Forensic Analysis of Digital Attack Tool Artifacts

Fletcher Bayley and Diane Gan

CSAFE Centre, School of Computing & Mathematical Sciences, University of Greenwich, UK
D.Gan@gre.ac.uk

Abstract. This work was to investigate the forensics artifacts left by network attack tools within Linux and UNIX operating systems and to develop an application called HexaFind. The application enables a forensics investigator to collect the digital evidence left behind by the usage, installation or removal of specific attack tools. The main objective was to decrease the complexity of forensic investigations within these operating systems and to increase the detection rate of forensic artifacts relating to criminal or civil evidence of malicious conduct.

Keywords: Linux, UNIX, forensics investigation, digital evidence, network enumeration, digital artifacts, HexaFind, big data

1 Introduction

The continued and growing trend towards the frequency and severity of network attacks against corporations, private individuals and even towards countries has prompted the development of network attack detection tools. In order to defend against threats, security is not only required in a pre-emptive scenario, but also ex post facto, whereby the criminal has been detected and the acquisition of evidence has begun in order to facilitate a conviction.

Many system files within the Linux and UNIX operating systems are modified when different types of attack tools are used and these key incriminating files or repositories may be overlooked by law enforcement and forensic investigators when attempting to collect evidence left by a malicious user. It is also extremely useful for a forensic investigator to be able to identify whether an attack was perpetrated by the user, or whether the source of the attack came from outside and this can also be proved, by detailed analysis of the logs and resources within Linux. Currently there are only a few tools that can directly identify the use of a digital attack tool, within the Linux/UNIX operating system and therefore this work should help to address this imbalance.

Work similar to HexaFind has been identified. Hargreaves et al. discuss a framework which extracts timeline data to identify high level events using pattern matching [1]. An open-source tool to collect forensics evidence is proposed by Zhang et al., which runs on a live DVD/USB based on Linux [2]. Both of these papers include a discussion of Log2timeline, which is mainly a Windows based tool, but it does run on Linux platforms and creates a single timeline by identifying various artifacts and suspicious files within the file system [3]. Fsaudit is a scripted tool, written in Perl and created for the auditing of file system changes [4]. It has some similarities to HexaFind in that it highlights any suspicious changes or recently modified files.

As a Linux system is entirely based around text, whether stored as a string or an integer, all values are held within files. If it is not a file, then it is a process (which often cannot be interrupted). Therefore, the assumption is made, that the contents of a Linux/ UNIX hard drive may be enumerated easily. Strings held within files are able to identify compromised systems or those which had been used for attack purposes, by dynamically searching through every file (ignoring running processes), using specific search criteria. Results returned are based on a ratings value system using keywords, similar to a search engine. Search engines enumerate the results, looking for keywords and returning hits based on their inclusion/ exclusion. In contrast, the scripted application developed here uses data mining to examine the hard drive, returning results based on keywords and provides them with a relevancy rating.

The tool HexaFind was developed to assist a forensics investigator in determining if any digital attack tools have been used in a Linux operating system. The attack tools tested were Metasploit, Wireshark, Nmap, tcpdump and LOIC (Low Orbit Ion Cannon). By examining the system for digital fingerprints, a forensic investigator would be able to determine if an attack has occurred by identify the key tell-tale signs (digital artifacts) left behind by these tools using data mining.

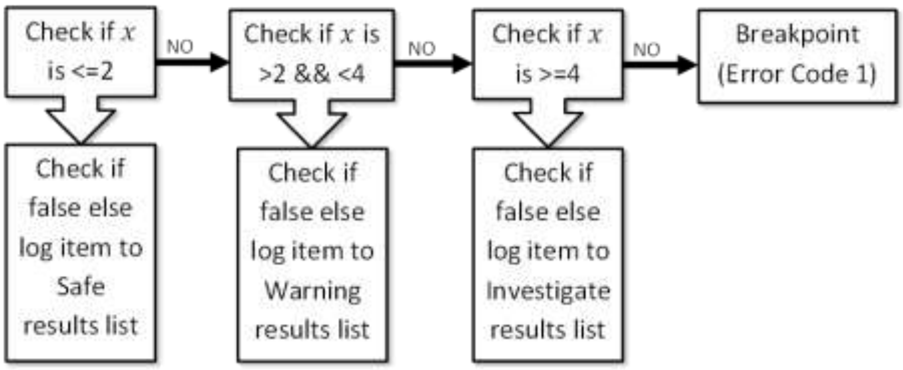
This paper is laid out as follows. Section 2 discusses the methodology used to build the HexaFind application, section 3 is an overview of the application and section 4 discusses the results obtained by running the tool after specific attacks had occurred. A discussion is presented in section 5 and the conclusion is found in section 6.

2 Operational Methodology of the Application

The application has been built using Bash (Bourne-again shell) scripting within a UNIX POSIX environment, working with multiple distributions. This programming language was chosen in order to keep the program portable, and reduce complexity. All results retrieved by the scripted application are based around the semantics of language or symbolic notation. Due to the way in which Linux creates and edits files attempting to find artifacts which are created, modified or deleted during the same time frame, but are not symbolically or semantically linked, is an inaccurate methodology when concerning in-vivo or recently in-vitro systems. The scripted application instead relies on symbolically/semantically related files which contain information related to the operation of the network attack tools specified by the user or by using the inbuilt default. Recently edited/created files are still examined, except without the main premise that their identification will be based around file time stamps, which are easily modified. Therefore, rather than identifying files/artifacts which have been marked as created or modified, the basis of the local search is founded instead on the semantic/symbolic link between the file or its content and the network tool(s) specified as the search query. The operation of the tool can be further customized by editing the symbolic word lists. These word lists are part of the program, and offer an internal way to customize the results seen. By using UNIX based text editors, these word lists can be customized and thus easily changed.

There are four different categories of result within the filtered data set. These four categories comprise: OK [Safe], Warnings [Suspicious], Investigate [Malicious], and False [Incorrect]. The respective category relevancy ratings are shown below, where x is the number of matched keywords:

$$(x \leq 2), (x > 2 \leftrightarrow x < 4), (x \geq 4), (false \rightarrow > 0) \qquad (1)$$



Check if x is <=2
NO
Check if x is >2 && <4
NO
Check if x is >=4
NO
Breakpoint (Error Code 1)
Check if false else log item to Safe results list
Check if false else log item to Warning results list
Check if false else log item to Investigate results list

Fig. 1. HexaFind Logic Engine Diagram

The above ratings (1) are based on the inclusion of certain symbolic words – such as "**port**", "**attack**" or "**scan**". This method of analysis also tests for false positives, ensuring incorrect results are ignored. All keywords used by the scripted application can easily be edited. The results are based on a ratings value system which uses key/ symbolic words. For example, an integer is incremented for each additional key/ symbolic word that is present within the result, see Fig 1. Therefore the result in this case would return a ratings value of three, for the three symbolic words mentioned above. These are derived as follows, with the bold text indicating the specified key/ symbolic word**:-**

"/./tmp/logs/attack.txt:nmap **attack** log. 64 hosts **scan**ned. 443/tcp **port** open, vulnerable"

The logic engine used is proprietary. An argument is provided by the user (the query) for all known tools. This can be the name of a specific attack tool such as "nmap", or a null string (""). If a null string is used then all the tools specified in the default attack list will be included in the search. The unfiltered results are iteratively held and processed in a "for loop". The loop continues until all results have been processed.

If the string is not null, then the entries within the unfiltered results file are processed against the contents of the current attack list. For each matching term in the current attack list, a relevancy rating of one star (+1) is added to the individual result. After this first processing stage has passed, the partially filtered result is compared against the keywords list, irrespective of a null query. The second stage of processing adds a relevancy rating of one star (+1) for each matching term in the keywords list.

The third stage is testing for false positives. The result is compared to a false positive file, whereby specific terms are set, and any results which match have their "false" variable incremented and are subsequently removed from the filtered, processed, dataset, regardless of the relevancy rating.

The fourth stage is classification. Here, results are classified into their respective categories and logged to the corresponding results file. This operation continues until all results have been processed. This also tests for false positives, allowing for incorrect results to be ignored. All symbolic words used by the scripted application can easily be edited by the user. These results and their categories are then calculated as a percentage of the total. The percentages are then used to dynamically create the 3D pie chart via the amended 3rd party script [4]. When all the results are collated, the investigator is presented with the option of choosing between HTML or text output and the report file is dynamically generated.

3 Overview of the Application

The scripted application makes most use of inbuilt Linux and UNIX based command line packages in order to retrieve the results. This big data set is achieved through the use of regular expressions and widely available fundamental packages. The application HexaFind comprises five modules which are shown in Fig. 2.

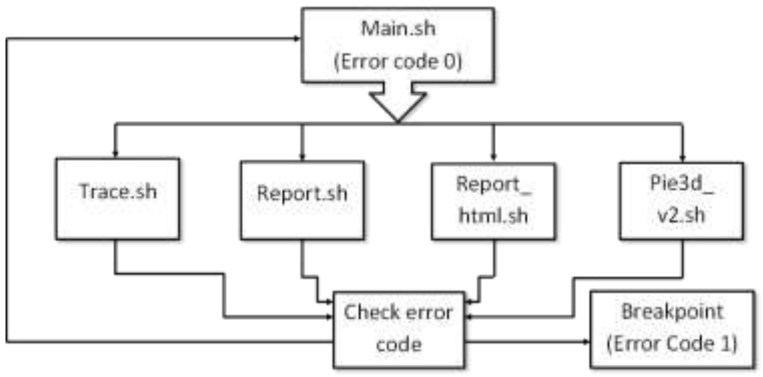


Fig. 2. HexaFind Module Design Diagram

- Main.sh – The main program module is the textual user interface. This module runs the other scripted modules iteratively with the user specified mode of operation, until they return an error code other than zero. If the error code is equal to zero, the main module returns to the start of execution, and waits for user input.

- Trace.sh – The forensic fingerprinting module searches and enumerates the contents of the local hard drive in order to identify the attack traces. This provides unfiltered results, which are then converted, filtered and classified within this module.
- Report.sh – This is the basic text reporting module, originally designed to be portable, which runs automatically. This module offers a textual output, which is similar to that displayed through the console to the user, making it useful for command line only deployments which have no access to a graphical user interface.
- Report_html.sh – The HTML reporting module (Report_html.sh) can also be run to create a HTML formatted output. It runs the 3rd party 3D pie chart scripted module in order to simplify and visualize the dataset.
- Pie3d_v2.sh – A 3rd party 3D pie chart creation script [5] has been modified to enable the investigator to visualize the results. This is run by the HTML reporting module, but can be run independently. However, data is required to first be placed into a structured text file in order to define the visual layout of the 3D pie chart.

4 Results from HexaFind

A number of tests were performed by attacking the server using commonly available attack tools (Metasploit, Wireshark, Nmap, tcpdump and LOIC). The results from HexaFind are presented below.

```
[==========HexaFind:Trace Result #44=========]
/./tmp/msfe-nmap20130102-30793-1s8ajiu:</nmaprun>
Current Tool: nmap (1)
Current Keyword: msfe (1)
Current result:45 has a relevancy rating of [**] 2 stars
                                                           [ OK ]
```

Fig. 3. An OK [Safe] Result

Fig. 3 shows an OK result, as it has only achieved a relevancy rating of 2 stars. This therefore this will be classified as a safe result [OK]. The result shown details the use of the Metasploit Framework Environment or “msfe” used in conjunction with Nmap.

```
[==========HexaFind:Trace Result #85=========]
/./tmp/bitrock_installer_13595.log:Uninstalling                  /opt/metasploit-
4.5.0/common/share/nmap/nselib/data/snmpcommunities.lst...
Current Tool: nmap (1)
Current Keyword: snmp (1)
Current Keyword: log (2)
Current result:86 has a relevancy rating of [***] 3 stars
                                                           [ WARNING ]
```

Fig. 4. A Warning [Suspicious] Result

A warning is shown in Fig. 4, which is the result of one matching attack tool ("nmap") and two matching keywords ("snmp" and "log") being filtered from the processed string.

```
[==========HexaFind:Trace Result #43=========]
/./tmp/msfe-nmap20130102-30793-1s8ajiu:<runstats><finished       time="1357136502"
timestr="Wed Jan  2 14:21:42 2013" elapsed="188.73" summary="Nmap done at Wed Jan
2  14:21:42  2013;  1  IP  address  (1  host  up)  scanned  in  188.73  seconds"  ex-
it="success"/><hosts up="1" down="0" total="1"/>
Current Tool: nmap (1)
Current Keyword: ip (1)
Current Keyword: msfe (2)
Current Keyword: host (3)
Current Keyword: scan (4)
Current result:44 has a relevancy rating of [*****] 5 stars
                                                            [ INVESTIGATE ]
```

Fig. 5. An Investigate [Malicious] Result

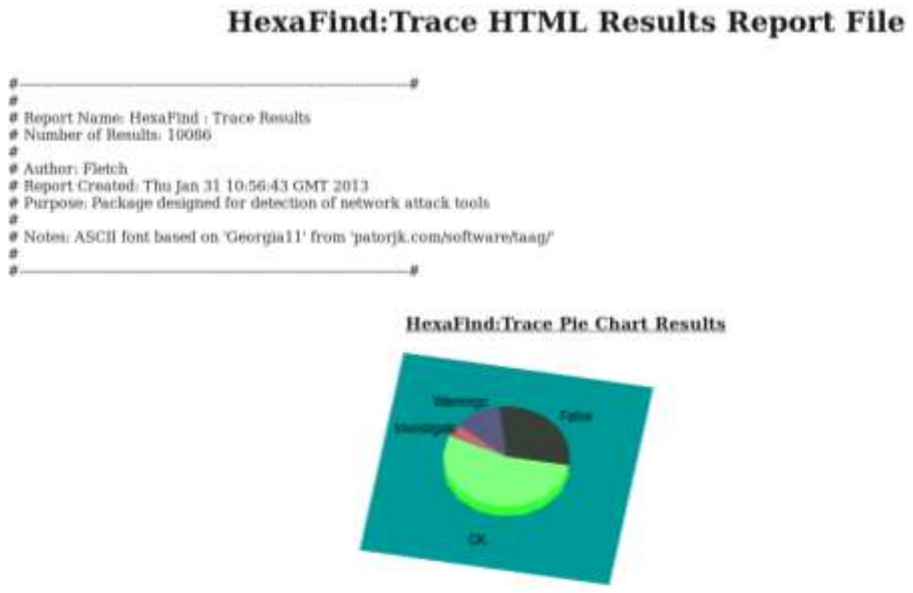
Fig. 5 shows a malicious result that has been detected due to the query specified (attack tool name = nmap), in addition to the four separate keywords that have been found, which are "ip", "msfe", "host" and "scan".

```
[==========HexaFind:Trace Result #746=========]
/./opt/metasploit-4.5.0/common/lib/python2.5/site-
packages/zenmapGUI/TopologyPage.py:# ***********************IMPORTANT NMAP LICENSE
TERMS************************
Current Tool: nmap (1)
Current Keyword: port (1)
Current Keyword: log (2)
Current result:747 has a relevancy rating of [***] 3 stars
                                                            [ FALSE POSITIVE ]
```

Fig. 6. A False Positive [Incorrect] Result

Fig. 6 illustrates an incorrect result, due to the term ".py**:**" being specified within the false positive input file. The string ".py**:**" indicates the presence of a Python script, and so this was removed, as large numbers of false positive results were being generated. If identified as a false positive, then the current result is appended to the false positive list, overriding any previous decisions made.

A sample section of the final report is shown in Fig. 7. As can be seen, the details of the investigator, with dates and times, are included, along with the total number of results analysed and a dynamically generated 3D pie chart of the attacks identified. This gives a high level overview of the current report and its findings. The HTML report is easier for the investigator to read.



HexaFind:Trace HTML Results Report File

```
#------------------------------------------------------#
#
# Report Name: HexaFind : Trace Results
# Number of Results: 10086
#
# Author: Fletch
# Report Created: Thu Jan 31 10:56:43 GMT 2013
# Purpose: Package designed for detection of network attack tools
#
# Notes: ASCII font based on 'Georgia11' from 'patorjk.com/software/taag/'
#
#------------------------------------------------------#
```

HexaFind:Trace Pie Chart Results

Fig. 7. HexaFind HTML Report

5 Discussion

Often attackers do not think about the long term consequences of their actions and the possible repercussions. Although those that do will often take precautionary measures in order to hide files from view or remove them entirely. This is due to the datum that the implementation of relatively simple techniques could ultimately be able to hide relevant important facts from system administrators or forensic investigators. Even being able to replace keywords in a system would cause problems for this project, as most of the algorithms that are implemented are based on information that is stored textually throughout the system. If the attacker is technically skilled they may be able to alter critical evidential information that is held within the system. Log items can be extremely helpful and they are often the only piece of information which connects an attack to a computer or user. If the log files are successfully erased from the computer, then it will make the investigator's job of proving that an attack was initiated near impossible. Therefore any small changes that occur to a system must also be accounted for. Even the smallest detail may assist in convicting a suspect of tampering with log files or system files.

Using shell scripts, and particularly Bash, makes the system highly portable, as the scripts do not need to be compiled for each heterogeneous system environment.

The big data sets used were often over one hundred times the physical size of the original application, and therefore a key requirement was efficiency. Another issue encountered was being able to accurately process each individual result and pipe the data stream to the other scripted applications to effectively process the unfiltered original results.

6 Conclusion

HexaFind is a scripted tool that can assist investigators in aggregating and elucidating key information from a big data set. The objective was to develop a systematic methodology for identifying offensive digital attack tools and the digital artifacts that they leave when they are run on a Linux system. This can be used to successfully prove that a particular attack tool had been used. Working with Linux/ Unix POSIX systems has provided an extremely flexible environment to work in. This also meant that the application could be used across a wide variety of distributions without changes being required for each different system. A future development will be to expand the number of attack tools included within the tool. HexaFind is distributed through the use of a dedicated development website, which can be found at http://www.hexafind.com/. Since opening it has consistently received over 1,500 hits per week.

References

1. Hargreaves, C., Patterson, J.: An automated timeline reconstruction approach for digital forensic investigations. In The Proceedings of the Twelfth Annual DFRWS Conference, Digital Investigation, vol. 9, pp S69–S79, Elsevier, (2012)
2. Zhang, J., Wang L.: An Integrated Open Forensic Environment for Digital Evidence Investigation, Wuhan University Journal of Natural Sciences 2012, vol.17, No.6, pp. 511-515 (2012)
3. Log2timeline, http://log2timeline.net (2011)
4. Hranicky, J., Fsaudit, http://www.cise.ufl.edu/~jfh/jst
5. Gnuplot tricks: Another simple 3D pie chart with gnuplot, http://gnuplot-tricks.blogspot.co.uk/2009/05/another-simple-3d-pie-chart-with.html