSTEM: RATIONALIZATION USING OPEN-SOURCE SOFTWARE OF GENERAL PURPOSES

Igor Vukovic[1]
Ministry of the Interior of the Republic of Serbia

Abstract: Digital forensics is a part of everyday work of law enforcement agencies and corporate security and has numerous challenges. Forensic investigators need to make a continual investment in new tools and methods of extracting data from a large number of different devices. They can save financial resources in the second critical phase of the digital forensics process, which is an analysis and search for evidence. This paper discusses the possibilities of development of a forensic analytical system which primarily enables cross-search through data and collaboration of investigators by using general purpose open-source tools. The main feature of the open-source tool is that they are mostly free of charge and free to use, which provides a significant saving. Bash scripts that organize tools and commands execution on the specific task allow a forensic investigator to create custom tools that meet their needs and increase job efficiency.

Keywords: digital forensics, open-source tools, NoSQL database, Bash script.

INTRODUCTION

Digital forensics has made progress in the last few decades of its development from "obscure tradecraft to an important part of many investigations" (Garfinkel, 2010: 64), and today digital forensic tools are in everyday use. Forensic investigation is a complicated and time-consuming process that requires researchers skills from multiple fields, such as information security, penetration testing, reverse engineering, programming and behaviour profiling in order to reach the ultimate goal, which is to create a clear picture of the crime committed using a computer (Koen, 2009).

The two main areas in which digital forensics is applied most intensively are law enforcement agencies and corporate security, with each having a different approach (Bassett, Bass & O'Brien, 2006). In brief, corporate security focuses on adequate response and damage assessment (Cohen, Bilby & Caronni, 2011), while

1 igor.vukovic@mup.gov.rs

law enforcement agencies seek to find traces of the crime, computer programs or data that can indicate the link between the offence and the perpetrator or victim in a large amount of data. With respect to both these areas, digital forensics can be defined as a process in which scientific principles are applied in order to analyse electronically stored information (artifacts) on one or more digital devices, in order to determine the sequence of events that led to a particular incident, as well as the understanding and reconstructing events that must have transpired in generating the said artifacts (Raghavan, 2013).

Modern digital forensics faces numerous obstacles, even without the use of anti-forensic techniques. The necessary picture of what happened in a particular case is most often challenging to create due to inconsistencies and a lack of incriminating evidence (Koen, 2009). We could divide the forensic investigation process into two phases, Acquisition Phase and Analysis Phase (Brian Carrier, 2002), each with its challenges.

The process of data acquisition is hampered by a large number of various devices from which it is necessary to collect digital evidence; different methods of protecting devices from unauthorized access; data is no longer on devices but on different virtual storage (cloud technology). A particular frustration for the forensic community is generated by mobile phones since the beginning of their proliferation. As mobile phones develop faster than other technologies, a significant problem is the development of the tool that would be considered as an industry standard (Yates, 2010). The paradigm of "internet of things" and the development of even more devices that contain forensic artifacts will further complicate the acquisition process. Also, the evidence needs to be collected from several technological zones, from the devices themselves, the Network zone, and the Cloud zone (Miljković, Čabarkapa, Prokin & Budimir, 2018). Another issue is the protection of modern devices that does not require a specific user activity and advanced technical knowledge because various authentication mechanisms (passwords, fingerprint scanning, or face recognition) are already present by default. Encryption of data storage is already a constant problem for forensic investigators, and with further hardware development, it becomes more a rule rather than an exception. In order to overcome all problems, forensic investigators must continuously invest in new data collection tools and techniques, so any kind of rationalization in this phase of the process is impossible.

The data analysis phase consists of searching for evidence among the collected data. We want to confirm the hypothesis of some event or find arguments to refute the hypothesis, or at least show that a particular system has altered in order to conceal evidence (Brian Carrier, 2002). The point is, however, in challenges that arise in order to reach this evidence like data diversity, a large amount of data, struggle with storage space, and bottlenecks affecting performance (Wang & Alexander, 2015). Even considering those challenges, the analysis phase is where forensic investigators can make rationalization of financial resources. The purpose of this paper is to demonstrate that it is possible to create a centralized forensic

analytical system using open-source technology, with the emphasis on the use of tools that are not forensic, but for general purposes. Linking the functionality of particular tools was done by using a Bash script that requires knowledge of programming basics (variables without too much consideration about the type, conditional statements, and control flow). Application of the different tools and the Bash scripts allows the forensic investigator to build the system according to his needs or the needs of a particular case. Investigators can always improve their scripts by merely adding new tools with a necessary set of options.

Researches on the subject of free software in digital forensics analyse solutions that are more oriented to specific forensic tasks such as OpenLV (Vidas, Kaplan & Geiger, 2014), a forensic email analysis tool using dynamic visualization (Stadlinger & Dewald, 2017), or tool for forensic acquisition and analysis of the VMware virtual hard drive (Hirwani, Pan, Stackpole & Johnson, 2012), without any possibility for investigators to collaborate with each other. In recent papers, there is no analysis of the possibility of using shell scripts and open-source tools in order to create an analytical system. The forensic utility *fiwalk* is somewhat applicable to the concept and represents a batch forensic tool, a solution that uses the Python script, relies on an open-source forensic tool, but under the collaboration implies the exchange of XML files (Garfinkel, 2009).

The rest of the paper is organized as follows: the second section presents technologies and commands used to build a system and the way how this can be achieved; the third section shows the results of the work through demonstration of functionalities required for such system and relation of applied technologies to requirements are discussed; a brief conclusion also contains guidelines for further work; the last section is providing a list of used references.

BUILDING A SYSTEM

Data that digital forensic investigators collect and analyse are related to the file, such as its path to the original medium, name, size, type, and other information obtained from the file system. There are metadata which also represent file parameters (for example, a mark and the model of the camera which took the photo, if GPS is on we can find the spatial references) but they are part of the file content, not the file system. Then, investigators search the content of the file itself.

Data collection

Applications that are usually built-in as part of operating systems distributions based on the Linux kernel could provide the needed information. For example, with the *stat* command, we can learn the size and the last file modification time, the *baseman* can extract the file name from the absolute path, a *file* can obtain file type information. By installing packages for forensic purposes, the commands of

these packages can be used to get all the available details mainly found in file systems such as NTFS (like a file creation time) and other non-Linux environments.

After the mounting source of digital evidence, which can be a forensic copy, storage medium from the case (with the usage of write blocker) or any other media that contains data for analysis, it is necessary to go through all the folders and find all the files. The *find* command can be used for this purpose, as it returns a list of absolute paths of all files. With *read* command, we can go through each absolute file path using them as the input parameter for other tools and commands. Besides, the full path information itself is a forensic artifact. An example of using the commands mentioned above for extracting file information data is shown in Figure 1.

```
#!/bin/Bash
find "$(pwd)"  -type f | while read -r f_path;
      do
      case_name="$1"
      device_description="$2"
      f_name=$(basename -- "$f_path")
      f_parent_path=$(dirname "$f_path")
      f_ext=$(echo "${f_name##*.}" | tr '[:lower:]' '[:upper:]')
      f_type=$(file -b "$f_path")
      f_mime=$(file --mime "$f_path")
      f_mod_time=$(stat -c '%y' "$f_path")
      f_size=$(stat --printf="%s" "$f_path")
      f_sum=$(md5sum "$f_path" | awk '{ print $1 }')
…
done
```

Figure 1: *Collecting file information from the file system*

In addition to file information that various tools already collected from a file system, the next tool named *exiftool* extracts a various set of metadata depending on the file type as shown in Figure 2.

```
#!/bin/Bash
find "$(pwd)"  -type f | while read -r f_path;
      do
      …
      temp_dir="/home/eve/forensics/temp_dir/"
      case $f_ext in
            JPEG | JPG | PNG | PSD)
                  fmd_make=$(exiftool -s -s -s -make "$f_path")
                  fmd_model=$(exiftool -s -s -s -model "$f_path")
                  fmd_software=$(exiftool -s -s -s -software "$f_path")
                  fmd_lat=$(exiftool -s -s -s -gpslongitude "$f_path")
                  fmd_long=$(exiftool -s -s -s -gpslatitude "$f_path")
                  fmd_w=$(exiftool -s -s -s -imagewidth "$f_path")
                  fmd_h=$(exiftool -s -s -s -imageheight "$f_path")
                  fmd_mdate=$(exiftool -s -s -s -filemodifydate "$f_path")
                  f_md="metadata: [{make:'$fmd_make',
model:'$fmd_model',software:'$fmd_software',lat:'$fmd_lat',long:'$fmd_long',
w:'$fmd_w',h:'$fmd_h',mdate:'$fmd_mdate'}],"
                  ;;
            …
      esac
…
done
```

Figure 2: *Collecting metadata with exiftool command*

Each tool tries to display a more transparent list of results, which is also true for *exiftool*. To provide only the necessary data (image size, processing software

and other), we need to execute *exiftool* command repeatedly several times with the different options (repeating the option -s tree times separates the value from a result, for example, image height).

The most important information for the forensic investigation is within the contents of the file. Unlike the tools that collect data from the file system and metadata, in order to extract as much information from the contents of the file one open source tool is not enough, for each type of file or set of files with a similar purpose we need different ones. For example, the LibreOffice open-source package offers the ability to read the contents of the file it creates, as well as several other proprietaries file types like Microsoft Office files (Word, Excel, PowerPoint).

We usually use LibreOffice application over a graphical user interface, but it also could be executed from the command line interface. What we should keep in mind is how to read the file to obtain the expected results. When it comes to the LibreOffice application, it needs to provide an absolute path to the file and directory where we save the output, that is, the text file with the LibreOffice file content. The output directory in the script shown in Figure 3 is for temporarily storing content until its further processing. The same principle applies to the *pdftotext* command, which converts PDF file to text file based on the absolute path of the PDF file. What is different is that for pdftotext tool the user defines the output file name of a temporary text file with the content of the PDF file, but in the case of the *libreoffice* command we need to extend the script for two more lines in order to get the name of a temporary file. The name of the temporary files is required later during script execution to read the text content and to delete the temporary file.

```
#!/bin/Bash
find "$(pwd)"  -type f | while read -r f_path;
      do
      …
      temp_dir="/home/eve/forensics/temp_dir/"
      case $f_ext in
            …
            DOC | DOCX | ODT)
                  libreoffice --convert-to txt --outdir $temp_dir
"$f_path" &> /dev/null
                  f_ext_len=$(expr length $f_ext)
                  subs=$(echo ${f_name:0:-f_ext_len})
                  tempfile=$temp_dir$subs"txt"
                  ;;
            PDF)
                  pdftotext -q "$f_path" $temp_dir"temp.txt"
                  tempfile=$temp_dir"temp.txt"
                  ;;
            …
      esac
…
done
```

Figure 3: *Conversion of Office and PDF file contents into a text*

Open-source applications and tools allow us to convert files into more accessible formats, for which it would otherwise be necessary to have original proprietary applications, for example, Adobe Photoshop or Corel Draw. Inkscape is a profes-

sional open-source vector graphics editor, and its natural working environment is a graphical interface, but with the *inkscape* command, it is possible to convert various formats used in graphic design within a script as shown in Figure 4 into a PNG format readable with any image preview application or Internet browser.

```
#!/bin/Bash
find "$(pwd)"  -type f | while read -r f_path;
      do
      …
      temp_dir="/home/eve/forensics/temp_dir/"
      case $f_ext in
            …
            CDR | SVG | EPS | AI)
                  img_rep_path="$img_repository""$case_name"-$f_sum.png
                  inkscape "$f_path" --export-png="$img_rep_path"
                  img_rep_path=",'img_rep_path':'"$img_rep_path"'"
                  ;;
      …
      esac
…
done
```

Figure 4: *Converting files with vector graphics into a raster-graphic file format*

Microsoft Windows operating system, for example, uses extensions of file names to determine which type of file it is, but forensic investigators cannot rely on extensions. They can be deliberately replaced by users, making it harder to find specific types of files, or applications that could exclude extensions entirely (Firefox Mozilla puts the SQLite file name extension, while Google Chrome does not). In Figure 5, we can see how to identify file type by using the *file* command.

Applications often use SQLite as a database manager, so Figure 5 shows how the *sqlite3* extracts tables and their contents stored in the SQLite file found in the case. Unlike most other files, the *sqlite3* application cannot access the SQLite file from the write-protected medium. Therefore, we copy the SQLite file to the temporary directory in order to access it, read content from all tables, and save it as a text for further processing.

```
      …
      temp_dir="/home/eve/forensics/temp_dir/"
      case $f_ext in
            …
            *)
               sqlite=$(echo $f_type | grep 'SQLite' | wc -l)
               if [ $sqlite -gt 0 ]
               then
                  temp_sqlite=$temp_dir"sqlite"
                  cp $f_path $temp_sqlite
                  echo ".tables" | sqlite3 "$temp_sqlite" | while read -r
table; do
                  for word in $table
                        do
                        echo "naziv tabele: "$word >> $tempfile
                        sqlite3 -csv $temp_sqlite "select * from '"$word"';"
>> $tempfile
                        done
                  done
                  rm "$temp_sqlite"
                  tempfile=$temp_dir"temp"
                  fi
      …
      esac
```

Figure 5: *Extraction of SQLite tables content into a text file*

Using the --mime option of a command *file*, it is possible to determine whether it is a binary file. If this is not the case, the contents of the file are textual and can be read instantly, while in the case of a binary file, we can extract a group of printable characters with *strings* command as shown in Figure 6.

```
…
temp_dir="/home/eve/forensics/temp_dir/"
case $f_ext in
      …
      *)
         charset=$(echo $f_mime | grep 'binary' | wc -l)
         sqlite=$(echo $f_type | grep 'SQLite' | wc -l)
         if [ $sqlite -gt 0 ]
         then
         …
         elif [ $charset -eq 0 ]
         then
            f_content=$(cat "$f_path")
            f_content=$(echo $f_content | tr -cd '[:alnum:].,/: ')
         else
            strings -10 "$f_path" > $temp_dir"temp"
            tempfile=$temp_dir"temp"
         fi
esac
…
```

Figure 6: *Reading text files and printable characters from binary files*

Differences in results of using *strings* command and the application that can read the contents of a particular binary format are shown in Figure 7. The text that represents the contents of the file on the right side of Figure 7 we cannot find in a text composed of printable characters on the left side of the figure, meaning that strings command does not provide complete information about the content.

dzigi@dzigi-Lenovo-Z50-75: ~/Desktop

```
<< /AcroForm 4 0 R /LastModified (D:20090624150438) /
MarkInfo << /LetterspaceFlags 0 /Marked true >> /Meta
data 5 0 R /OCProperties << /D << /AS [ << /Category
[ /View ] /Event /View /OCGs [ 6 0 R ] >> << /Categor
y [ /Print ] /Event /Print /OCGs [ 6 0 R ] >> << /Cat
egory [ /Export ] /Event /Export /OCGs [ 6 0 R ] >> <
< /Category [ /View ] /Event /View /OCGs [ 7 0 R ] >>
 << /Category [ /Print ] /Event /Print /OCGs [ 7 0 R
] >> << /Category [ /Export ] /Event /Export /OCGs [
7 0 R ] >> ] /OFF [ ] /ON [ 6 0 R ] /Order [ ] /RBGro
ups [ ] >> /OCGs [ 9 0 R 6 0 R 18 0 R 7 0 R ] >> /Pag
eLayout /OneColumn /Pages 20 0 R /PieceInfo << /Marke
dPDF << /LastModified (D:20090624150438) >> >> /Struc
tTreeRoot 40 0 R /Type /Catalog >>
<< /Type /ObjStm /Length 211 /Filter /FlateDecode /N
1 /First 4 >>
<< /DA (/Helv 0 Tf 0 g ) /DR << /Encoding << /PDFDocE
ncoding 140 0 R >> /Font << /Helv 145 0 R /ZaDb 146 0
 R >> >> /Fields [ ] >>
<< /Subtype /XML /Type /Metadata /Length 4017 >>
<?xpacket begin="
```

dzigi@dzigi-Lenovo-Z50-75: ~/Desktop

```
l crime. This unfortunate
situation may potentially allow computer criminals to
 commit crimes using
technologies for which no proper forensic investigati
ve technique currently exists.
Such a scenario would ultimately allow criminals to g
o free due to the lack of
evidence to prove their guilt.
A solution to this problem would be for law enforceme
nt agencies and
governments to invest in the research and development
 of forensic technologies
in an attempt to keep pace with the development of di
gital technologies. Such an
investment could potentially allow new forensic techn
iques to be developed and
released more frequently, thus matching the appearanc
e of new computing
devices on the market.
A key element in improving the situation is to produc
e more research
```

Figure 7: *Comparison of the results of the strings command and pdftotext command working on the same file*

Data import

Figure 8 shows how the script can import all the collected information into the MongoDB NoSQL open-source database. From temporary files where the script stored textual content of a subject file, the script read data, and then along with other information collected from the file system and metadata, they are entered as a document within a database collection. The *mongoimport* command imports data into a database that script previously saved in a file as a JSON matrix.

```
#!/bin/Bash
find "$(pwd)"  -type f | while read -r f_path;
      do
      …
      temp_dir="/home/eve/forensics/temp_dir/"
      …
      if [ ${#tempfile} -gt 0 ]
      then
            f_content=$(cat "$tempfile")
            f_content=$(echo $f_content | tr -cd '[:alnum:].,/: ')
      fi
      echo "[{'case_name':'$case_name',
'device_description':'$device_description',' f_name':'$f_name',
'f_parent_path':'$f_parent_path', 'f_ext':'$f_ext', 'f_mime':'$f_mime',
'f_type':'$f_type', 'f_path':'$f_path', 'f_mod_time':'$f_mod_time',
'f_size':'$f_size', 'f_sum':'$f_sum',
"$f_md"'f_content':'$f_content'"$img_rep_path"}]" | iconv -f latin1 -t utf8
-c > $temp_dir"temp.json"
      mongoimport --quiet --db forensics --collection files --type json --
jsonArray --file $temp_dir"temp.json"
      …
done
```

Figure 8: *Importing collected data into a NoSQL database*

Analysis and Reporting

Bash scripts and various commands can be used to search over imported data and create reports in a context that the forensic investigator needs or chooses. The script is obtaining database search results in the JSON format after that command *jq* parses data into an array. The report function saves the data in a text file with name extension HTML that can be opened later in the Internet browser. One of the ways to search the database and create a report is shown in Figure 9.

```
report() { while read line; do echo $line>> report.html; done; }
echo "db.files.aggregate([
      {\$project:{_id:0,f_path:1,f_ext:1,img_rep_path:1,f_sum:1,f_content:1,
length:{\$strLenCP:'\$f_content'}}},
      {\$match:{f_ext:{\$in:['PDF','AI']}}}]).forEach(printjson);" >
$temp_dir"q.js"
mongo --quiet forensics '/home/eve/forensics/temp_dir/q.js' | jq -r
'(.f_path)+"\t<img src=\""+(.img_rep_path)+"\
width=\"300\"">\t"+(.length|tostring)' | while read -r row;
#+"\t"+(.length|tostring)'
do
      IFS=$'\t' read -r -a items <<< "$row"
      echo "<tr>" | report
      for item in "${items[@]}"
      do
            echo "<td bgcolor=\"#F5D0A9\">" | report
            echo $item | report
            echo "</td>" | report
      done;
      echo "</tr>" | report
done
```

Figure 9: *Database search and report creation*

RESULTS AND DISCUSSION

Digital forensic cases often include several suspects, and each of them has few devices which need storage for large amounts of data. For the sake of efficiency in such cases, it is necessary to simultaneously engage the investigational forensic team, which is why one of the requirements is the development of new forensic techniques, procedures, and tools, namely collaboration (Garfinkel, 2010). The tool must provide the possibility of real-time collaboration, in asynchronous way, and remote access to achieve the most optimal conditions, as well as the effects of team work (Garfinkel, 2010; Hibshi, Vidas & Cranor, 2011).

One of the prerequisites for collaboration is the parallel import of data that allows the team of investigators to import data from multiple devices from the same case or several different cases at the same time. Launching, for example, one script with various instances of the Terminal program in Linux can concurrently import data from one computer or multiple computers in the network (Figure 10).

Figure 10: *The five instances of the Terminal program use the same script and at the same time import data into the database*

Figures 11 and 12 show the simultaneous import of data by recording two different states of the importing system, a change in the number of documents in the database and the size of the log files.

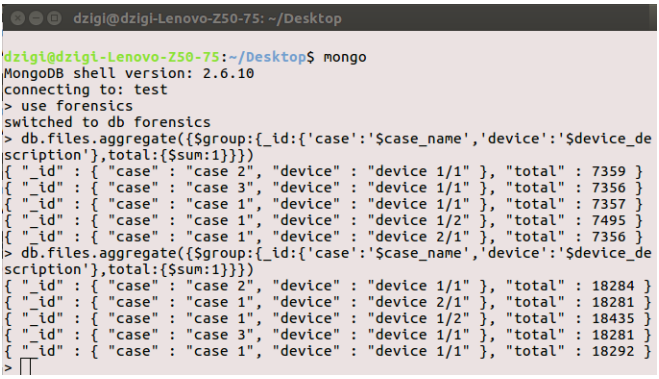

Figure 11: *Change of the number of documents in the database during import*

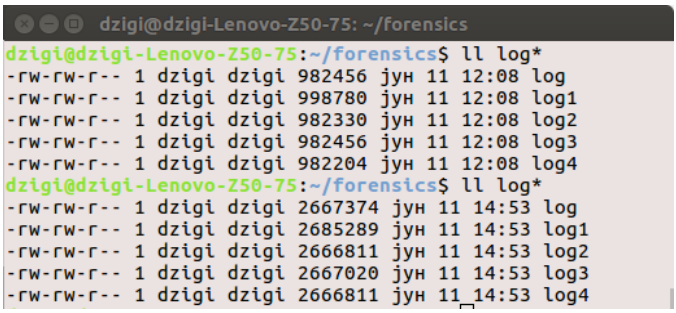

Figure 12: *Change of the log file size during the data import*

Open-source tools that can meet the requirements are database managers, and we distinguish two types by the databases they manage: Relational and NoSQL (Could be Non-SQL or Not only SQL).

The use of NoSQL databases for forensic purposes is suitable even though they do not provide the implementation of ACID (Atomicity, Consistency, Isolation, Durability) properties of a database transaction that should provide validity even in the event of errors, system failures for various reasons as relational managers do (Roussev, 2011). After the initial import of data, forensic investigators access data only by reading, and not changing or deleting them, so consistency issues of the database are not expected (Federici, 2013).

The advantage of using NoSQL databases is that the concept of a document is much closer to file properties which ultimately differentiates by type, quantity, and content of its metadata and other parameters. In Figure 13, there are two MongoDB collection files, the first document contains data about a file that does not have metadata, and the other includes metadata, without the need to explicitly change the structure as it would be the case with relational databases. NoSQL databases solve this problem by offering the ability to store objects and matrices in one field (Boicea, Radulescu & Agapin, 2012).

Figure 13: *An example of two documents from the same database collection with a difference in the structure*

Scalability is another important feature of the NoSQL database manager. The amount of new information entering into forensic analytical systems is significant, and it is difficult to anticipate what resources are sufficient to meet the needs of the job. The ability to expand storage capacity without disturbing the functioning of the working database gives priority to NoSQL systems relative to relational managers (Boicea et al., 2012).

In addition to storage space and collaboration requirements, the basis for each forensic analytical tool is the ability to search to find information or digital evidence (Koen, 2009). In Figure 14 there are two examples of search queries that use regular expressions for finding IP addresses, as well as searching multiple terms at the same time and displaying only the data from the corresponding fields (in this case it is an absolute path).

```
db.files.aggregate([
      {$match:{f_content:{$regex:'\b((25[0-5]|2[0-4][0-9]|[01]?[0-
9]?)(\.|$)){4}\b'}}}
      ])

db.files.aggregate(
      {$match:{"f_content":{$in:[/<search term>/,/<search term>/,...]}}},
      {$project:{"_id":0,"f_path":1}}
      )
```

Figure 14: *MongoDB queries enable different searches for forensic artifacts*

NoSQL databases are more efficient than relational when it comes to response rates. The same applies to open-source relational database managers (Gyorodi, Gyorodi, Pecherle & Olah, 2015), as to the proprietary database managers, even for those which are considered an industry standard (Boicea at al., 2012; Wu, Huang & Lee, 2015). That is why NoSQL databases are a recommended solution for intensive enrolment applications and queries against massive amounts of data (Gyorodi et al., 2015). Creating indexes for fields containing texts further accelerates the search.

In order for a forensic tool to be considered reliable, it is necessary to meet conditions such as the accuracy of the information particular system produces; that there are no errors; data must always be available for analysis, and that there is adequate support in order to eliminate the detected errors (Koen, 2009). The last reference opponents of open-source software often use to criticize, because of support and work on specific projects based on enthusiasm. In this research, scripts consist of open-source tools and commands whose primary purpose is not digital forensics, but as an everyday application in different areas of human work. The number of users of such applications exceeds the size of the forensic community gathered around a particular tool, which allows errors to be detected and repaired more efficiently.

The main reason why forensic investigator needs to create a tool for himself lies precisely in the fact that most of the current forensic tools, whether they are

free or proprietary, are designed in "evidence-oriented" fashion (Hibshi et al., 2011: 82), or "for finding evidence where the possession of evidence is the crime itself" (Garfinkel, 2010: 68). Also, manufacturers of a forensic software are always balancing between demand and profit, and even though comprehensive, their tools and systems do not need to respond to every request by a forensic investigator or a specific case (Horsman, 2019). Customization of a forensic analytical system by selecting only specific tools related to a particular context, in order to improve efficiency and effectiveness of the script, lead to another condition that a forensic tool should satisfy which is usability (Hibshi et al., 2011). Most forensic investigators are not programmers, but even if they are, they are not willing to write a code from the beginning (Hibshi et al., 2011). By using commands and Bash script, they already have all the functionality they need to complete the task.

Forensic tools and systems, regardless of their efficiency supported by high-performance hardware, are often not sufficient to finish a task on time, so investigators are using triage. It represents a way to shorten the time of analysing all the collected evidence by focusing on specific directories or file types instead of doing the keyword search of all files (Cohen et al., 2011). Importing only data about the file that the script collects from the file system is a process that takes a relatively short time. After that, a query on a database can help with the triage showing the locations of searched file types sorted by their quantity in the specific location as shown in Figure 15.

```
db.files.aggregate(
      {$match:{f_ext:{$in:['DOCX','DOC','PDF']}}},
      {$group:{_id:'$f_parent_path', total:{$sum:1}}},{$sort:{total: -1}}
      )
```

Figure 15: *An example of a query that helps during triage*

Using the script shown in Figure 9 after we searched for information about files, we could also create a report in HTML format which is convenient for submitting reports since all operating systems in use today contain Internet browsers. Creating a report is the end of the forensic investigator's work on the case.

Forensic investigators are inclined to rely on commercial solutions, especially those dealing with criminal digital forensics, believing that specific tools have been accepted by courts (Meyers & Rogers, 2005). The problem with commonly accepted commercial tools is that forensic investigators can hardly testify in court on reliability of the tool and the procedures because manufacturers tend to hide everything (Meyers & Rogers, 2005). The essence of the trial is in the forensics investigators themselves and their knowledge about the methods and tools they use, and not what tool it is. Open-source is a more convenient solution because investigators can get a clearer insight into what a particular tool does and how, and the methods discussed in this paper allows automation of testing itself using the same principles creating a script with various commands and options.

CONCLUSION

The open-source concept speaks of freedom, such as using the software for any suitable purpose; freedom to study an application's source code; free redistribution of the applications, and for the improvement of the software for a particular purpose (Koen, 2009), but the fact is that such software is in most cases free of charge. Ubiquitous computing increase volume and variety of data available for digital forensic analysis (Popović, Kuk & Kovačević, 2018), however, this paper shows that by using open-source tools in the matter of digital forensics, we can achieve more than "something for nothing" (Goode, 2005: 669). Free tools developed under various open-source licenses provide great possibilities of applying in the field of digital forensics, and building of an efficient costumed centralized analytical system is more of a question of specific needs of a forensic investigator or forensic teams.

The advantage of using different open-source tools for collecting information using Bash scripts is that the system can be developed gradually as the forensic user finds and masters a new tool. Any new tool alone or combined with other tools can upgrade the script or could be used to create a new one.

Working with Bash scripts looks like programming, but in fact, it is much closer to organizing the running of various tools while choosing their options is a condition for gaining the expected results. It is important to emphasize that the relation of what one could achieve comparing with a length of the script itself is inversely proportional, while the syntax is more straightforward than in programming languages.

Systematization of various file types the content of which can be converted to text and imported into a database, as well as the investigation of the ability of script to work with compound files, such as ZIP and other archive formats, will be subject to the future work.

REFERENCES

1. Bassett, R., Bass, L. & O'Brien, P. (2006). Computer Forensics: An Essential Ingredient for Cyber Security. *Journal of Information Science and the Technology*, 3(1), 22-32.
2. Boicea, A., Radulescu, F. & Agapin, L. I. (2012). MongoDB vs Oracle - database comparison. *3rd International Conference on Emerging Intelligent Data and Web Technologies*, conference publications (Page 330-335). Bucharest, Romania: Institute of Electrical and Electronics Engineers.
3. Brian Carrier (2002) Open Source Digital Forensics Tools, the Legal Argument, Downloaded May 14, 2019 https://pdfs.semanticscholar.org/2825/a02f96a7962cff236443fbdd1d61931a071e.pdf.

4. Cohen, M.I., Bilby, D. & Caronni, G. (2011). Distributed forensics and incident response in the enterprise. *Digital Investigation*, 8, 101-110.
5. Federici, C. (2013). AlmaNebula: a computer forensics framework for the Cloud. *Procedia Computer Science*, 19, 139 – 146.
6. Garfinkel, S.L. (2009). Automating Disk Forensic Processing with SleuthKit, XML and Python. *Fourth International Systematic Approaches to Digital Forensic Engineering, International Workshop*. Oakland, California, United States of America: Institute of Electrical and Electronics Engineers.
7. Garfinkel, S.L. (2010). Digital forensics research: The next 10 years. *Digital Investigation*, 7, 64-73.
8. Goode, S. (2005). Something for nothing: management rejection of open source software in Australia's top firms. *Information & Management*, 42, 669–681.
9. Gyorodi, C., Gyorodi, R., Pecherle, G. & Olah, A. (2015). A Comparative Study: MongoDB vs. MySQL. *13th International Conference on Engineering of Modern Electric Systems*, conference publications (Page 189). Oradea, Romania: Institute of Electrical and Electronics Engineers.
10. Hibshi, H., Vidas, T. & Cranor, L. (2011). Usability of Forensics Tools: A User Study. *Sixth International Conference on IT Security Incident Management and IT Forensics,* conference publications (Page 81-91). Stuttgart, Germany: Institute of Electrical and Electronics Engineers.
11. Hirwani M., Pan Y., Stackpole W. & Johnson D. (2012). Forensic Acquisition and Analysis of VM ware Virtual Hard Disks. *International Conference on Security and Management*. Las Vegas, Nevada, United States of America: University of Detroit Mercy.
12. Horsman, G. (2019). Tool testing and reliability issues in the field of digital forensics. *Digital Investigation*, 28, 163-175.
13. Koen, R. (2009). *The development of an open-source forensics platform*. Pretoria, South Africa: University of Pretoria.
14. Meyers, M. & Rogers, M. (2005). Digital Forensics: Meeting the Challenges of Scientific Evidence. *IFIP International Conference on Digital Forensics*, conference publications (Page 43-50). Orlando, Florida, United States of America: National Center for Forensic Science.
15. Miljković, A., Čabarkapa, M., Prokin, M. & Budimir, Đ. (2018). The Importance of IoT and IoT Forensics, *Archibald Reiss Days*, 1(2), 395-404.
16. Popović, B., Kuk, K. & Kovačević, A. (2018). Comprehensive Forensic Examination with Belkasoft Evidence Center, *Archibald Reiss Days*, 1(2), 419-433.
17. Raghavan, S. (2013). Digital forensic research: current state of the art. *CSI Transactions on ICT*, 1(1), 91-114.
18. Roussev, V. (2011). Building Open and Scalable Digital Forensic Tools. *Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic En-*

gineering. Oakland, California, United States of America: Institute of Electrical and Electronics Engineers.

19. Stadlinger, J. & Dewald, A. (2017). A Forensic Email Analysis Tool Using Dynamic Visualization. *Journal of Digital Forensics, Security and Law*, 1(12), 7-14.
20. Vidas, T., Kaplan, B. & Geiger, M. (2014). OpenLV: Empowering investigators and first-responders in the digital forensics process. *Digital Investigation*, 11, 45–53.
21. Wang, L. & Alexander, C. A. (2015). Big Data in Distributed Analytics, Cybersecurity, Cyber Warfare and Digital Forensics. *Digital Technologies*, 1(1), 22-27.
22. Wu C. M., Huang, Y. F. & Lee, J. (2015). Comparisons between MongoDB and MS-SQL Databases on the TWC Website. *American Journal of Software Engineering and Applications*, 4(2), 35-41.
23. Yates, M. (2010). Practical Investigations of Digital Forensics Tools for Mobile Devices. *Information Security Curriculum Development Conference*, conference publications (Page 156-162). Kennesaw, Georgia, United States of America: InfoSecCD '10.