NTNU
Norwegian University of
Science and Technology

The study of keyword search in open source search engines and digital forensics tools with respect to the needs of cyber crime investigations

Joachim Hansen

Master in Information Security
Submission date: December 2017
Supervisor: Katrin Franke, IIK
Co-supervisor: Andrii Shalaginov, IIK
Kyle Porter, IIK

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

The study of keyword search in open source search engines and digital forensics tools with respect to the needs of cyber crime investigations

Joachim Hansen

NTNU
Norwegian University of
Science and Technology

MIS4900 Master's Thesis - Information Security
NTNU in Gjøvik, 2017

Department of Information Security and Communication Technology
Faculty of Information Technology and Electrical Engineering
NTNU in Gjøvik
PO box 191
NO-2802 Gjøvik, Norway

Abstract

This master thesis consists of both a theoretical and practical part. In the theoretical part of the thesis are three main parts of study: 1) Exploration of experimental search methodologies used in a Digital forensics setting. 2) Analysis of the differences in documented search capabilities between a set of open source search engines and open source forensics tools capable of keyword search. 3) Identified and summarized publicly available Digital forensic related datasets.

For the first area of exploration no surveys published in the period 2014-2017 could be found. Therefore, this exploration tackles a missing gap in the current knowledge.

The second exploration creates an in-depth and up-to-date analysis of differences in search capabilities, not found anywhere else. This analysis is useful for forensic examiners and researcher that want to know which application is most suitable for their problem domain.

The third exploration extends previous lists of its kind, and adds many new unlisted forensic related datasets. This list, is to the best of my knowledge, the largest collection, of publicly forensic related datasets published in any paper. This addition in the paper will be useful for researchers in many subfields of Information security who are looking for a dataset to use in their research. Using publicly available datasets will also make their experiments more reproducible.

Some of the datasets are also used in the practical part of the thesis. The practical part is a benchmark experiment where the open source search engines are tested on how well they perform at indexing, searching and memory performance during searching. *Elasticsearch* was generally better then Solr at index creation time, minimizing index size and response time for the first run of search terms. *Solr* outperformed *Elasticsearch* on second run of search terms. The difference between the search engines with regard to memory performance during searching was negligible.

There are two main limitations with the experiment. The first being that the experiments are performed on only one virtual host machine. This environment does not allow testing for how well the search engines perform at distributed search. The second main issue is that only the default configurations was tested (out-of-the-box setup) with *Solr* and *Elasticsearch*. If more configurations had been tested, then some of the variables such as sharding and segment count could be controlled. Up-to-date experiments with the same testing methodology could not be found. The experiments provide information that is useful for forensic examiners when deciding which search engine is best suitable for their forensics tasks.

Contents

List of Figures

List of Tables

Listings

Abbreviations

AF	Anti Forensics
DFI	Digital Forensic Investigation
FP	Forensic Practitioner
IR	Information Retrieval
TC	Time Complexity
MC	Memory Complexity
MVT	Memory Visualisation Tool
RBT	Red Black Tree
SE	Search Engine

1 Introduction

The purpose of this chapter is to present the reader with the topic of the thesis, the problem description, justification for doing the research, the research questions that will guide the research, and the planned contributions of the research.

1.1 Topic covered by the project

Digital forensics investigations have to deal with a digital landscape where the amount of data increases in volume each year [1]. The Big Data problem introduces problems such as how can forensic practitioners (FP) process the data collected in their investigation in a reasonable amount of time and figuring out how to best handle the storage requirement of the data. Using relational databases to process, the data is not appropriate as the largest portion of the data is unstructured [2].

Information retrieval systems like search engines (SE) have been used to help locate enterprise data. SE used in enterprises also have to deal with large volumes of heterogeneous data [3].

This master thesis aims to evaluate the performance of search engines and search engine functionality on forensic data.

1.2 Keywords

Digital forensics, search engines, benchmarking, forensics tools, forensics datasets, and open source tools.

1.3 Problem description

Forensic practitioners in digital forensics has to process large quantity of structured and unstructured data. The processing of data has to be reliable, forensically sound and ideally be solved using an algorithm with a low memory usage and time complexity. Forensic practitioners can use one of many Search Engines (SE) to aid them on this task.

Information regarding which SE are available and how their search and indexing capability compares to one another with regard to both features and performance, is valuable for forensic practitioners to decide if the given search engine is the best fit for their application.

1.4 Justification, Motivation and benefits

Digital forensic investigations have a Big Data problem. Without tools that can search the data within a small time frame and provide relevant results, forensic investigation cannot examine the evidence effectively. This can in turn negatively affect the justice system capability of convicting criminals.

The ability for search engines to process large amount of digital forensic data will be evaluated in this project. The benchmark can provide relevant information needed to determine whether to invest resources on integrating search engines into the digital forensic investigation process.

Throughout the process we have gotten a strong impression that there is a lack of:

- Recent surveys on how search is used in digital forensic investigations.
- Recent in depth comparison of search capabilities between search engines and forensics tools.
- Recent surveys of Digital Forensics related datasets that can be used in forensic research.
- Comparison of performance between search engines using the same publicly available datasets for benchmarking.

By contributing to the above list, forensic practitioners will have better information to make conscious decision on which SE that best aid them on the forensic process. In addition, the usage of publicly available forensic datasets will make the experiments easily reproducible for all who wish to replicate the results.

1.5 Research questions

This paper addresses two main research questions:

1. What is the state of the art of keyword search in digital forensic?
   (a) How can search be applied in a digital forensic setting?
   (b) What are the open source search engines that are still in development?
   (c) What are the open source forensic tools capable of search that are still in development?
   (d) What are the advertised search capabilities/features of the open source forensic tools and open source search engines?
2. What open source search engines performs best with keyword searches, when using the same terms and forensic related datasets?
   (a) What are the publicly available forensic related datasets for testing?
   (b) Which search engine performs the most efficient indexing with respect to the resulting index size and index creation time?
   (c) What is the best search engine with respect to minimizing search time and the number of clear cut misses on searches with single, multi and non-existing search terms?
   (d) Which search engine used the least memory during search?

Research questions 2a, 2b and 2c also have the null hypothesis H0:
There are no significant differences in performance from one search engine to another.

1.6 Contributions

Theoretical novelty

The thesis summarizes the experimental usage of search in a Digital Forensic setting from papers published in the period 2014-2017. The time period was chosen as to prevent adding outdated forensics practices. No surveys in this publication time could be identified: this makes analysis and synthesis important.

A list of open source search engines and open source forensic tools capable of keyword search.

This paper also presents a in depth comparison of a set of open source search engines and forensic tool with respect to documented search capabilities. Similar up to date and in depth comparisons of this kind could not be found.

The last theoretical contribution is an extensive list of summarized publicly available forensic datasets. This list is not one of its kind, but extends on previous works and add

many datasets that where not listed before. To the best of my knowledge, the summary is the largest list of forensic related datasets presented in any reviewed relevant papers. The novelty of this list is the number of diverse datsets and the summery.

Practical novelty

The novelty of the experiments is the **combinations** of the elements in the list below:

1. Usage of a diverse set of publicly available forensic related datasets. The datasets provides dataset sizes and content that is relevant to forensic practitioners. Moreover, the usage of publicly available datasets enables reproducible benchmark.
2. The benchmarking experiments on the open source search engine *Solr* and *Elasticsearch* where all performed on the same virtual machine. The search engines were benchmarked using the same datasets, similar preprocessing steps, same amount of documents indexed at a time, uploading data to the search engine using the Curl command and using the same measurements.
3. The benchmarking of *Solr* and *Elasticsearch* were done with out-of-the-box configurations (default configurations)

1.7 Choice of methods and outline

This thesis uses a combination of quantitative and qualitative methods to answer the theoretical and practical research questions in this paper. The research questions are answered by performing three systematic literature reviews and one experiment.

The reviews are in their own separate chapters (see chapter 2 - 4) with their scope/-goal, research questions, methodology, and results. The practical research is described in detail in chapter 5 and 6.

Outline:

- The 2nd chapter shows the usage of search in recently published papers on Digital Forensics, and provides a list of maintained open source search engines.
- The 3rd chapter first shows the reader with a list of open source forensic tools capable of search, and then provides an in depth comparison of the documented search capabilities of a set of open source search engines and open source forensic tools.
- The 4th chapter summarizes a large list of publicly available forensics related datasets.
- The 5th chapter describes how the experiments where performed.
- The 6th chapter shows the results and analysis of findings.
- The 7th chapter provides the conclusion, discussion on findings, theoretical and practical implications, discussion on reliability and validity of results, limitations and possibilities for future work.

2 Systematic literature review 1 - Search in digital forensics and Open source search engines in the wild

2.1 Outline

The literature review is divided up in the following subsections:

1. Application of search in digital forensic investigation: A review on the literature for the last 5 years on how search can be applied to digital forensic investigations. This section is further divided into collection, examination and analysis; are phases in the digital forensic investigation process model discussed in [4]. **There seem to be a lack of recent surveys on the topic of usage of search in digital forensic. Many of the paper used in this systematic literature review are using search with experimental methodologies in a forensic setting**
2. Search engines: A overview of the search capabilities for a number of search engines that are open source, recently in development and that are not primarily web search engines.
3. Search utility: A look into the utility of the search engines search functionality.
4. How search engines should perform in a digital forensic domain

Research question

1.a) How can search be applied in a digital forensic setting?
1.b) What are the open source search engines that are still in development?

2.2 Application/experimental use of search in digital forensic investigation

Collection phase

Privacy law can regulate what method FP can use when collecting evidence. One paper [5] created a privacy protected scheme, where FP can perform a keyword search on encrypted emails. The individual emails could only be decrypted if the amount of exact matching non-blacklisted keywords provided by the FP are equal or above a certain threshold. Blacklisting or whitelisting certain keywords can make it harder for an attacker to perform a dictionary attack.

The paper by [6] argued that volume information found in the open source distributed file system platform XtreemFS is of interest to FP. The information can be used to search to find particular volumes of interest and the size of the volumes to determine if acquisition is practical. FP can search for the string "xtreemfs@" to find out if a node is connected to XtreemFS.

Evidence amongst junk:

Email spam folders are often overlooked by FP as they mostly consist of junk [7]. Criminals can craft their messages in such a way that it will be picked up by the spam filter and hide their activities from law enforcement. Keyword searches and manual review of the spam emails is therefore important to find obfuscated evidence. The folder could be a way for criminals to obfuscate their activities, and should therefore be collected and searched.

Examination phase

Searchable hash databases:

It was claimed in [8] that it is commonplace for Forensics Practitioners (FP) to maintain a database of hashes of know illegal images and videos. FP can hash media collected in an investigation and search the database for matches. This approach has obvious limitation against anti forensics (AF) approaches such as resizing of the images. To improve upon this scheme the paper creates a custom database called *hashdb*, that stores hashes of the individual data blocks of files. This solution is more resistance against small file modification, as many of the data blocks would remain unchanged. Searching the database for matches of crime media can return a single match or a candidate list.

Searchable reference database

One study [9] showed that usernames and passwords found in computer memory could be used to identify which websites the credentials belong to. A search condition like "&Email" and "&Passwd" can be used to search for usernames and passwords in memory. Some usernames and passwords that belongs to particular websites can be retrieved with a unique search pattern, others can be found by using the same search condition. The non-unique search conditions can use the session component found in memory to uniquely identify the website. Having a reference database for this mapping can be useful for forensics examiners that want to understand suspect activity online. Maintaining the reference database beyond the most common websites would be impractical.

Approximate hash based matching:

While not being widely adopted by the digital forensics community, approximate matching can be used to detect semantically and syntactical similar files and match it against a reference dataset [10]. Semantically similar files are files such as images that look alike in the eyes of humans. For example, otherwise identical images, one in white and black and the other in color would be perceptually the same file. The application of searching for semantically similar files can aid FP to find the origin of files of interest. Syntactical similar files are files that look similar on the byte level. Approximate hash based matching (AHBM) is not appropriate for images as they can look the same, but have different encodings. AHBM are also well suited for dealing with unstructured data such as text files, memory dumps and fragmented files. The paper concludes that the same results can be accomplished with string search as with approximate matching, but this would require far more from the FP.

Inexact search

One paper [11] created a search algorithm called *ScalClone* that aims to find exact and inexact code fragments between analyzed and un-analyzed malicious assembly files. Exact fragments are identified by searching for regions with the same hash value. Inexact fragments are fragments that share many mnemonics and operand types. They are identified by first constructing a binary vector with respect to feature frequency and features mean value, and then comparing the co-occurrences of the fragments. If the co-occurrences count is greater or equal to the similarity threshold, then the fragment is considered a inexact clone. Inexact search is not affected by reordering, as the frequency of the mnemonics remains unchanged. Obfuscation by adding do-nothing instruction drops the recall rate to 90% and compiler optimization drops it to 62%.

Deduplication:

One issue with collected forensic images of storage devices, such as hard disk drive (HDD) is duplicated files [12]. Processing duplicated files leads to unnecessary overhead in the examination phase. One way to solve this issue is by arranging the files in a red black tree structure (RBT). Duplicate nodes in this structure can be found by searching using *wildcards*. After identifying duplicate nodes, their child nodes will be rearranged in the tree and then the duplicate node will be removed from the structure. The time complexity for searching, inserting and removing nodes in RBT is $O(\log_2(n))$ for the average and worst case. This proposed solution does not state in detail how their scheme identifies files with the same content. While identifying the same file names using *wildcard* seems reasonable, hash matching is more appropriate for telling if two files have identical content.

A proposal was made in [13] to identify duplicate images where the file name, file extension or file attributes (e.g hidden, compressed, encrypted and protected Operating System File) did not match the source image. The proposal used the source modified timestamp to search for duplicate files. 1000 files spread across 30 folders totaling 3.09 GB in size was processed in 1 minute and 32 seconds. The same files spread across 300 folders took 16 minutes 23 seconds longer to process. Therefore, its application is limited to environments with a small number of folders. The proposal is also vulnerable to tampering done to the modified timestamp attribute.

Examination limited by law:

According to [14] the United State Supreme Court is beginning to demand that the examination process is limited in its scope. This means that the goals and objectives must be clearly stated, as well as a justification for what the examiner will search for, and the boundary of the search. Failure to comply could negatively affect their case in court. This restriction might force a better resource management of the examiner resources. However, it can also make it more difficult to examine evidence that is hidden in unusual locations, as its examination would be difficult to justify. Simply searching for everything in a Gigabytes or Terabytes search space would not solve the problem as this task is infeasible even when using common digital forensics tools or automated tools [15, 16]. The courts also put constraints on how long seized data can be processed by the examiner, before it is returned to its owner [17]. It is argued in [18] that the searching by the examiner, can be aborted after the most probable places have been processed. More specific search criteria can reduce privacy violations and reduce number of false positive hits [19]. The question then arises how specific can you be before negativity influencing the recall rate.

RAM search

A survey [20] stated that string search in volatile memory examination is useful in order to find residue of user activity, passwords, encryption keys and side effects of malicious scripts. Searching in swapped out memory pages in *Windows* can potentially provide evidence of old user activity, as the swapped file is often not cleared after system reboot [21].

Pool tag scanning is a type of exhaustive search on volatile memory that is used to find data structures such as direct kernel object manipulation (DKOM) which is used by malware to hide processes [22]. The study [22] stated that exhaustive search might not

be appropriate for time sensitive investigations. They therefore created pool tag quick scanning, which reduces the search space to memory pages related to pool allocations. The search space reduction can be "multiple orders of magnitude" and the accuracy of the search results remains high.

The use of visualization to aid search:

A comparison was done in [23] to test the accuracy and speed of which experienced participants in networking, Windows operating system, malware and incident response, are to solve forensics tasks. The participants where given the same tasks and the same forensics image. They were split into two groups, one that used normal text search and the other that searched using a memory visualization tool (MVT). The MVT showed relationships between the data and had a whitelisting algorithm that removed known good files from the search space. The results showed that the participants that used the MVT completed the tasks faster and more accurate. I infer from the text that the number of participants is 10 (minus one outlier). Laying too much weight from the results on this low sample size might not be appropriate.

Issues with keyword searches

The study [24] compared the state of the system before and after forensics examination using the following bootable forensics environments: *Knoppix v7.0, Helix 3 Pro 2009R3 and Kali Linux v1.0*. Keyword searches were used during the examination process to simulate an investigation. The hash value taken on the forensics image before and after examination, did not match in any case. It was mainly the “last accessed” timestamps on files that was altered after the examination. Performing keyword searches in those environments can therefore be problematic in cases where establishing a timeline is important.

It is argued in [25] that keyword searches resulting in large number of false positive hits, can be reduced by using background knowledge from the investigation.Also while keyword search algorithm are useful, they are inept at processing terabytes of data. An alternative to keyword search is *Fuzzy search/fuzzy matching*. This search can be used to find elements missed by the normal keyword search such as misspelled words and slang terms. [15].

The use of clustering to improve the usefulness of search and suggestion in search:

One study [16] used keywords search terms to cluster forensic data to reduce examination overhead. There is one cluster per search term. In order to help the examiner choose good search terms, the system returns the most frequent used search terms found in the forensics data. Both with and without suggestions, the system performs well with respect to average precision and recall. The system is also scalable as the runtime grows linearly with the number of documents.

FP has to search though large volumes of heterogeneous data. One study [26] evaluated the performance of clustering techniques on a forensic dataset containing 2640681 search hits. They achieved an precision improvement of a factor 15 over non-clustering and a overall average precision of 67%.

Analysis

Orphan Files - deleted files

Finding evidence of deletion of user activity on the suspect machine is of interest of FP [27]. Searching the Update Sequence Number (USN) Journal file on the NTFS can reveal when and where files have been created, viewed, renamed, moved or deleted. Another study [28] showed how searching for the string 'for deletion' in a Hadoop Distributed File System (HDFS) is useful to find evidence of deleted files. The paper [29] claimed that only the row directory is overwritten with a NULL value when a row is deleted in the database DB2 or SQL server. This allows a FP to search these databases for the deleted rows and restore them by considering the valid row directory values of their previous and following row directory entry.

One study [30] mined 1100 chat logs to find the most significant terms, users and chat sessions. Two bigraphs are constructed. The mapping in the first bigraph is such that we can observe which term (Hub) has been said by which users (Authorities) and what terms (Hubs) have been said by a user (Authority). The second bigraph has similar mapping, but the Hub is the term and the authority is the chat session. A self-customized hyperlink-induced topic search (HITS) algorithm is used to iteratively set the Authority and Hub score. A selection of the highest scoring users, chat sessions and terms are used together with user metadata and session metadata to construct a social graph. Clustering is applied on the social graph to find shared interest and interactions between users.

One study [31] showed how traces found from volatile memory in IEEE 802.11 wireless devices, that is in radio range from each other can answer important forensics questions like Who, When and Where. There are two types of broadcast traffic frames that can answer these questions. As their format is known, they can easily be found by using regular expression search. The probability that the frames are still in the devices volatile memory depends on external and internal conditions like the extent and nature of the broadcast traffic processed by the device and the configurations of the device. This methodology would therefore only work in a few real life scenarios and mostly in non-urban areas.

File carving:

Search helps file carving tools identify header, footer and fragments used to identify where a file begins and end and use this information to restore the file [32]. Some file carving tools are able to restore files independent on the underlying file system. Exhaustive search can be used to find each combination of header and footer of a video and then try to validate/decode on the restored file to see if it is a valid video. Search can be used to find the order of the fragments and codecs search codes to identify fragments belonging to videos.

Encoding:

The FP may encounter digital environments where the binary data is encoded using multiple different UNICODE encodings and that the type of UNICODE are unknown [33]. The share number of possible UNICODE encodings means that the same text can be represented in many different ways. Resolving the underlying encoding in the worst case can require number of search passes equal to the number of possible encodings. The average case is much better as many encodings are not widely used. The regular expression search engine *lightgrep* aims to deal with the encoding problem. Lightgrep uses

UNICODE characters as string literals in the regex expression to be encoding independent. For handling the encoding Lightgrep uses multi pattern search enabling it to search for multiple encodings in parallel. The search engine currently supports 180 encodings making it possible to perform UNICODE-aware searches.

2.2.1 Open source Search engines in the wild

In table 1 is a collection of open source search engines still in development. The columns of this table are explained in the list below. The update column is the last advertised change to the software from the point of this review.

- S_1 = *Full text search*
- S_2 = *Faceted search*
- S_3 = *Spatial/Geospatial search*
- S_4 = *Fuzzy search*
- S_5 = *Streamed search*
- S_6 = *Phonetic search*
- S_7 = *Semantic search*

Table 1: Open source desktop/intranet search engines and their default search capabilities
Source: [34–63]

Name	S_1	S_2	S_3	S_4	S_5	S_6	S_7	**Update**
Dezi	✓	✓						28.11.2016
Apache Solr	✓	✓	✓	✓	✓			06.03.2017
Sphinx	✓			✓				08.09.2016
Sifaka	✓							25.01.2017
OpenSearchServer	✓	✓	✓			✓		13.01.2017
Luwak					✓			06.03.2017
Datafari	✓	✓						23.03.2017
Elasticsearch	✓	✓	✓	✓		✓		24.04.2017
groonga	✓		✓					24.04.2017
tantivy	✓							23.04.2017
tntsearch	✓		✓	✓				20.04.2017
pouchdb-quick-search	✓							22.02.2017
OpenSemanticSearch	✓	✓		✓			✓	16.04.2017

2.2.2 Search utility - short overview

According to the whitepapers [64], [65] *Full text search (FTS)* is suitable for finding relevant documents in a large set of unstructured data. A lot of the data gathered in a forensic investigation is unstructured [66]. It is more appropriate to use FTS to respond to ad hoc request than requests with a predefined answer [64]. A document in FTS is considered a list of searchable terms (e.g. words and numbers) [65]. The terms are usually indexed in order to make them easier to search.

Faceted search is a way of traversing the corpus based on categories (facet) and subcategories (facet values) [67]. In faceted search, it is possible to find the same the same data points by using different traversal paths. *Faceted search* is useful for exploring the corpus and the facet values aid the searcher to create more precise search phrases. It is common practice in *faceted search systems* that only the most frequent facet values are shown. This makes finding more obscure items difficult.

Fuzzy keyword search retrieves both documents that matches exactly with the search

phrase and those within a similar distance [68]. The distance can be measured by using the Levenshtein distance. Which compares the minimum number of insertions, deletions or substitutions are needed for string A to equal string B. The paper [69] claims that auto completion is helpful when the searcher have do not have sufficient domain knowledge of the dataset he is searching.

Phonetic search is matching based on similar sounding words [70], [55]. One example of a phonetic algorithm is *Soundex*. It encodes a word into a 4 character code starting with the same character as the word [70]. Similar sounding characters like s,f,p and v are represented by the same number. Repeating characters, vowels and certain letters are ignored by the algorithm. Truncation and padding are used to make sure that all words are represented by a 4 character code. The limitation with this approach is that only words starting with the same letter would have a chance to match with the same code. Phonetic algorithms are designed to handle specific languages, making them limited in their utility [55]. The aim of Phonetic search is not improving precision but to increase the recall rate.

Geospatial search is searching a corpus where the documents have associated geographic data such as latitude and longitude. One example of using the location data is to search for registered criminals that lived in the vicinity of a crime scene [71]. It can also be used to find all previous search warrants on an address or all search warrants in some proximity to a given address.

Documents that do not contain the terms of the user query can still be relevant [72]. Classical retrieval based on lexicographic term matching will not retrieve documents that are lexicographically different but semantically similar. To improve information retrieval of documents Semantic search can find semantically similar terms that are often overlooked by using stemmed synonyms or Ontology. Ontology models a domain into concepts, attributes and relations [73].This model provides the semantic reasoning needed to retrieve meaningful documents with respect to the user query [74].

Streamed search was explained in [75, 76]. In traditional full text the documents are often indexed using inverted indexing to optimize the time it takes to find the queried documents. Running all possible queries on the documents works well if the complexity of the queries and the data velocity is low. Network log files are an example of a stream (continuous data flow) where traditional search is impractical. Stream search uses inverted indexes on queries instead of documents. By doing so, it is possible to take the new log entry and query the inverted index to see which indexed queries match the new entry. Now the search has identified the minimum number of queries that need to run on the new entry. This approach could potentially save high amount of computer resources.

2.2.3 How search engines should perform in a digital forensic domain - short overview

The importance of the measurements precision and recall in Information Retrieval (IR) systems, like Search Engine (SE) depends on the application [77]. In the domain of Digital forensic Investigation (DFI) precision is more important in the early phases of the forensic investigation, as relevant evidence is vital to guide the process of finding new evidence. At the later stages of the DFI, recall becomes more significant than precision, as Forensic Practitioner (FP) wants all available evidence to build a court case.

2.2.4 Handling problems

The table 2 show which search phrases and search resources used to collect the sources and the number of resulting hits. Other sources like finding the search engines web pages were found in a snowball fashion. Relevant project links on Sourceforge and Github was used to locate the search engines.

Table 2: How the sources were located

Query	**Search resource**	**Hits**
("Abstract":enterprise AND search AND engine), Year: 2014-2017	ieeexplore.ieee.org	27
"Enterprise search", in abstract, year: 2014-2017, source type: Scholarly Journals	search.proquest.com	24
recordAbstract:(+enterprise +search) , year: 2014-2017	dl.acm.org	44
in abstract (Solr OR ElasticSearch)	arxiv.org	6
in abstract, title and keywords: Enterprise search, year 2014-2017	sciencedirect.com	47
in abstract:Solr OR ElasticSearch, Scholary journals, year:2014-2017	search.proquest.com	47
in abstract (Solr OR ElasticSearch), year:2014-2017	ieeexplore.ieee.org	72
recordAbstract:(Solr Elasticsearch), year:2014-2017	dl.acm.org/	18
information retrieval unstructured data (general search), year:2014-2017	ieeexplore.ieee.org	122
"information retrieval" "unstructured data" survey	scholar.google.no/	2990
(+"Digital forensics" +search) - any fields	dl.acm.org	7
(+"Computer forensics" +search) - any fields	dl.acm.org	7
in journal "Digital Investigation" : search, 2014-2017	sciencedirect.com	161
in book "Digital Forensics Threatscape and Best Practices" - year: 2016 - search phrase: search	sciencedirect.com	10
in publication "IEEE Transactions on Information Forensics and Security", year:2014-2017	ieeexplore.ieee.org	42
basic search " Digital forensics search", year:2014-2017	ieeexplore.ieee.org	65

3 Systematic literature review 2 - Search capability comparison between open source search engines and open source forensic tools

3.1 Purpose of the literature review

Inspect a set of open source search engines and, open source forensic tools that have search functionality, and identify and compare their various search capabilities.

Research questions

1.c) What are the open source forensic tools capable of search, that are still in development?

1.d) What are the advertised search capabilities/features of the open source forensic tools and open source search engines?

3.2 Protocol/methodology

3.2.1 The candidate set of open source search engines

For the candidate list of open source search engines, the table 1 of search engines from review 1 was used. Due to time constraints, only handful of these search engines will be used for further inspection. Therefore in table 3 the search engines have been ranked after a popularity criteria.The goal of the popularity criteria is not to find the best representation of the software popularity, but to get some estimate/rank/criteria that can be used to decide which software to first inspect and perform experiments on. The popularity criteria are made up of two parts. The first part is data from Google trends over the last 12 months for each of the software names in table 3. Trends was calculated on 08-10-2017 with category "computers and electronics". Categories are used to filter out unrelated searches. Then for each software name entered in Google trends, a .csv file was downloaded. The .csv files was opened in Microsoft Excel 2010 editor and the formula

=summer(B4:B55)

was used to calculate the sum estimated number of times the software name was searched during the 12 month period. This number represents a way to quantify user interest for the software. The second part is made up of how many articles between 2016 and 2017 the software name is mentioned alongside the phrase "search engine" in ieeexplore. Example search:

(Search engine) AND Dezi), year: 2016-2017

This value is a way to measure the scientific interest of these applications. The rows of table 3 are then sorted on the sum of these 2 values.

Table 3: A candidate list of open source search engines sorted on sum

Name	Searched by users (12 months)	ieeexplore mentions	Sum/criteria
Elasticsearch	4,649	114	4,763
Solr	3,826	85	3,911
Sphinx	3,718	40	3,758
Dezi	2,936	1	2,937
Luwak	2,438	0	2,438
groonga	2,058	0	2,058
Sifaka	1,669	0	1,669
tntsearch	1,595	0	1,595
Datafari	1,566	0	1,566
OpenSearchServer	1,361	0	1,361
OpenSemanticSearch	679	1	680
tantivy	574	0	574
pouchdb-quick-search	N/A	0	0

3.2.2 The candidate set of open source forensic tools

In the selection process for finding the open source forensic tools to inspect, a different criterion was used. The forensic software had to be open source, must have some search functionality, and have some degree of documentation and preferably still being maintained. The search phrases in the list below were used on 06-10-2017 (DD-MM-YYYY) to scan Google for compiled lists of open source forensic tools:

- "open source forensic tools" "keyword search"
- open source live forensics
- open source forensic string search
- forensic wiki keyword search open source

The resulting compiled lists of forensic tools are [78–84]. From that list a set of forensic tools was selected. The selected forensic tools can be seen in table 4

Table 4: The set of open source forensic tools for inspection
Source: [85–97]

Name	Category	Update
The Sleuth Kit (Autopsy)	Static analysis (forensic image)	15.10.2017
Hachoir	File explorer	13.12.2017
Volitility	RAM searcher	??.12.2016
GRR Rapid Response	Remote live forensics	12.12.2017
TestDisk and PhotoRec	Partition reader and image recovery tool	02.12.2017
bulk_extractor	Gives useful information regarding scanned files	26.06.2017
MIG: Mozilla InvestiGator	Remote live forensics	30.11.2017
guymager	Forensic imager	02.10.2017
Rekall Memory Forensic Framework	RAM searcher	06.12.2017
flare-floss	Malware static analysis	19.09.2017
inVtero.net	Memory dump analyser	28.11.2017
Wireshark	Network traffic analyser	30.11.2017

The inspection set of open source forensic tools and open source search engines

Forensic tools : where selected based on degree of documentation and tool category.

1. Sluth kit - Autopsy
2. Volatility
3. MIG
4. Hachoir
5. Wireshark

Search engines : where selected based on popularity.

1. Elasticsearch
2. Solr
3. Sphinx
4. Dezi

3.2.3 The inspection process

The inspection process will be performed as a combination of targeted manual inspection and keyword searching of technical documentation and source code comments of the software being inspected. Targeted manual inspection is looking at portions of the documentation more likely to include relevant information. Some documentation pages might be very old or indicate that the documented feature is experimental. Some of these sources was excluded from the review. One issue with the inspection process is that inferences often had to be made on out of context images and text that describes the search capabilities. Moreover, the description was often quite short. Ideally, the software capabilities would be confirmed by practical tests, but this may not be feasible due to time constraints.

List of planned to use keywords/things to look after during manual review:

- character/symbol limit (max/min)
- truncation character
- match
- regex
- grep
- Wildcard
- scan
- filter
- sort
- encode/encoding (names of encodings)
- index
- rank/ranking
- string, substring
- name list of string matching algorithms
- keyword
- fuzzy
- preproccesing
- encryption search
- compression of search data (list of compression algorithms)
- search storage (RAM, Cache, disk)
- Terms
- Hits
- multi threading and search

- list of known search capabilities
- plugins, forks, add-on, module
- query
- search
- deduplication/hashing
- Visualization of search results
- Customized search
- Parameters
- algorithms
- What can it search, where does it search
- look at advertised set of features
- Byte level search
- concurrent search
- save search results (file formats)
- Keyword lists
- automate
- Hash search
- File search
- Supported file types
- Unicode searching
- Wizard
- Skip known files when indexing
- Can you search while indexing?
- Exact match and substring match.
- Related pages in the documentation
- truncation search / stemming
- Sdhash, fuzzy hash matching, approximate hash based matching (AHBM)
- Report search
- search tree view
- search partitions, boot sector
- search recovery
- search highlighting
- unallocated storage
- search archived files
- email search
- binary search / sequential search
- Periodical search
- rule/criteria based search
- Stemming
- Phonetic search
- Faceted search
- Semantic search
- fuzzy search
- Exhaustive search
- byte comparison / character comparison
- Boolean search

- Language detection
- structure of the search (e.g. string, JSON object)
- Masked results (e.g. meta data) / obfuscation of sensitive data.
- Stripping - removal of sensitive data
- Clustering of search results
- Scrolling
- Relevance score - (TF/IDF)
- Summary / Aggregated results
- "More like this" quary
- Index and field boosting (weighting)
- boolean operators
- Fixed relevancy score
- export search result

3.2.4 Inspection summary forensic tools

Inspection of Sluthkit - Autopsy

Name: sleuthkit autopsy
Sources: [98–114]
Positive methodology:

- Searches google on 08-10-2017 (DD-MM-YYYY) for:
  - Autopsy navigation tree
  - hash hits autopsy
  - sleuthkit autopsy search
  - keyword search ingest
- Searches google on 09-10-2017 (DD-MM-YYYY) for:
  - sluthkit case insensitive
  - symbol limit sleuthkit
  - EBCDIC encoding slueth-kit
  - sleuthkit keyword
- Looked at advertised features
- Looked at Autopsy User's Guide
  - Manual Analysis
  - Automated Analysis (Modules)
  - Reporting

Inspection Result

- Keyword search:
  - Bases on Apache Solr
  - Support for concurrent search on the same index
  - Do not perform byte level search, but preprocess the documents with Tika and perform the search on the output of this process.
  - Can automate search by creating lists of keywords. Keywords list can be imported/exported.
  - Supports both exact and substring matching. Substring matching is will not work with spaces and punctuation characters.
  - Users can decide between case sensitive and case insensitive search.
  - Can create HTML and Excel reports of search hits
  - Text Content Viewer: provides keyword match highlighting in matched files
  - Periodical searches are supported
- Regular expression:
  - Includes predefined regular expressions for emails, telephone numbers, URL and IP
  - Based on a perl implementation
- Rule based search allows to set a search criteria such as the file name, if files or directory should be searched, file extension, directory path
- Index:
  - First strings are searched to be extracted and then index. The default settings are english and UTF8 and UTF16. But these settings can be changed. UTF16LE and UTF16BE are also supported for string extraction.
  - EBCDIC are **NOT** supported.
  - Searches are done on indexes
  - There is a option to skip known files (HASH) when indexing (deduplication).
  - Can search while indexing (index is incomplete)
- Supported files:
  - RTF
  - PDF
  - HTML

  - DOC, DOCX
- File/directory search:
  - can search on file name
  - Can provide a date interval [from,to] and match the files that has their file attributes modified, accessed, changed or created file attributes within this interval.
  - Can search for files that matches a size criteria
  - Can search for known bad hashes, known hashes and unknown hashes. This is made possible with a hash database. Hash lookups is done by binary search.
  - Can search for deleted files.
  - can search unallocated storage
  - can search archived files
- Can search for partitions and boot sector
- Search results:
  - Store results as XML or HTML?
- 3rd party software/plugins/modules with search capability:
  - Approximate Hash Based Matching (AHBM) or fuzzy hash matching with sdhash
  - Reference database: VirusTotalOnlineChecker is a addon to Autopsy and can check the hash of files against the VirusTotal Database.
  - PTK is a addon for sluethkit that allows keyword search of memory dumps.

Inspection of Volatility

Name: Volatility
Sources: [115–125]
Positive methodology:

- Looking for the word scan in the "doxygen-generated" manual
- Searched google on 11-10-2017 (DD-MM-YYYY) for
  - BaseScanner volatility
  - volitility framework regex
- search Volatility github repository on 11-10-2017 (DD-MM-YYYY) for
  - search
- Looking at the Command-Reference-Mal in github repository for Volatility

Inspection Result

- RAM scans:
  - PSDispScan searched physical RAM for _EPROCESS data structures. This can reveal information of killed and hidden processes.
  - MultiPoolScanner can find many different types of pooltags using different pool tag scanners.
  - BaseScanner is a type of exhaustive search that process one and one byte, it is a more general scanner.
  - malfind: Looks for malware in processes by searching for VAD tags and permissions.
  - kdbgscan and kpcrscan searches for values that look like kdbg abd kpcr values
  - Have scans for tcp connections, files, synlinks, drivers, sockets etc.
- Regular expression:
  - Yarascan allows making search based on Regular expressions, wildcards or **YARA rules**. Search by yarascan can be made **case insensitive**
- The find_module function uses **binary search on sorted input** to find the mapping between modules and virtual memory.
- Evolve is a interface of Volatility that can be run in network browser. Evolve enables SQL queries to be performed on the data from the scans, and searching the table view of the results.
- The html renderer allows you to view the output in a browser in a table view and sort by any field or search the output.

Inspection of Mozilla InvestiGator

Name MIG: Mozilla InvestiGator
Sources: [126–129]
Positive methodology:

- Looked though:
  - Mozilla InvestiGator: File module
  - "Concepts & Internal Components" documentation.
  - Mozilla InvestiGator: Memory module
- searched on MIG github repository on 11-10-2017 (DD-MM-YYYY) for
  - search

Inspection Result

- Can use a target field to search for certain "agents", which are a software that interface between MIG and the remote host.
- Support for regular expressions.
  - For file search there is a option for setting a criteria that all records within a file need to match a regular expression.
  - regexes support UTF-8 encoding
- Searches are structured as a JSON object
- Have a non default option of getting masked meta data from searches.
- Does not follow directory links as this can lead to loops.
- can search for files with content that match a MD5, SHA1, SHA2 or SHA3 hash
- search filters:
  - Can filter search on files based on name, extension, size, set of permissions for the file and the modification time attribute. Multiple filters can be applied for the same search (form of boolean search). The default behaviour is that not all filters have to match, but this can be changed.
  - A option for retrieving all files that did not match the filters
  - Can limit the number of search hits and constrain the search to only process directories on certain hierarchical levels (will not process a subdirectory that is further down the hierarchy than the limit).
- RAM search:
  - Sequential search can be performed on multiple buffered memory regions. Have the option to jump over memory or terminate the search after x number of characters have been read.
  - can search for matching bytes strings

Inspection of Hachoir

Name: Hachoir
Sources: [130–133]
Positive methodology:

- Looked though documentation pages:
  - hachoir-metadata program
  - hachoir-strip program
  - hachoir-grep program
  - hachoir-subfile program
  - Hachoir3 for developers

Inspection Result

- Uses the Lucene library for fulltext search
- Can search for images in general or for specific image filetype. Searches can jump to read from a given byte point and files size can be used as a search criteria.
- The program has a module called grep that works with string data, it can print all strings in a file, search for case sensitive, or case insensitive strings in file. This functionality comes with a high memory cost.
- Have the ability to remove sensitive metadata (stripping)
- Supports the ISO-8859-1, UTF-8, UTF-16 encodings.
- Support for regular expression

3.2.5 Inspection summary search engines

Inspection of Elasticsearch

Name: Elasticsearch
Sources: [134–166]
Positive methodology:

- Looked though documentation pages for Elasticsearch version 5.6:
  - Search APIs
  - Aggregations
  - Indices APIs
  - Query DSL
- Searched google on 16-10-2017 (DD-MM-YYYY) for:
  - elasticsearch compression index
  - elasticsearch deduplication index hashing
  - Elasticsearch stemming
  - elasticsearch case search
- Searched google on 20-10-2017 (DD-MM-YYYY) for:
  - elasticsearch field boosting
  - elasticsearch export search result

Inspection Result

- Elasticsearch uses distrubuted search and can search specific indexes/shards and retrieve all documents written by a given author, or retrieve all documents of a given type. There is also a option to search all indexes. This can be useful for an forensic examiners that want to focus on a given cluster of evidence. If indexes contain more important information than others, then you can elevate priority of these indexes. The indexes involved in a given search can also be printed. Indexes are compressed using the DEFLATE algorithm in order to minimize the storage requirements.
- Option to terminate a search process during execution by a command, by a elapsed time threshold or by setting a max number of documents to be retrieved.
- Can sort search results on multiple search fields. For numerical values sum, max, min, average and median can be sorted on. Sorting on latitude and longitude in some instances are also possible. Field collapsing can be combined with sorting to get only the leading sorted documents on the collapsed field.
- Support for wildcard in search for both extending and limiting the scope of the search.
- Elasticsearch allows to customize what is returned from a search (message, values, etc) with Script Fields. Can compile a summary of the search results by using aggregations in ElasticSearch.
- With post filter ElasticSearch can narrow search results based on membership status. For example post filter can narrow search results that have the same producer, product line, and colour.
- Multiple fields can be selected to highlight search hits on.
- Can customize how scrolling is done with the search results. Alternatively the set of current search results can be used to retrieve the next batch of search hits.
- For calculating the relevancy of documents ElasticSearch uses term frequency/inverse document frequency (TF/IDF). TF/IDF considers how often the search term is in the documents, gives higher weight

for more uncommon search terms in the index and gives terms matching with shorter fields higher priority then matching long fields. The relevance score can be used as a threshold to exclude less relevant search hits. It is also possible to set the weight of importance for specific fields.

- With the Profile parameter the user can get debug information regarding the performance of the search
- Support for boolean quary against numerical data, matching words, type, matching prefix (similar to substring matching) on terms, and range data. The boolean quary supports AND, OR, NOT, grouping operators with '()', joining quires etc. An alternative to the not operation, is using a list of matching terms that should get reduced relevancy score. The queries can also use regular expression and wildcard matching. By setting the Levenshtein Edit Distance the boolean queries can get results based on fuzzy matching. MoreLikeThis quary identifies similar documents to those the user lists.
- Can delete some or all cache data for indexed documents.
- Multi threading support as well as support for multiple concurrent searches.
- Little support for deduplication, you can search for matching fields as indicators for duplicate files. Deduplication was not identified as a advertised feature of ElasticSearch during the inspection.
- Good support for stemming (mapping to root of a word)
- Support for case insensitive and case sensitive search by using or omitting the lowercase function.
- A feature that ElasticSearch lack is the ability to export search results

Inspection of Solr

Name: Solr
Sources: [167–183]
Positive methodology:

- Looked though pages in Solr Ref Guide 7.0
  - Searching
- Searched google on 20-10-2017 (DD-MM-YYYY) for:
  - solr deduplication
- Searched google on 21-10-2017 (DD-MM-YYYY) for:
  - solr regex
  - solr character limit
  - solr substring match
  - solr tf idf
  - solr language detection

Inspection Result

- Possible to highlight matching search terms in documents with coloured borders
- Support for clustering search results in labelled clusters.
- Just like ElasticSearch Solr also support MoreLikeThis queries.
- Search hits can be be sorted on all non-multi valued fields.
- It is possible for the user to reduce the search results by ignoring the first x documents in the result set, or by setting a maximum number of allowed results. A filter query can also be used to specify multiple conditions to reduce the size of the resulting set.
- A user can set a max elapsed time for searching, search processes that lasts longer than this threshold is terminated.
- Solr enables users to customize the information presented with the result set.
- Support for wildcard characters, fuzzy matching using Levenshtein Distance and setting a minimum of criterias that have to match.
- A user can set the importance of a given field. Unlike ElasticSearch Solr can also set a fixed relevancy score for any documents that match a search term, the presence of more than 1 matching terms in the results set would be irrelevant for the relevancy score.
- Support matching documents on numerical and text fields that fall within a specified interval.
- Solr supports the distinct boolean operators AND, OR, NOT, +(single term have to be present in document) and Grouping with parenthesis.
- With Solr you can export a sorted result set in JSON format.
- Solr has a feature for recommending key terms to user, that they can use to search.
- Solr have hash signatures and fuzzy matching to detect duplicated documents. These can be used for deduplication.
- Solr field collapsing works similar to elasticSearch
- Solr support documents encoded and indexes using UTF-8 encoding.
- Solr supports regular expressions.

- EdgeNGramFilterFactory in Solr can be used for sub-strings matching.
- Solr uses TF-IDF just as ElasticSearch
- At index time Solr can figure out the language of the documents being indexed.
- Possible to make text lowercase and hence support case insensitive search

Inspection of Sphinx

Name:Sphinx
Sources: [184–193]
Positive methodology:

- Looked though pages in Sphinx 2.2.11-release reference manual
  - searching
  - Additional functionality
- Searched google on 22-10-2017 (DD-MM-YYYY) for:
  - sphinx regex

Inspection Result

- For Boolean search Sphinx support AND, OR, NOT and grouping by parenthesis.
- Sphinx can use relevance ranking schemes for search hits: TF-IDF, user defined field importance (as in Solr and ElasticSearch), if the search terms are in the same order in the search string and the documents, the count of distinct matching terms, overall number of matched terms etc.
- It is possible to sort on 1 to 5 document fields, sorting on time or by a customized math function.
- Can cluster search hits on fields or by time information
- For scaling Sphinx allows users to manually setting up distributed searching.
- Support regular expressions, wildcards and sub-string matching.
- Can set the number of possible concurrent searches.
- Can process UTF-8 encoding.
- Matching terms highlighting in documents

3.2.6 Search capability comparison

Table 5 shows a short summary of the preceding analysis. A checkmark means that the given program has the capability and an empty cell means that it does not.

Table 5: Comparison of search features
Source: [37–42, 51–56, 98–193]

Capability	Sleuthkit	Volatility	Mozilla InvestGator	Hachoir	ElasticSearch	Solr	Sphinx
Regular expression	✓	✓	✓	✓	✓	✓	✓
Decide/Insensitive case	✓	✓	✓	✓	✓	✓	
Concurrent search	✓				✓		✓
Automate search, with respect to keyword list	✓						
Import keywords	✓						
Export keywords	✓						
Periodical search (search at regular time intervals)	✓						
Substring matching	✓				✓	✓	✓
Export search results	✓				✓	✓	
Match highlighting	✓				✓	✓	✓
UTF-8 Encoding support	✓		✓	✓	?	✓	✓
UTF-16 Encoding support	✓			✓	?		
ISO-8859-1 Encoding support				✓	?		
Deduplication support	✓					✓	
Approximate hash based matching	✓						
Orphan/deleted file search	✓						
RAM search	✓	✓	✓				
Matching memory structures (pre-made)		✓					
Hash database lookups	✓						
Wildcard		✓			✓	✓	✓
Binary search	✓	✓					
HTML renderer for search results		✓					
Support for masking sensitive fields			✓				
Exact hash matching			✓				
System provided keyword suggestions						✓	
AND, OR, NOT, GROUP boolean operators					✓	✓	✓
+ boolean operator (term have to exist)						✓	
File search filter			✓				
Retrieval of documents not matching filters			✓				
Set max search hits			✓		✓	✓	
Stripping senstive metadata			✓				
Increase search priority of important indexes					✓		
Terminate search after a given elapsed time					✓	✓	
Sorting search results					✓	✓	✓
Customized message/ post-search action					✓	✓	
Aggregated summary of search results					✓		
Narrow search results with post filter					✓	✓	
Set relevancy weight for field					✓	✓	
MoreLikeThisQuery					✓	✓	
Search result clustering						✓	✓
Minimum matching criteria						✓	
Fixed relevancy score						✓	
Field collapsing					✓	✓	
Support for TF-IDF					✓	✓	✓
Language detection on index time						✓	
Fuzzy matching					✓	✓	✓
Faceted search					✓	✓	
Phonetic search					✓		
Geospatial search					✓	✓	
Streamed search						✓	

4 Systematic Literature review 3 - Digital forensic related datasets

4.0.1 Purpose of the literature review

Identify and summarize publicly available datasets that relates to digital forensic and consider their applicability for this thesis experiments. Table 6 shows examples of relevant datasets.

Research question

2.a) What are the publicly available forensic related datasets for testing?

Catagory	Abbreviation	Example dataset
Forensics images	IMG	*The Real Data Corpus (RDC)*
Files	FILE	*RAISE (RAw ImageS datasEt)*
RAM contents	RAM	*Memory Buddies Traces(MBT)*
Network	NET	*Common crawl*
Malware	MAL	*Kharon dataset*
Email	EM	*The Webb Spam Corpus 2011*
SMS	SMS	*The NUS SMS Corpus*
Password	PASS	*Yahoo Password Frequenc*
Phishing	PHI	*Phishing Websites Data*
Spam	SPAM	*TREC 2011*
Authorship	AUTH	*Personae*
Financial data/ fraud	FIN	*CMS dataset*
Forgery corpus	FORG	*MICC-F2000*
Collection of different datasets	COLL	*CAIDA data*

Table 6: Examples of digital forensic datasets

Decisions was made to limit the scope of the data collection, by excluding biometric datasets such as images of fingerprints, hand signature, gait, voice recognition and iris. However, the review will include authorship attribution corpus.

4.0.2 Protocol/methodology

1. Search digital libraries, scan scientific articles for names, direct links or sources related to the datasets above, and use this information on google search engine to identify individual datasets or repositories of datasets.
2. Document search phrases that resulted in identifying new datasets.
3. Repeat step 1 and 2 with other resources like github, Kaggle and figshare to locate more datsets.

In the planning phase of the literature, my supervisors aided me by providing 3 sources to publicly available datasets resources

1. *Govdoc1*
2. *AZSecure-data*
3. *Enron dataset*

These datasets/collections have also been included in my review.

During the collection phase of the literature review, three related reviews were identified. The first review was from 2014 and identified 7 datasets [194]. The second review is as recent as 2017 and compiled a list online of 79 digital forensic related datasets [195], [196]. The third review [197] was from 2014 and focused on network related datasets. It identified 10 datasets, two of which are broken links, and four available datasets not considered for inclusion in this review due to time considerations.

This review expands on the three reviews and its findings were largely independent from the three previous works. The table 7 shows how this review expands the knowledge on the topic of publicly available forensic related datasets. In table 7 the index, name and category of the collected datasets is presented as well as which of the 4 reviews included it[where this paper is counted as one of the 4 reviews]. The assigned dataset indexes, names and categories will be the same throughout this review. Not all dataset creators explicitly named their creation, in those cases a new name was assign Ad hoc to the dataset. More information about each individual dataset or repository can be found in section 4.0.4. Two datasets in the review from 2014 [194] was not included in this paper. The first was a face recognition dataset and the second was a forensic image dataset called MemCorp, which I could not identify online. The review from 2017 [195] included 39 datasets that was not identified in this systematic literature review. The 39 dataset has been included in this review as a collection of dataset that is summarized in table 9 with index 21 and name DFCF review. DFCF review contains four language corpora, and languages were not considered a category for forensic related corpora in this paper. The difference in methodology with this review and [195] seems to be that this review did not restrict the publishing year when searching though scientific databases, different categories was considered relevant, and dataset databases like Kaggle and figshare was searched in this review.

Index	Name	Catgory	This paper/review	in review [194]	in review [195]	in review [197]
1	BAC	AUTH	✓			
2	CTFAC	AUTH	✓			
3	PCSN	AUTH	✓			
4	TBGC	AUTH	✓			
5	Personae	AUTH	✓			
6	PAN-Enron	AUTH	✓			
7	PAN/CLEF12	AUTH	✓			
8	PAN13	AUTH	✓			
9	PAN14	AUTH	✓			
10	PAN15	AUTH	✓			
11	RCTAC	AUTH	✓			
12	MUD03	AUTH	✓			
13	netresec	COLL	✓			
14	MTA13-17	COLL	✓			
15	pcapr	COLL	✓			
16	PCAPsDB	COLL	✓			
17	CAIDA	COLL	✓		✓	✓
18	csmining	COLL	✓			
19	AZSecure-data	COLL	✓			
20	Digital Corpora	COLL	✓		✓	
21	DFCF review	COLL	✓		✓	
22	GIfiles	EM	✓	✓		
23	USHCE	EM	✓			
24	419 dataset	EM	✓			
25	MLE200	EM	✓			
26	Enrondata	EM	✓	✓		
27	Raise	FILE	✓			
28	SherLock	FILE	✓			
29	AndroZoo	FILE	✓			
30	CTD15	FIN	✓			
31	UCSD-FICO-09	FIN	✓			
32	CMS	FIN	✓			
33	PaySim	FIN	✓			
34	BankSim	FIN	✓			
35	MICC	FORG	✓			
36	Brian Carrier	IMG	✓		✓	
37	RDC	IMG	✓	✓	✓	
38	CFReDS	IMG	✓	✓	✓	
39	VirusShare	MAL	✓			
40	BIG15	MAL	✓			
41	Drebin	MAL	✓		✓	
42	DroidWare	MAL	✓			
43	MILCOM16	MAL	✓			
44	Kharon	MAL	✓			
45	Mudflow	MAL	✓			
46	ISOT2010	MAL	✓			
47	ECML/PKDD07	MAL	✓			
48	CSIC10	MAL	✓			
49	BlogPcap	MAL	✓		✓	
50	Malrec	MAL	✓			
51	CTU-13	MAL	✓			
52	ISCX Botnet	MAL	✓			
53	ISCXAB	MAL	✓			
54	DAROA98/99	NET	✓	✓		✓
55	DARPA2000	NET	✓			✓
56	MAWILab	NET	✓			✓
57	KDD Cup99	NET	✓			
58	UNSW-NB15	NET	✓			
59	NSA-CDX	NET	✓			
60	ADFA	NET	✓			
61	Kyoto data	NET	✓			
62	crawdad	NET	✓		✓	✓
63	ICS-pcap	NET	✓			
64	Common Crawl	NET	✓			
65	NSL-KDD	NET	✓			
66	ISCXTNT	NET	✓			
67	ISCXVNV	NET	✓			
68	ISCXIDS	NET	✓			
69	YPFC	PASS	✓			
70	VincentPassword	PASS	✓			
71	MBT08	RAM	✓			✓
72	NUSSC	SMS	✓			
73	WebbSC11	SPAM	✓			
74	DITSSC	SPAM	✓			
75	TREC05-07	SPAM	✓			
76	Hewlett spam	SPAM	✓			
77	WEBSPAMUK07	SPAM	✓			
78	microblogPCU	SPAM	✓			
79	TREC11	SPAM	✓			
80	SPAM/HAM	SPAM	✓			
81	phishtank	PHI	✓			
82	millersmiles	PHI	✓			
83	PWDS15	PHI	✓			

Table 7: Where the datasets/collections can be found

4.0.3 Search phrases and justification

Documents were excluded from consideration if their title had little relation to information security, and if the document format was not easily searchable. An example of the

latter case is pdf documents scanned by a scanner machine, where full text search of the text content is not applicable. Without the assistance of search, the process of finding the datasets would be too time consuming.

In table 4.0.3 is a summary of the collection phase of the literature review. Entries included in this table all lead to finding new datasets. An entry has an ID number, search phrase + search options, database name (search resource) and the number of hits for the search phrase. Entries with ID 1-5 are essentially full text search (matching based on meta data and text content). Fulltext search lead to more false positives, then only Meta search. But was used in cases where the number of hits was manageable. An example for when fulltext was deemed unmanageable can be seen in entry 6, where meta search was used instead. The phrase 'forensic dataset' was used to find different types of relevant datasets, but this phrase alone is not good enough. This is because relevant papers may use publicly available datasets, but does not contain the term 'forensic'. Therefore more specific search terms from list in subsection 4.0.1 was also used. In entry 9 the NOT operator was used to discard biometric datasets. Entry 10 in table 4.0.3 returned hits that both included the phrase 'IDS dataset' and the term 'Network' in the meta data, and excluded hits that contained some already known network datasets. The term IDS was used to reduce the number of non-network related articles. This term may exclude some relevant hits, but its usage is justified as the other search phrases also covered some network related datasets. In entry 16, the first 10 results was used on Google to look find datasets on Github. This was done, as it was tricky to identify relevant repositories using Githubs internal search. In entry 18 figshare did not provide the number of hits. Therefore Not Available (N/A) is in the #Hits column for this entry.

ID	Search phrase (comma (,) separates search options)	DB	Hits
1	forensic corpora, exact phrase match	link.springer	22
2	forensic corpus', advanced search, both words must match (be present) in any field	dl.acm	9
3	forensic corpora', advanced search, both words must match (be present) in any field	dl.acm	3
4	forensic dataset', advanced search, both words must match (be present) in any field	dl.acm	61
5	forensic corpus, full text search	search.arxiv	112
6	forensic dataset, in metadata only	ieeexplore	94
7	malware dataset, in metadata only	ieeexplore	174
8	((password dataset) NOT biometrics), in metadata only	ieeexplore	19
9	Spam dataset, in metadata only	ieeexplore	173
10	(((((((((IDS dataset) AND Network) NOT DARPA) NOT KDD) NOT KDD99cup) NOT DARPA98) NOT DARPA99) NOT DARPA-98) NOT DARPA-99) NOT NSL-KDD), in metadata only	ieeexplore	118
11	fraud dataset, in metadata only	ieeexplore	104
12	Forensic dataset, in All Sources(Computer Science), no books	sciencedirect	1100
13	fraud	kaggle	10
14	spam	kaggle	3
15	email	kaggle	18
16	dataset github	google	576000
17	spam	figshare	107
18	network	figshare	N/A

Table 8: Summary of search strings used to find relevant sources

4.0.4 Search summary - datasets:

Table 9 is a summary of the identified datasets in this review. An entry in this table is explained in the list below:

- Column I = Numbered index
- Column Name = The short name of the dataset
- Column Acc = Access, where P=public and R=By request
- Column DT = Data type, where S = Synthetic, R=Real and H=Hybrid Column CAT= Catagory, where the catagories abbriviations is shown in table 6
- Column Size= Size is either given in S=samples, MB, GB, (comrpressed/uncompressed) or Not avaliable (N/A)
- Column Description= A description that will include the name of the dataset, where it can be downloaded from, include original paper if available and additional details about the dataset.

Table 9: Comparative summary of digital forensic datasets

I	Name	Acc	DT	Cat	Size	Description
1	BAC	P	R	AUTH	19320 S	The Blog Authorship Corpus (BAC): A authorship corpus of 681288 Blog entries and 19320 problems [198], [199] Obtain dataset here[1]

[1] http://u.cs.biu.ac.il/~koppel/BlogCorpus.htm

. . . continued

I	Name	Acc	DT	Cat	Size	Description
2	CTFAC	P	S	AUTH	20 S	A capture the flag (CTF) authorship corpus (CTFAC) [200], [201]. The corpus have been used in the multi-classification problem of classifying the origin of the exploit attempts to one of 20 CTF teams. The data is available in JSON format and includes source and destination of attack, timing information and histogram of payload. Obtain dataset here [2]
3	PCSN	R	R	AUTH	609 S	Polish Corpus of Suicide Notes (PCSN) are real suicide letter written by both young and old polish men and women from the period of 1999-2009 [202], [203]. Obtain dataset here [3]
4	TBGC	P	R	AUTH	12 S	The Brennan-Greenstadt corpus (TBGC) contains two documents from each of the 12 participating authors [204], [205]. In the first text the authors attempted to obfuscate the characteristics of their writing. And in the second text the authors tried to imitate the writing style of a different writer. Obtain dataset here [4]
5	Personae	R	R	AUTH	145 S	The paper claims that the size of the German corpus Personae makes it possible to classify the author of the text as well as the author personality [206], [207]. Personae consist of 145 bachelor student essays with lengths around 1400 words. The students, took a personality test. This test made classification of their personality possible. But it is difficult to infer from the sources [206], [207] whether the personality test is part of the dataset or not. Obtain dataset here [5]
6	PAN-Enron	P	R	AUTH	12338 S	PAN-Enron corpus is a subset of the Enron dataset and can be used for authorship attribution and verification. 24% of the samples is from non-Enron authors while the rest is from the Enron set [208], [209]. Names and email addresses was omitted from the dataset. Obtain dataset here [6]

[2] https://www.dropbox.com/sh/17d4eyg0cwoxg8s/AAA5g1NvQw-tUoZPvldloddRa?dl=0
[3] http://www.pcsn.uni.wroc.pl/
[4] https://psal.cs.drexel.edu/index.php/JStylo-Anonymouth
[5] https://www.clips.uantwerpen.be/datasets/personae-corpus
[6] http://pan.webis.de/data.html

. . . continued

I	Name	Acc	DT	Cat	Size	Description
7	PAN/ CLEF12	P	R	AUTH	N/A	The dataset contains training and test data for several authorship attribution scenarios based on works of fiction. Each scenario has a different amount of authors, number of documents, and minimum word length [210], [209]. Obtain dataset here [6]
8	PAN13	P	R	AUTH	110 S	Authorship classification on English, Spanish and Greek texts. Most of the documents are in the word length range 1001-1500 words [211], [209]. Obtain dataset here [6]
9	PAN14	P	R	AUTH	4959 S	Authorship attribution corpus with documents written in English, Dutch, Spanish, and Greek [212], [209]. University students created the Dutch and English documents. And the Spanish and Greek documents was obtained from newspapers. Obtain dataset here [6]
10	PAN15	P	R	AUTH	3701 S	Authorship attribution corpus with documents written in English, Dutch, Spanish, and Greek. The authors of the Dutch documents was Students at a university in Belgium [213], [209]. English documents was taken from theatre plays. Spanish and Greek documents was obtained from opinion articles. Obtain dataset here [6]
11	RCTAC	P	R	AUTH	1000 S	Reddit Cross-Topic AV Corpus (RCTAC) consist of 1000 Reddit users and their comments from 2010-2016 on 1388 different subjects [214], [215]. Obtain dataset here [7]
12	MUD03	P	N/A	AUTH	750000 S	Masquerading User Data 2003 (MUD03). A dataset of 750000 UNIX commands [216], [217]. The commands are either from one of 50 legitimate users or a user imitating one of the 50. Each legitimate user has 5000 commands that is theirs and a random proportion of 10000 commands that is attribute to them or a masquerading person. Obtain dataset here [8]
13	netresec	P	P,R	COLL	N/A	This website has compiled a list of available PCAP files online [218]. The list contain PCAP files that have been used in competitions, conferences, and PCAP files that contain malware or exploits. Obtain dataset here [9]

[7] https://www.dropbox.com/sh/f2mlp6u5vervx9b/AABr_c7qrmahCqUviIu3ORz6a?dl=0
[8] http://www.schonlau.net/intrusion.html
[9] https://www.netresec.com/?page=PcapFiles

. . . continued

I	Name	Acc	DT	Cat	Size	Description
14	MTA13-17	P	N/A	COLL	≈ 1100 S	malware-traffic-analysis (MTA) website host a collection of PCAP files and malware samples [219]. Obtain dataset here [10]
15	pcapr	P	S,R	COLL	6,050,710,9 S_1 / 3465 S_2	A large publicly available searchable database that contains 60507109 packets and 3465 pcaps [220]. Obtain datasets here [11]
16	PCAPs-DB	P	S,R	COLL	N/A	PCAPsDB is a repository of different PCAP files, some of which are Malware [221]. Obtain dataset here [12]
17	CAIDA	P	R	COLL	71 S	CAIDA Data: A collection that contains a mixture of publicly availably and by request network datasets [222]. These network dataset focuses on Autonomous System (AS) topology, denial of service attacks, worms, anonymized passive network traffic monitoring and darknet traffic. Obtain dataset here [13]
18	cs-mining	P	R	COLL	10 S	A collection of spam datasets, news articles, system calls from malware and Reuters-21578 dataset used for text categorization [223]. Obtain datasets here [14]
19	AZ-Secure-data	P	R	COLL	111 S	Is a resource that contains list of geopolitical discussions, dark web forum threads, phising and legitimate sites, malware, network traffic and data from social media communication [224]. Obtain datasets here [15]
20	Digital Corpora	P,R	R	COLL	N/A	Is a collection that contain: • Cell phone images • Disk images and the already mentioned RDC dataset. • Network traffic including the already mentioned DARPA 1998,1999 and 2000 datasets. • Govdocs1 a dataset with ≈ 1000000 files. [225] Obtain datasets here [16]
21	DFCF review	P	R,S	COLL	79 S	DATASETS FOR CYBER FORENSICS (DFCF) contains 39 datasets of type: chat logs, APK, File, Malware, forensic images, picture, email, network, RAM dumps, language corpora not otherwise considered in this review [195], [196]. Obtain datasets here [17]

[10] http://malware-traffic-analysis.net/
[11] http://www.pcapr.net/home
[12] https://www.evilfingers.com/repository/index.php
[13] https://www.caida.org/data/overview/
[14] http://csmining.org/index.php/data.html
[15] http://www.azsecure-data.org/other-data.html
[16] https://digitalcorpora.org/corpora
[17] http://datasets.fbreitinger.de/datasets/

. . . continued

I	Name	Acc	DT	Cat	Size	Description
22	GIfiles	P	R	EM	5,000, 000 S	The Global Intelligence files (GIfiles) are a collection of 5 million leaked emails from Stratfor, that gives insight into how the intelligence community operates [226], [227]. Obtain dataset here [18]
23	USHCE	P	R	EM	60 MB S	≈ 7000 PDF pages of [US president candidate] Hillary Clinton emails (USHCE) was released as a result of a freedom of information claim [228]. The dataset contains both email content and metadata. Obtain dataset here [19]
24	419 dataset	P	R	EM	2500 S	419 fraud emails: Content and metadata from 2500 fraud emails [229]. Obtain dataset here [20]
25	MLE-200	P	R	EM	200 S / ≈ 2MB	Multilanguage emails(MLE): Spanish, English Portuguese Emails [230]. Obtain dataset here [21]
26	Enrondata	P	R	EM	N/A	Enrondata is a website that has a compiled list that links to differernt versions of the ENRON dataset that is in PST or MIME format and with different record sizes [231]. Obtain dataset here [22]
27	RAISE	P	R	FILE	350GB N/A	RAw ImageS datasEt (RAISE): 8156 Unprocessed and high resolution images. The images are taken by the following cameras: Nikon D40, Nikon D90 and Nikon D7000 [232], [233]. The original paper states that this dataset can be useful to test image forgery algorithms [232]. Obtain dataset here [23]
28	SherLock	R	R	FILE	10 billion S	SherLock is a Android Smartphone dataset that contains running application/process information, sensory data and OS data captured with normal user privileges [234], [235]. The dataset also has labels that can be assigned to describe ongoing malicious activity on a phone. Obtain dataset here [24]

[18] https://wlstorage.net/torrent/gifiles/
[19] https://www.kaggle.com/kaggle/hillary-clinton-emails
[20] https://www.kaggle.com/rtatman/fraudulent-email-corpus
[21] https://figshare.com/articles/Corpus_200_Emails/1326662
[22] https://enrondata.readthedocs.io/en/latest/references/data/
[23] http://mmlab.science.unitn.it/RAISE/
[24] http://bigdata.ise.bgu.ac.il/sherlock/#/download

. . . continued

I	Name	Acc	DT	Cat	Size	Description
29	Andro-Zoo	R	R	FILE	5,546,565 S	AndroZoo dataset includes over 5 million android applications (APKs) [236], [237]. The APKs was obtained by crawling multiple APKs distributors such as google play, AppChina, torrents etc. Efforts was made to avoid downloading duplicate files from the same vendor. But creators of the dataset gives no guarantees that the same file was not downloaded from multiple vendors. Each sample contains a zipped apk file, with its byte code, meta data, signed certificate and miscellaneous files. Obtain dataset here [25]
30	CTD15	P	R	FIN	68MB / 284807 S	Credit transaction dataset (CTD) A numerical dataset of 284807 bank transactions [238], [239]. Of all the transactions only 492 of them are fraudulent. In the feature vector there are 28 principal components, the elapsed time between transactions, the transactions amount, and the fraud/not fraud class label. Obtain dataset here [26]
31	UCSD-FICO-09	P	R	FIN	> 100000 S	UCSD-FICO-09 is a electronic commerce fraud dataset with anonymized features [240], [241]. The numerical dataset has both labelled (training) and unlabelled(testing) samples. Obtain dataset here [27]
32	CMS	P	R	FIN	≈ 6 GB	Real financial statements in .csv format from Centers for Medicare & Medicaid Services [US], in the year 2013-2016 [242], [243]. Obtain dataset here [28]
33	PaySim	P	S	FIN	182MB	A smaller subset of the Synthetic PaySim dataset is available on Kaggle [244], [245]. The synthetic data was generated based on real world examples. The feature vector contain time information, transaction type, identifier for the user(s) involved in a transaction, old and current state of the balance sheet and class labels regarding if the transactions is considered fraudulent. Obtain dataset here [29]
34	Bank-Sim	P	S	FIN	13MB	BankSim is a synthetic fraud dataset with 587443 legitimate and 7200 illegitimate transactions/samples [246], [247]. The simulating agent generating the dataset is based on real bank transactions. The feature vector contains age range, gender catagory, what the transaction was spent on, zip code etc. Obtain dataset here [30]

[25] https://androzoo.uni.lu/
[26] https://www.kaggle.com/dalpozz/creditcardfraud
[27] https://www.cs.purdue.edu/commugrate/data/credit_card/
[28] https://www.cms.gov/OpenPayments/Explore-the-Data/Dataset-Downloads.html
[29] https://www.kaggle.com/ntnu-testimon/paysim1
[30] https://www.kaggle.com/ntnu-testimon/banksim1

. . . continued

I	Name	Acc	DT	Cat	Size	Description
35	MICC	P	R	FORG	220/ 2000 S	MICC-F220 and MICC-F2000 are datasets that contains untouched images and images where parts of the image is modified by scaling, rotating and scaling [248], [249]. The datasets have been used to benchmark a copy-move forgery algorithm. Obtain dataset here [31]
36	Brian Carrier	P	R	IMG	14 S	A collection of forensic images made/hosted by Brian Carrier [250]. The 14 forensic images can be divided up into the following categories: NTFS file systems, FAT file system, ISO9660 file system and a memory image. Brian created scenarios to test string search, partitions with multiple file systems, file carving etc. Obtain dataset here [32]
37	RDC	R	R	IMG	70TB Compressed	The Real Data Corpus (RDC) is data collected of digital devices from the secondary market [251]. The dataset contains hard drives images, flash memory images and CDROMS. According Obtain dataset here [33]
38	CFReDS	P	S	IMG	16 S	Computer Forensic Reference Data Sets (CFReDS) can be used for forensic tool testing [252]. CFReDS includes forensic images and simulated data for memory forensics, file carving, string search and file recovery. Obtain dataset here [34]
39	VirusShare	R	R	MAL	2,938,567,4 S	VirusShare.com is a virus sharing website with currently 29385674 malware samples [253]. Obtain dataset here [35]
40	BIG15	P	R	MAL	≈ 500GB	A dataset for classifying known malware and their associated malware family [254]. There are in total 500GB worth of malware samples, that belongs into one of 9 families of malware. Obtain dataset here [36]
41	Drebin	R	R	MAL	5560 S	The Drebin Dataset have 5560 malicious android applications that can be categorized into one of 179 malware families [255], [256]. Obtain dataset here [37]
42	DroidWare	P	S	MAL	399 S	DroidWare is a malware dataset for the android platform. The dataset is made up of 278 benign and 121 malicious samples [257], [258]. Each sample has a 152 feature vector of Android application permissions. Obtain dataset here [38]

[31] https://github.com/lambertoballan/sift-forensic/blob/master/README.md
[32] http://dftt.sourceforge.net/
[33] https://digitalcorpora.org/corpora/disk-images/real-data-corpus
[34] https://www.cfreds.nist.gov/
[35] https://virusshare.com/
[36] https://www.kaggle.com/c/malware-classification/data
[37] https://www.sec.cs.tu-bs.de/~danarp/drebin/
[38] https://github.com/RECOVI/DroidWare

. . . continued

I	Name	Acc	DT	Cat	Size	Description
43	MIL-COM16	P	S	MAL	4S	Synthetic dataset with 4 botnet samples. The botnet actions in each sample differs from injection, reconnaissance, command and control (C&C) communication channels and botnet prorogation [259], [260]. Obtain dataset here [39]
44	Kharon	P	R	MAL	7 S	Kharon dataset contains malware documentation, that has been used to benchmark GroddDroid capability to trigger malicious code [261], [262]. The documentation was obtained though Static and dynamic analysis on a set of malware samples. The documentation includes the location of the malicious code blocks, the trigger conditions, and how the malware acts when triggered. Obtain dataset here [40]
45	Mudflow	P	R	MAL	18204 S	The Mudflow dataset was used to train a "benign or malign" binary classifier, on the information flow from benign Android applications obtained from the Google store [263], [264]. Instead of focusing on what resources the applications request access to, the classifier determines if the usage of these resources are to be deemed normal. The dataset contains 2866 benign and 15338 malicious apps. Obtain dataset and scripts here [41]
46	ISOT-2010	P	R	MAL	N/A	ISOT is a dataset that is built up of benign and malicious network traffic, from multiple sources [265], [266], [267]. The malicious samples is off the Storm and Waledac botnets. Obtain dataset here [42]
47	ECML/PKD-D07	R	H	MAL	50000 S	ECML/PKDD-2007: A dataset made up of 40 000 normal queries, 9000 exploits with descriptions on the target environment and 1000 exploits without target information [268]. The dataset is in XML format and contain attacks from 7 different categories. Obtain dataset here [43]

[39] https://cybervan.appcomsci.com:9000/datasets
[40] http://kharon.gforge.inria.fr/dataset/
[41] https://www.st.cs.uni-saarland.de/appmining/mudflow/
[42] https://www.uvic.ca/engineering/ece/isot/datasets/index.php#section0-0
[43] http://www.lirmm.fr/pkdd2007-challenge/index.html#dataset

. . . continued

I	Name	Acc	DT	Cat	Size	Description
48	CSIC10	P	R	MAL	610000 S	The motivation behind the creation of the HTTP-CSIC-2010 dataset, was a shortage of malware datasets that exploits real web applications [269]. HTTP-CSIC-2010 contains malign and benign labelled request against a Spanish electronic commerce web application. The samples with malicious consequence target the confidentiality and integrity of the web application resources. Obtain dataset here [44]
49	Blog-Pcap	P	N/A	MAL	1000 S	A list of 1000 Malware PCAP files. The blog also contain other malware samples as well [270], [271]. Obtain dataset here [45]
50	Malrec	P	R	MAL	24389 S	The Malrec Dataset: A dataset of the system calls and arguments made by malicious software [272]. Obtain dataset her [46]
51	CTU-13	P	R	MAL	334 S	Malware Capture Facility Project: A repository of Botnet and benign network traffic [273]. Obtain dataset here [47]
52	ISCX Bot-net	R	H	MAL	N/A	ISCX Botnet: Is a derivative dataset from multiple sources [274], [275]. The dataset contains normal traffic and malicious traffic from 16 different families of botnets. Obtain dataset here [48]
53	ISCXAB	R	R	MAL	1929 S	ISCX Android Botnet (ISCXAB): This dataset is built up of AnserverBot, Bmaster, DroidDream, Geinimi, MisoSMS, NickySpy, Not Compatible, PJapps, Pletor, RootSmart, Sandroid, TigerBot, Wroba and Zitmo botnets in the form of Android application package (APK) files [276], [277]. Obtain dataset here [49]
54	DARPA-98/99	P	S	NET	38/50 S	DARPA 1998 and 1999 is datasets of simulated network traffic used to assess the detection capabilities of intrusion detection systems [278], [279]. DARPA 1998 contains 38 categories of UNIX based attacks. DARPA 1999 increases the number of categories to 50 and added Windows NT based exploits as well. Obtain dataset here [50]
55	DARPA-2000	P	S	NET	2 S	DARPA 2000 has simulated data from two distributed denial of service attacks [280]. Obtain dataset here [51]

[44] http://www.isi.csic.es/dataset/
[45] https://www.dropbox.com/sh/7fo4efxhpenexqp/AACmuri_1-LDiVDUDJ3hVLqPa?dl=0
[46] http://moyix.blogspot.no/search?q=dataset
[47] https://stratosphereips.org/category/dataset.html
[48] http://www.unb.ca/cic/research/datasets/botnet.html
[49] http://www.unb.ca/cic/research/datasets/android-botnet.html
[50] https://ll.mit.edu/ideval/data/index.html
[51] https://ll.mit.edu/ideval/data/2000data.html

. . . continued

I	Name	Acc	DT	Cat	Size	Description
56	MAWI-Lab	P	R	NET	N/A	The MAWILab database contains labels, that categorize network anomalies. It can be used to assist in evaluating the performance of intrusion detection systems (IDS) [281], [282]. Obtain dataset here [52]
57	KDD Cup99	P	S	NET	743M	KDD Cup 1999 Data: A synthetic dataset that is made up off network traffic samples [283]. These samples is labelled benign or malign [284]. Malign samples are attempting to attack availability, to perform privilege escalation, to imitating a local user and to perform reconnaissance. The dataset is produced based on the DARPA98 dataset. Obtain dataset here [53]
58	UNSW-NB15	P	H	NET	2,540, 044 S	UNSW-NB15: The samples are labelled malign or benign. Each sample has 49 features that includes variables such as time to live (TTL), IP information, sequence number, time between TCP SYN and TCP ACK etc [285], [286]. The malicious samples aims to identify vulnerabilities by perform active reconnaissance and by using fuzzed inputs. The malware samples also attempts to install backdoors, target the availability of services, opening a shell to run arbitrary code and to compromise new hosts. There are in total 2 540 044 samples spread across 4 .csv files, a smaller subset of this dataset is used to create a training and a test set. Obtain dataset here [54]
59	NSA-CDX	P	R	NET	5 S	A collection of Cyber Defense Exercises (CDX) from the National Security Agency (NSA) [287]. In the CDX 2009 collection there are logs of DNS, web server, and IDS. Obtain datasets here [55]
60	ADFA	P	N/A	NET	N/A	The ADFA Intrusion Detection Datasets: According to the author in [288] the ADFA dataset has higher complexity, more attack options, frequency of attacks/normal traffic is more evenly distributed, and more extensive in scope then the KDD 98/99 evaluation datasets [289]. Given the reasons above the author concludes that the ADFA is the better dataset to assess the performance of Host based IDS (HIDS). Obtain datasets here [56]

[52] http://www.fukuda-lab.org/mawilab/data.html
[53] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[54] https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/
[55] http://www.usma.edu/crc/sitepages/datasets.aspx
[56] https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-IDS-Datasets/

. . . continued

I	Name	Acc	DT	Cat	Size	Description
61	Kyoto data	P	R	NET	19683 MB	A numerical dataset that tracks network activity from Honeypots and sensors that is in the management control of Kyoto University [290], [291]. The network activity has been tracked from the end of 2006 to December 2015. The dataset includes 24 features, 14 of them is based on the KDD Cup 99 feature vector and the last 10 features was added to better describe what happens on the network. The latter 10 features are described below: • Binary valued features that cover the observation of IDS alerts, Malicious connections and exploits of the network traffic. • Abnormal session feature state if the network traffic is benign or is a known/unknown attack. • Sanitized IP address, ports and session information was also captured. Obtain dataset here [57]
62	crawdad	P	R	NET	N/A	crawdad is a compiled list of publicly available datasets of wireless protocols [292]. Obtain dataset here [58]
63	ICS-pcap	P	N/A	NET	N/A	ICS-pcap: A repository of ICS/SCADA pcaps gathered from multiple sources [293]. Obtain dataset here [59]
64	Common Crawl	R	R	NET	≈ 2,000,000 S	Common Crawl: A dataset of web content and metadata from 2000000 crawled webpages [294]. Obtain dataset here [60]
65	NSL-KDD	R	S	NET	N/A	NSL-KDD dataset: Is a subset of KDD99 [284], [295] . Preproccsing that was performed on NSL-KDD: • Deduplication of training and testing set samples in NSL-KDD • The count of samples that is hard to classify in the NSL-kDD set, is inversely proportional to the percentage of hard samples in KDD. Obtain dataset here [61]

[57] http://www.takakura.com/Kyoto_data/
[58] https://crawdad.org/all-byname.html
[59] https://github.com/automayt/ICS-pcap
[60] https://aws.amazon.com/public-datasets/common-crawl/
[61] http://www.unb.ca/cic/research/datasets/nsl.html

. . . continued

I	Name	Acc	DT	Cat	Size	Description
66	ISCX-TNT	R	R	NET	22GB	ISCX Tor-nonTor (ISCXTNT): Contains both normal and Tor traffic [296], [297]. The real Tor traffic was generated from network, email and filesharing protocols. And audio, and video streams from popular applications. The samples/traffic is assign a class label for what application or protocol they belong to. Obtain dataset here [61]
67	ISCX-VNV	R	R	NET	28GB	ISCX VPN-nonVPN (ISCXVNV) dataset: Similar to the ISCX Tor-nonTor dataset [298], [299]. It has non-VPN traffic and labelled VPN traffic from multiple applications and protocols. Obtain dataset here [61]
68	ISCX-IDS	R	H	NET	2,297, 128 MB	ISCX IDS (ISCXIDS): This dataset includes real traffic for IPv4, IPv6, UDP, TCP, ARP, DNS, ICMP, HTTP, SMTP, SSH, IMAP, POP3, and FTP generated by using agents that simulate users interacting with these protocols [300], [301]. Obtain dataset here [61]
69	YPFC	P	R	PASS	N/A	Yahoo Password Frequency Corpus(YPFC): A sanitized password frequency corpus that protect the privacy of the user accounts [302], [303] . The scheme also protects up to two duplicate accounts, that has similar passwords. The sanitization is performed to prevent adversaries to gain knowledge of individual users. Obtain dataset here [62]
70	Vincent-Password	P	N/A	PASS	2,000, 000 S / 20 MB	Password dataset with 2000000 samples [304]. Obtain dataset here [63]
71	MBT08	P	R	RAM	N/A	Memory Buddies Traces(MBT): This dataset is built up of the memory contents from computers and servers. The dataset contains metadata of the underlying platform, metadata of live processes, and the hash values of the 4KB memory pages [305], [306], [307]. The server hosting this dataset also host network related datasets (PCAP, TCP traffic, synthetic attacks etc). Obtain dataset here [64]

[62] https://figshare.com/articles/Yahoo_Password_Frequency_Corpus/2057937

[63] http://www.datasciencecentral.com/forum/topics/password-dataset-for-you-to-test-your-data-science-skills

[64] http://skuld.cs.umass.edu/traces/cpumem/memtraces/

. . . continued

I	Name	Acc	DT	Cat	Size	Description
72	NUSSC	P	R	SMS	87300 S	The NUS SMS Corpus (NUSSC) includes 55835 English and 31465 Chinese SMS messages [308], [309]. To avoid bias or promote message diversity in the sampling process, the individual SMS messages was captured without considering any particular topic. The SMS messages can be download in JSON, XML and SQL format. Obtain dataset here [65]
73	Webb-SC11	P	R	SPAM	≈ 350000 S/ 1GB compressed	The Webb Spam Corpus 2011(WebbSC11): A custom crawler was built to collect spam web pages [310], [311]. The resulting collection was preprocessed to remove instances of legitimate websites and websites that could not get resolved. The dataset contains both the spam and the HTTP sessions for the spam servers. Obtain dataset here [66]
74	DITSSC	P	R	SPAM	1353 S	The samples from DIT SMS spam corpus (DITSSC) is a collection of reported SMS spam, by UK mobile users. The corpus is stored in a XML format. Unique entries in the collection was assured by performing case insensitive depublication [312], [313]. Obtain dataset here [67]
75	TREC-05-07	R	N/A	SPAM	3 C	Spam corpora from TREC. This server host 3 spam corpus from 2005-2007 [314]. Obtain dataset here [68]
76	Hewlett spam	P	R	SPAM	4601 S	A spam dataset created by Hewlett-Packard Labs [315]. The feature vector contains: • frequencies of words and characters, • the average length, max length and total count of "uninterrupted sequences of capital letters" • class label Obtain dataset here [69]
77	WEB-SPAM-UK07	P	R	SPAM	1,058, 965,55 S	WEBSPAM-UK2007: A labeled spam dataset of 105896555 entries [316]. Obtain dataset here [70]
78	micro-blog-PCU	P	R	SPAM	221579 S	microblogPCU Data Set: A labeled spam dataset [317]. Example features: • Microblog poster gender, username, user id • Number of followers • Number of reposts Obtain dataset here [71]

[65] https://github.com/kite1988/nus-sms-corpus
[66] https://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html
[67] http://www.dit.ie/computing/research/resources/smsdata/
[68] http://trec.nist.gov/data/spam.html
[69] http://archive.ics.uci.edu/ml/datasets/Spambase?ref=datanews.io
[70] http://chato.cl/webspam/datasets/uk2007/
[71] https://archive.ics.uci.edu/ml/datasets/microblogPCU

. . . continued

I	Name	Acc	DT	Cat	Size	Description
79	TREC-11	R	R	SPAM	1,600,000,0S	TREC 2011 microblog dataset contains 16 million normal and spam twitter posts [318]. Obtain dataset here [72]
80	SPAM/HAM	P	R	SPAM	273 MB/5780 S	Image Spam Dataset: contains images that is originated from a normal (HAM) email message or a spam message (SPAM) [319], [320]. This dataset can be used as a training and testing set in a SPAM or HAM classification problem. Obtain dataset here [73]
81	phishtank	P	R	PHI	N/A	phishtank is a open community where the users can add, search and validate instances of network phsising [321]. Search repository here [74]
82	millersmiles	P	R	PHI	N/A	millersmiles is a repository of phising samples and allows users to add to the collection or search though the repository [322]. Search repository here [75]
83	PWDS-15	P	R	PHI	2456 S	Phishing Websites Data: A collection of 2456 phishing websites [323], [324]. Example features in this dataset: • Long URL, Tiny url or abnormal url. • Special symbols in URL: '@', '//', and '-' • The count of dots in URL • Domain registration date • Protocol information and ports Obtain dataset here [76]

Table 9: Comparative summary of digital forensic datasets

[72] http://trec.nist.gov/data/tweets/
[73] http://www.cs.jhu.edu/~mdredze/datasets/image_spam/
[74] http://www.phishtank.com/phish_archive.php
[75] http://www.millersmiles.co.uk/
[76] http://archive.ics.uci.edu/ml/datasets/Phishing+Websites#

5 Experimental design

5.1 Experiments - overview

A list of the elements that the experiment contains. The elements are further explained in the subsections: (see sections 5.2.1 - 5.2.9)

- Experiments with fulltext searching
- Experiments performed on a set of search engines.
- Set of keywords based on domain knowledge of datasets and Search for strings that is not present in the dataset
- Searching within a index (i.e. not searching across all indexes or multiple indexes at the same time).
- Search time
- Cache temperature
- Memory measurement during search
- Search Accuracy - count of clear cut misses
- Out of box configurations (default values)

The setup for the experiment with respect to:

- environment,
- search engines,
- preprocessing of datasets,
- indexing the datasets,
- and how the search query was formulated

can be seen in section 5.3 - 5.5

Research question

2.b) Which search engine performs the most efficient indexing with respect to the resulting index size and index creation time?

2.c) What is the best search engine with respect to minimizing search time and the number of clear cut misses on searches with single, multi and non-existing search terms?

2.d) Which search engine used the least memory during search?

5.2 Experiments

5.2.1 Experiments with fulltext searching

A decision was made to consider a line (anything before \n) in the datasets as a JSON document [with a single JSON field content] to be imported. All searches will be against the content field. This decision was made as unstructured full text search was the primary focus of this thesis experiments and this import scheme made the import process much easier. The considered alternatives was (1) to identify starting and ending patterns in the datasets representing a document, and (2) to create structured JSON documents with multiple JSON fields of the datasets and perform searches against multiple fields. These alternatives where rejected as they were considered time consuming, non-trivial, and searches against multiple fields would make this approach less of a full text search

experiment.

5.2.2 Selection of search engines for experiments

Due to time constraint the 2 most popular open source search engines (*ElasticSearch* and *Solr* is selected from table 3 to be part of the experiments.

5.2.3 Selection of dataset and search terms for experiments

In the list below is different datasets covering a subset of the categories in table 6. For more information on the datasets cross reference index with I in table 9

1. Fraud: this category is covered by the Enron email dataset obtained from [325](Index=26). We put this email dataset under fraud as the company Enron was involved In one or the largest fraud cases to date.
2. Network: snort IDS log file (Index=59)
3. Email: Hillary Clinton emails (Index=23)
4. Malware: Windows P32 is a dataset that I got access to though my supervisor. The original papers was [326,327]. The feature of the JSON dataset is explained below:
   - MD5 of the file
   - application report, program characteristics, network traffic information and peframe provided by virustotal.
   - Output from the *Linux* file command
   - Output from the *Linux* string command.
5. Spam: DITSSC (Index=74)
6. SMS: NUSSC (Index 72)

The search types that are tested are described in the list below.

1. case: Single term search (e.g. a word or sequence of symbols)
2. case: multiple term search (e.g. sentence)
3. case: searching for something that is not present in the dataset (i.e. MD5 hash of the string "Joachim Hansen" = "02edbd94746bc69677e969a89c4eb0d8").

For full list of all the searches performed see tables 13 and 14 in appendix B.4.

5.2.4 Searching within a index.

All datasets are given their own index and all searches will be performed within these indexes.

5.2.5 Search time

Solr uses the variable *QTime* and *Elasticsearch* uses the variable *took*, which returns the amount of milliseconds it takes to process a response such as indexing or searching [328, 329]. For example, the time it takes to print to console is not considered a part of the *QTime* and *took* measurements. Therefore, with indexing both the application specific response time variables and the *Linux time* command will be used. The time command will show the real time (the actual elapsed time of the indexer, which includes the time waiting for I/O). The time command will not be used when searching as the query would not need to write much to disk (unlike the indexer) and it would be done is a few milliseconds.

5.2.6 Cache temperature - warming up the cache

It is assumed that the applications do some form/level of initial caching of the dataset before any search has been executed on the indexes. The experiments/tests will be per-

formed twice. This should make the cache go from somewhat warm to a bit warmer. It would be interesting to see the potential performance gain from multiple runs of the same tests.

5.2.7 Memory measurement during search

Monitoring memory stats with the top command during indexing was decided against, as both the indexer and top command would compete for the right to write to disk. This would result in a considerable longer index time. There was less issues with monitoring during searching, as most of the searches are done in a couple of milliseconds. While not being an ideal decision, it had to be made due to time constraints.

1. Memory measurement during search (see listings 5.1):
   (a) The command top was used with no added delay (-d 0) which results in NEAR continuous updates/measurements. The top command was initiated manually to measure the performance of *Elasticsearch* (-p 8254) just prior to the searching of the datasets in *ElasticSearch* or *Solr*, the top command ran during the searching and was terminated manually soon after the completion of the search process. The columns for VIRT, RES, Shared,CPU, mem, Time+ in the top output was printed to a .csv file with ; as the delimiter.
2. Remove identical csv records (see listings 5.2)
   (a) The .csv file is preprocessed to only contain unique records (NO duplicate records). This is done to primarily keep records that reflect the performance of the search engine during searching.
3. Concatenation of csv files, removing records with missing values for the 3 first columns and the removal of character g (see listings 5.3)
4. Calculating Average, Max, Min, Delta, and Mode for VIRT, RES, SHARED memory from the concatenated .csv file in excel

Listing 5.1: Top command output to csv

```
top -b -d 0 -p 8254 |
grep "^ " |
awk -v OFS=';'
'{print $5, $6, $7, $9, $10, $11}'
>> /home/search/Downloads/test20.csv
```

Listing 5.2: Remove identical records

```
cat /home/search/Downloads/test20.csv |
awk 'NR%2==0' | awk '!seen[$0] {print}
{++seen[$0]}' >> /home/search/Downloads/d.csv
```

Listing 5.3: Concatenation of csv files and removal of missing observations

```
cat *.csv | awk -F';' 'NF > 3' | sed 's/g//g'
>> /home/search/Downloads/topElasticCat.csv
```

5.2.8 Search accuracy - count of clear cut misses

All query strings with exception with MD5("Joachim Hansen") are supposed to exists with at least 1 occurrence in the respective dataset. This is enforced by exacting a single term or multiple terms while going through the datasets manually. A clear cut miss is therefore defined as a non-MD5 search that results in 0 search hits.

5.2.9 Out of the box configurations/default values

The comparison of the search engines *Solr* and Elasticsearch are not done with the same application configurations. But with the out of the box settings (i.e. default values) in *Solr* and *Elasticsearch*. The thesis author sees the importance of benchmarking applications using multiple sets of configurations to avoid testing one application with sub-optimal configurations against one with near optimal settings. Due to time constraints, finding the fairest comparison is outside the scope of this paper. With that in mind comparing the software as is, without modifying them could be considered fair in this setting, as these are the settings set by the author of the program. A more thorough discussion about possible additional experiments can be seen in the future work. The way that the dataset are preprocessed, the search queries, import process/indexing, ways of measurements, are identical with exception to some application specific formats and API calls.

5.3 Setting up the environment

OpenJDK Version 1.8.0_131 was installed as a dependency for Elasticsearch. The rest of the installation was either a dependency for the search engines or to get the code used in this thesis to work properly.

1. sudo apt-get update
2. sudo apt-get install default-jdk
3. sudo apt-get install apt-transport-https
4. sudo apt-get install mysql-client unixodbc libpq5
5. sudo apt-get install mysql-server
6. sudo apt-get install gawk
7. sudo apt-get install php-curl

5.3.1 Environment - specifications

Virtual machine

- OS: *Ubuntu* 64 bit
- OS version: *Ubuntu* 16.04.3 LTS
- CPU/Cores: 6
- RAM: 40960 MB
- Storage: 2048 GB / 2 TB

How I interface with the virtual machine:

- Application 1: *VMware vSphare web client*
- Application 2: Google chrome Version 56.0.2924.87 m
- Application 3: Flash player 27 Beta installer (crashed with previous version)
- Plugin: *VMware Client intregation plug-in 6.0.0*

5.4 Setting up the search engines

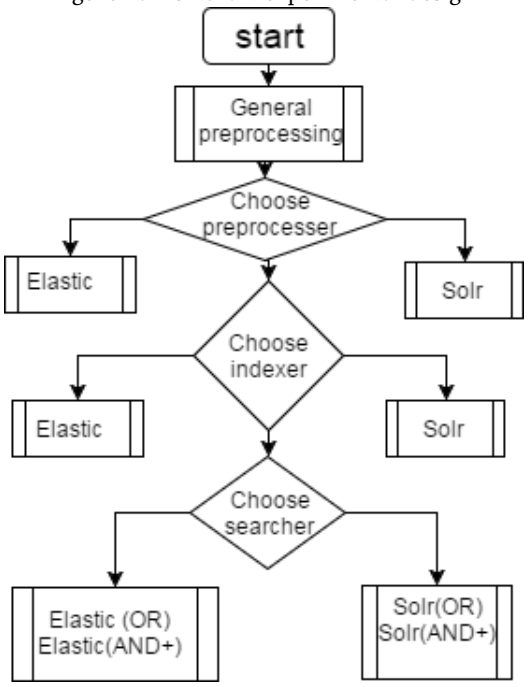
Order of instalment:

1. *Openjdk 1.8.0_131*
2. *Elasticsearch 6.0.1*
3. *Solr 7.1.0* and *Solr Cloud*

The following tutorials are used for installing *Elasticsearch* and *Solr* [330–333]

5.5 Dataset preprocessing, Indexing and search query

Figure 1: Flowchart experimental design



A short description of how the process of preprocessing, indexing and searching in the experiment are given here. As you can see in figure 1 the first step is running the datasets though a general preprocessing step. This step does:

- Remove *JSON* structure, *XML* structure and/or dataset specific structure from all the datasets.
- Removes characters that cannot be processed by the *JSON* parser in *Solr* or *Elasticsearch*.
- Removes junk (e.g. terms that is not interesting, such as many repeated characters)
- Removes unnecessary whitespace
- Removes empty lines
- Separate dataset entries on their own lines.
- Split the dataset into **bulk files** of 1000 lines each(each line is considered as a document for importing).

Then after general preprocessing the dataset bulk files have to go through two separate preprocessing steps, one for *Elasticsearch* and one for *Solr*. The Elasticsearch preprocessing step tries to get the bulk files in the following JSON format:

{ "index":{} }
{"content":"bulk file line n" }
{ "index":{} }
{"content":"bulk file line n+1" }
"empty line"

And *Solr* to get the bulk files in this *JSON* format:

[
{"content":"bulk file line n" }
{"content":"bulk file line n+1" }
]

A single content field entry is 1 *JSON* document to be indexed. The output from these steps I call bulk *Elastic* and bulk *Solr* respectively. The next 2 steps are *Elastic* and *Solr* specific bulk indexer. These indexer takes the *Elastic* and *Solr* bulk files as input and indexes all of the bulk files/the entire dataset. At these steps, the following measurements are taken:

- The *Linux command time* is used to capture total elapsed time of indexer.
- *QTime* in *Solr* and *took* in *Elasticsearch* are used as response time.
- A size command in *Elasticsearch* are used to get the size of the index. And the filesystem in my environment is inspected to find the index size in *Solr*.

The last two steps are searching with Solr and *Elasticsearch*. The Solr(OR)/Elastic(OR) and Solr(AND+)/Elastic(AND+) are similar. The (OR) search queries are using the boolean OR operator between multiple search terms. The (AND+) search queries requires that all search terms in the search string are present in the matching string, and that the terms are in the right order in order for their being a match. At these steps, the following measurements are taken:

- Clear cut misses
- Memory stats are captured with the *Linux top* command.
- *QTime* and *took* response time.

5.5.1 Dataset preprocessing

windows-p32 dataset

Order of operations:

1. Acquired dataset sql dump in raw format
2. Created database dumper2 in mysql
3. Imported sql dump as batch to mysql and database dumper2
4. Outputted a comma separated .csv dump of the rawData content
5. Moved csv dump from restricted folder using root privileges.
6. General preprocessing of the .csv file
7. Remove lines with less than 21 characters
8. Additional preprocessing
9. Split the file into multiple files of 1000 lines each.
10. *Elasticsearch* preprocessing
11. *Solr* preprocessing

Acquired dataset:

I acquired the malware dataset from Andrii Shalaginov which is one of my PhD supervisors. The dataset was a sql dump in raw format. In order to better deal with the dataset I decided to restore the database by importing it to mysql and then dump it as a comma separated .csv file.

Created database:

Listing 5.4 shows how the database dumper2 was created.

Listing 5.4: Database creation

```
mysql > create database dumper2;
```

Importing database

The dataset dump I was given was zipped. After unzipping I moved all unzipped .sql files to one folder for them-self. And renamed the name of the schema file to be before all other files in alphabetic order. The other files are already in alphabetic order (the order in which I will restore them). I used the code created by [334] in listing 5.5 to pipe an alphabetic sorted list of .sql file names to mysql and import it in batch to my dumper2 database.

Listing 5.5: Importing database

```
ls -1 *.sql | awk '{ print "source",$0 }' |
 mysql --batch -u root -p dumper2
```

Export database table contents to .csv file

The sql code from exporting my data to .csv in listings 5.6 was found on [335]. I select all fields from the rawData table in dumper2 database (the only table in the database). The fields are then printed within /quotation marks/, separated with comma and the new line break symbol is \n

Listing 5.6: Exporting database content

```
SELECT * FROM rawData
INTO OUTFILE '/var/lib/mysql-files/dumper2.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
```

Moved file from restricted folder

The dataset was outputted to /var/lib/mysql-files/ which is the only valid output folder due to a secure-file-priv variable. This folder is restricted and therefore I decided to use elevated privileges in listing 5.7 to move the dataset.

Listing 5.7: Moving dataset

```
sudo -i;
cp /var/lib/mysql-files/dumper2.csv
/home/search/Downloads/Datasets/dumper2.csv;
exit;
```

General preprocessing:

The listings 5.8 was used to clean up the dataset from junk sequences and characters significant for *JSON* such as square brackets, curly brackets and quotation marks. This was removed in order to prevent conflict with the *JSON* parser/importer in *Solr* and *Elasticsearch*. Some of the cleaning procedures will unfortunately affect legitimate data fields such as urls and hash values. The listings are explained in the list below in the order of operations/Pipe order.

1. Convert *Windows* or *MAC* carriage return and newline characters to *Linux*.
2. *sed*: Remove starting with \then followed by " then followed by ; or : and any sequence of character until a " character... then abort.

3. *sed*: Substitute character(s) with space
   (a)],\
   (b) },
   (c) }\
   (d) {\
   (e) ":
   (f) \"
   (g)]\
   (h) [\
   (i) ,\
   (j) <space>\
   (k)]"
   (l)]
   (m) [
   (n) "
   (o) :
   (p) {
   (q) }
4. sed: Remove empty lines
5. sed: Remove consecutive spaces

Listing 5.8: Cleaning up the dataset

```
cat dumper2.csv | dos2unix | sed -e 's/\\\";[^"]*//g'
-e 's/\\\":[^"]*//g' |
sed -e 's/\],\\/ /g' -e 's/\},/ /g' -e 's/\}\\/ /g'
-e 's/{\\/ /g' -e 's/\":/ /g' -e 's/\\\"/ /g'
-e 's/\]\\/ /g' -e 's/\[\\/ /g' -e 's/,\\/ /g'
-e 's/ \\/ /g' -e 's/\]\"/ /g' -e 's/\]/ /g'
 -e 's/\[/ /g' -e 's/"/  /g' -e 's/://g'
 -e 's/{/ /g' -e 's/}/ /g' | sed '/^\s*$/d'
 | sed 's/ \+/ /g'
```

Removing lines less then 21 characters

I tried experimenting with various batch sizes and concluded that 1000 JSON documents worked best as my batch size for indexing at the same time. The dataset was split into multiple files of 1000 lines each (1000 JSON documents). But before splitting, it was decided to remove all lines shorter then 21 characters, in order to reduce the size of the dataset. This was done in order to reduce the overhead from indexing this dataset, and because a lot of these shorter lines where uninteresting for searching purposes. I used the code from [336] in listings 5.9 to remove lines less then 21 characters, and to write a new file without these lines. By removing all lines less then 21 characters I reduced the dataset size from 17.6GB to 6.6GB

Listing 5.9: Removing lines

```
cat file | gawk 'length($0)>20' >> newfile
```

Additional preprocessing:

The listings 5.10 removes non ascii characters, removes non printable ascii characters except for new line and then escapes \with \\.

Listing 5.10: Removing lines

```
cat /home/search/Downloads/Datasets/
dumper2NoLinesLessthen21Char2 |
perl -pe 's/[^[:ascii:]]//g;' |
tr -cd '\12\40-\176' |
sed -e 's/\\/\\\\/g'
>> /home/search/Downloads/dumpertrLineFeedOnly
```

Split the file:

The listings 5.11 splits the dataset into multiple sub files of 1000 lines each. Each sub file will be named with the specified prefix with a numerical suffix (e.g. "prefix00").

Listing 5.11: Split the file up in multiple files of 1000 lines each

```
cat dumpertrLineFeedOnly |
 split -l 1000 -d /outputfolder/prefix
```

Snort IDS logs ID 59

1. General preprocessing (for full general preprocessing see appendix A.1)
2. *Solr* and *Elasticseach* specific preprocessing

General preprocessing:

Most steps in listings A.1 is explained in section 5.5.1. The newly added commands in listings A.1 can be seen in listings 5.12 and are explained in the list below:

1. substitute all newlines characters with the space character using xargs command.
2. At this point there are no new lines in the *Snort* dataset. I identified that each entry in this file starts with the character sequence "[**] [1". Before this character sequence a newline character is added. Now each entry are on their separate lines.
3. Substitute * with space.
4. - character in split tells it to work on the content being piped to it.

Listing 5.12: Newly added commands

```
xargs | sed 's/\[\*\*\] \[1/\n\[\*\*\] \[1/g' | sed 's/\*/ /g'
```

Hillary Clinton emails

Preprocessing steps:

1. General preprocessing (for full general preprocessing see appendix A.2)
2. *Solr* and *Elasticseach* specific preprocessing

Most of the preprocessing steps in listings A.2 have been explained in section 5.5.1 and 5.5.1. The newly added steps can be seen in listings 5.13. I used the command from [337] to replace all newline characters in the file. Then a pattern with the unicode character \u000C followed by the quote character ", was identified. This pattern occurs at the end of each email entry and was used to create a newline each time this pattern occur to have each email entry on their separate lines.

Listing 5.13: Newly added preprocessing

```
sed ':a;N;$!ba;s/\n/ /g'  |
perl -pe  "s/$(echo -en '\u000C')\"/\n/g"
```

Enron email dataset

1. General preprocessing (for full general preprocessing see appendix A.3)
2. *Solr* and *Elasticseach* specific preprocessing

Efforts was made to ensure that each email entry was on their own line by adding a line before "Date: " (see listings 5.14).

Listing 5.14: Newly added commands: Enron emails

```
perl -pe "s/Date: /\nDate: /g"
```

NUSSC sms dataset

1. General preprocessing (for full general preprocessing see appendix A.4)
2. *Solr* and *Elasticseach* specific preprocessing

New commands in listings 5.15 are described in list below.

1. Add a sequence of a's to the end of each message.
2. Remove all XML tags (command is from [338])
3. Substitute sequence of a's with newline
4. Remove pattern

Listing 5.15: Newly added commands: Enron emails

```
sed 's/<\/message>/<\/message>aaaaaaa/g' |
sed -E 's/<[^>]+>/ /g'|
perl -pe 's/aaaaaaa/\n/g' |
perl -pe 's/&amp//g'
```

DITSSC spam dataset

1. General preprocessing (for full general preprocessing see appendix A.5)
2. *Solr* and *Elasticseach* specific preprocessing

5.5.2 Index size

Elasticsearch

I used the command from [339] in listings 5.16 to get a list sorted on size that contains the name of the index, the number of documents in a index and the index size.

Listing 5.16: Index size Elasticsearch

```
curl -XGET 'localhost:9200/_cat/indices?
v&h=i,docs.count,tm&s=tm:desc&pretty'
```

Solr

To get the index size in *Solr* a filesystem search was done to find the directory containing the specific index and use the size of this directory as the index size. For example searching for "windows-p32" to locate the directory windows-p32_shard1 _replica_n1 with size 23.2GB.

5.5.3 Index creation time and indexing process

Elasticsearch

A script called ElasticSearchIndexBatch.sh was created to automate the process of indexing each bulk of 1000 json documents with Elasticsearch. The script needs to be run from the same directory as the batch files (already preprocced bulk files) ready to be imported. The script is run with the *Linux time* command (see listings 5.17) and with the

index name as the only command line argument. The script ElasticSearchIndexBatch.sh (see listings 5.18) process all files in in the current directory one by one, post the file to elasticsearch with port 9200 and with index name from the command line. All documents get the same document type regardless (somedoctype) of index. In this thesis, only searching within single index [at the time] is considered and not within a given document type. For each bulk of 1000 document indexed by Elasticsearch the response time are captured using grep and the numbers are extracted with tr. To capture the response value correctly the *?pretty* option is used to get the response time (i.e. "took":) to its own line. The response time is then added to the sum. At the end of script the final sum for response is echoed. The -s option was also used on curl to silence some extra verbose output, without this -s option getting the *took* value or the *QTime* might not be possible. A new index is crated with the default number of shards (i.e. 5) and default number of replicas (i.e. 1) [340]. This numbers was also independently verified using for example:

curl -XGET 'localhost:9200/snort-logs?pretty'

Listing 5.17: Calling the bulk index script for elasticsearch

```
time /home/search
/Downloads/Datasets/ElasticSearchIndexBatch.sh
windows-p32
```

Listing 5.18: Automate bulk indexing and sum the total milliseconds response time (ElasticSearchIndexBatch.sh)

```
#!/bin/bash
# have to provide index name
# in command line argument 1 (first and only argument)
# index name have to be in lowercase
# Have to be run in the same current
#working folder (PWD) as the batch files to be
# preproccesed
# Need ?pretty to get took in line for itself
# need to write i=$(commands) and not i = $(commands)
currDir=$(pwd);
indexname=$1;
sumTook=0;
declare -i took;
for f in $(ls -p | grep -v '/');
do
fileToBePreproccesed="$currDir/$f";
took=$(curl -s -H "Content-Type:
application/x-ndjson" -XPOST
localhost:9200/$indexname/somedoctype/
_bulk?pretty --data-binary
@$fileToBePreproccesed | grep 'took' | tr -dc '0-9');
((sumTook += $took));
done
# (($sumTook += $took)); was apparently incorrect
echo "Took in total: $sumTook";
```

Solr

Most of the steps in 5.19 have been explained in section 5.5.3. The most important thing to emphasize is that the collection name or index name have to be created in the

browser (e.g. firefox) in localhost:port (e.g. port=8985) prior to running the script in listings 5.19. This was not a problem with Elasticsearch as it generated automatically a new indexname with your indexname in listings 5.18 if the indexname did not exist before. In *Solr* the collection name is created with the Add collection button with default values (1 for numShards and 1 replication factor) and _default as configName.

Listing 5.19: Solr automating bulk indexing

```
#!/bin/bash
# have to provide index name in
# command line argument 1 (first and only argument)
# index name have to be in lowercase
# Have to be run in the
# same current working folder #(PWD) as the batch files
# to be preproccesed
# Need ?pretty to get took in line for itself
currDir=$(pwd);
indexname=$1;
sumTook=0;
declare -i took;
for f in $(ls -p | grep -v '/');
do
fileToBePreproccesed="$currDir/$f";
took=$(curl -s -XPOST
http://localhost:8985/solr/
$indexname/update?commit=true
 --data-binary @$fileToBePreproccesed
 -H 'Content-type:application/json'
   | grep "QTime" | tr -dc '0-9');

((sumTook += $took));

done
# (($sumTook += $took)); was apparently incorrect
echo "QTime in total: $sumTook";
```

5.5.4 Search engine preprocessing

Elasticsearch

Calling the *Elastic* prepossessing script is similar to calling the *Solr* prepoccing script in subsection 5.5.4 (except for the script name). The script in listings 5.20 goes though all files in the current directory (that should already went through general preprocessing) then make sure that each line (*JSON* document) gets the prefix {"content": " and the suffix "}. The *JSON* field content is the only field in the *JSON* document. A new line is inserted before each of these lines/documents including the first line, with the string {"index" : {} }. Then at the end of the file a new empty line is added to represent the end of the bulk. Example bulk file for indexing only:

{ "index":{} }
{"content":"Document example 1" }
{ "index":{} }
{"content":"Document example 2" }
"empty line"

Listing 5.20: Automate elasticsearch spesific preproccing

```
#!/bin/bash
# Have to be run as full path to output folder (ending
#with /) as command line argument 1
# Have to be run in the same current working folder
#(PWD) as the batch files to be preproccesed
currDir=$(pwd);

for f in $(ls -p | grep -v '/');
do
fileToBePreproccesed="$currDir/$f";
outputPath="$1$f";
cat $fileToBePreproccesed |
 sed -e 's/^/{"content":"/' |
  gawk 'NF{print $0 " \"}"}' |
   gawk ' {print;} NR % 1 == 0
   { print "{ \"index\":{} }"; }' |
    gawk 'BEGIN{print "{ \"index\":{} }";}{print;}' |
    head -n -1 |
    gawk 'END { print "";}{print;}' >> $outputPath;
done
```

Solr preprocessing

Listings 5.21 shows an example for how listings 5.22 is called. The script have to be called from the same current directory as the files being preprocessed for *Solr* and with the full path to the new directory for the preprocessed *Solr* files. Example bulk file:

```
[
{"content":"Document example 1" }
{"content":"Document example 2" }
]
```

Listing 5.21: Calling the Solr preprocessing script script

```
/home/search/Downloads/Datasets/SolrBatchPreproccing.sh
/home/search/Downloads/Datasets/batchMalwareSolr/
```

Listing 5.22: Solr bulk preprocessing

```
#!/bin/bash
# Have to be run as full path to output folder (ending
with /) as command line argument 1
# Have to be run in the same current working folder
#(PWD) as the batch files to be preproccesed
currDir=$(pwd);
for f in $(ls -p | grep -v '/');
do
fileToBePreproccesed="$currDir/$f";
outputPath="$1$f";
cat $fileToBePreproccesed |
sed -e 's/^/{"content":"/' |
awk 'NF{print $0 " \"},"}' |
awk 'BEGIN{print "[";}{print;}' |head -n -1 |
awk 'END { print "]";}{print;}' >> $outputPath;
done
```

5.6 Search query

5.6.1 Elasticsearch example

Listing 5.23: Elasticsearch full text match query

```
curl -XGET 'localhost:9200/
batch-andrii-malware2/_search?
pretty' -H 'Content-Type: application/json' -d'
{
    "query": {
        "match" : {
            "content" :
            "Trojan-Clicker.Win32.Agent.acu"
        }
    }
}
' >> /home/search/Downloads/outputElastic/$1
```

The second search query used with Elasticsearch is similar to that of listings 5.23 with the exception being that match is replaced with match_phrase. This changes the behavior of the search from just performing the OR operation between search terms to requiring both all terms to be present as well as taking the ordering of the terms into account when matching.

5.6.2 Solr example

The default request handler /select is used for searching [341] (see listings 5.24. For single term search only one term will be present in the search string content:\"term1\". When multiple search terms are used the + character is used to indicate a space between terms. All the terms have to be present and be in the right order, to be qualified as a match. The second search in listing 5.25 matches any document that has either the term your or the term subscription or both when searching in the content field.

Listing 5.24: Solr phrase search

```
curl -XGET localhost:8985/solr/mal3/
select?q=content:\"warning+thumbnail\"
```

Listing 5.25: Solr OR search

```
curl -G 'localhost:8985/solr/
batch-spam/select?rows=0&wt=json'
--data-urlencode
q='{!q.op=OR df=content}your subscription'
```

5.7 Changes to the original testing plan

It was originally intended to test Solr, Elasticsearch and Sphinx. The risk management plan created in this project had defined a maximum resource limits for the different deliveries in the master thesis. The time taken to make Sphinx work properly was too long to include it in the experiments.

6 Results and analysis

6.1 Dataset line length

6.1.1 Output

Information regarding the line lengths (length in number of characters) of the datasets used in the experiments can be seen in table 10. This information is created as to better understand the influence of number documents and documents size on the index creation time and resulting index size. This table was generated from applying a script that can be found in appendix B.1.1 on the datasets. The contents of this table includes the file size, average line lengths and different frequencies of line lengths, and these measurements was taken after general preprocessing was performed and prior to Solr/Elastic preprocessing was applied to the datasets.

Table 10: Number of lines in the datasets after general preprocessing

windows-p32	Snort logs ID 59	Hillary clinton emails	Enron Email dataset	ID 72 NUSSC SMS dataset	ID 74 DITSSC SPAM dataset
• size: 6,7 GB • Average: 148 • Total: 45117458 • 0-9: 22666(*) (empty lines) • 11-100: 42611043 • 101-500: 1486838 • 501-1000: 259446 • 1001-5000: 324490 • 5001-10000: 339038 • 10001-50000: 71357 • 50001-100000: 1049 • 100001-500000: 1107 • Longer lengths: 424	• size: 10 MB • Average: 386 • Total: 25820 • 0-9: 2 • 11-100: 33 • 101-500: 22822 • 501-1000: 2963 • 1001-5000: 0 • 5001-10000: 0 • 10001-50000: 0 • 50001-100000: 0 • 100001-500000: 0 • Longer lengths: 0	• size: 25.3 MB • Average: 3178 • Total: 7944 • 0-9: 0 • 11-100: 0 • 101-500: 7 • 501-1000: 1806 • 1001-5000: 5030 • 5001-10000: 711 • 10001-50000: 361 • 50001-100000: 26 • 100001-500000: 3 • Longer lengths: 0	• size: 1.4 GB • Average: 2438 • Total: 554800 • 0-9: 2 • 11-100: 661 • 101-500: 32645 • 501-1000: 156302 • 1001-5000: 316662 • 5001-10000: 33564 • 10001-50000: 13924 • 50001-100000: 692 • 100001-500000: 339 • Longer lengths: 9	• size: 12.5 MB • Average: 233 • Total: 55849 • 0-9: 20 • 11-100: 8652 • 101-500: 46820 • 501-1000: 349 • 1001-5000: 8 • 5001-10000: 0 • 10001-50000: 0 • 50001-100000: 0 • 100001-500000: 0 • Longer lengths: 0	• size 218.3 KB • Average: 160 • Total: 1353 • 0-9: 0 • 11-100: 102 • 101-500: 1251 • 501-1000: 0 • 1001-5000: 0 • 5001-10000: 0 • 10001-50000: 0 • 50001-100000: 0 • 100001-500000: 0 • Longer lengths: 0

6.2 Indexing performance

Figure 2 compares the *took* and *QTime* response time variables from 2 runs of the *Elastic/Solr* indexer on the experiment datasets. The figure shows a first run [in blue] and a second run [in red] for *Elasticsearch*, a separator, followed by a first and second run [Orange and green] with *Solr* for each dataset. The figure also present a table containing the response time in ms. The figure is logarithmically scaled with 2 as base as to be able to present both small and large values.

Figure 3 compares the real time (the total elapsed time of the indexer) with delta (delta = real time - response time), which is a measurement of how much of real time is spent on I/O. The figure presents 2 runs of real time and delta for *Elasticsearch* and *Solr* with a [white/blank/empty] separator between. This figure is also logarithmically scaled with 2 as base, have values in ms and a table summary.

Figure 4 is the aggregate representation of the table data in figure 2 and 3. Figure 4 have values in ms and is in normal scale. The figure shows how real time, response time and delta relate to one another.

Figure 5 shows the change in size from the size of dataset from after general preprocessing (see appendix B.3 and table 12 to see the size of dataset after general prepro-

cessing) to the resulting index size. The delta is defined in this context as:

general preprocessing dataset size - resulting index size

Delta is then presented in 2 runs in both *Elasticsearch* and *Solr* as percentage change. If the change is a size reduction, then the percentage will be shown as a negative number.

Figure 2: Index time Took and QTime

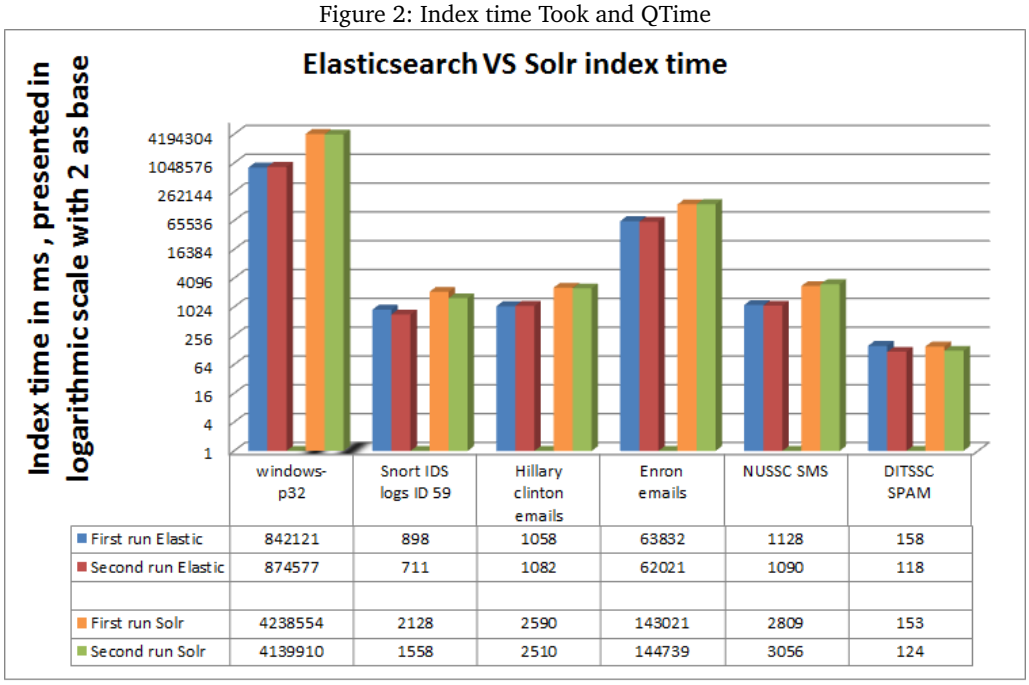
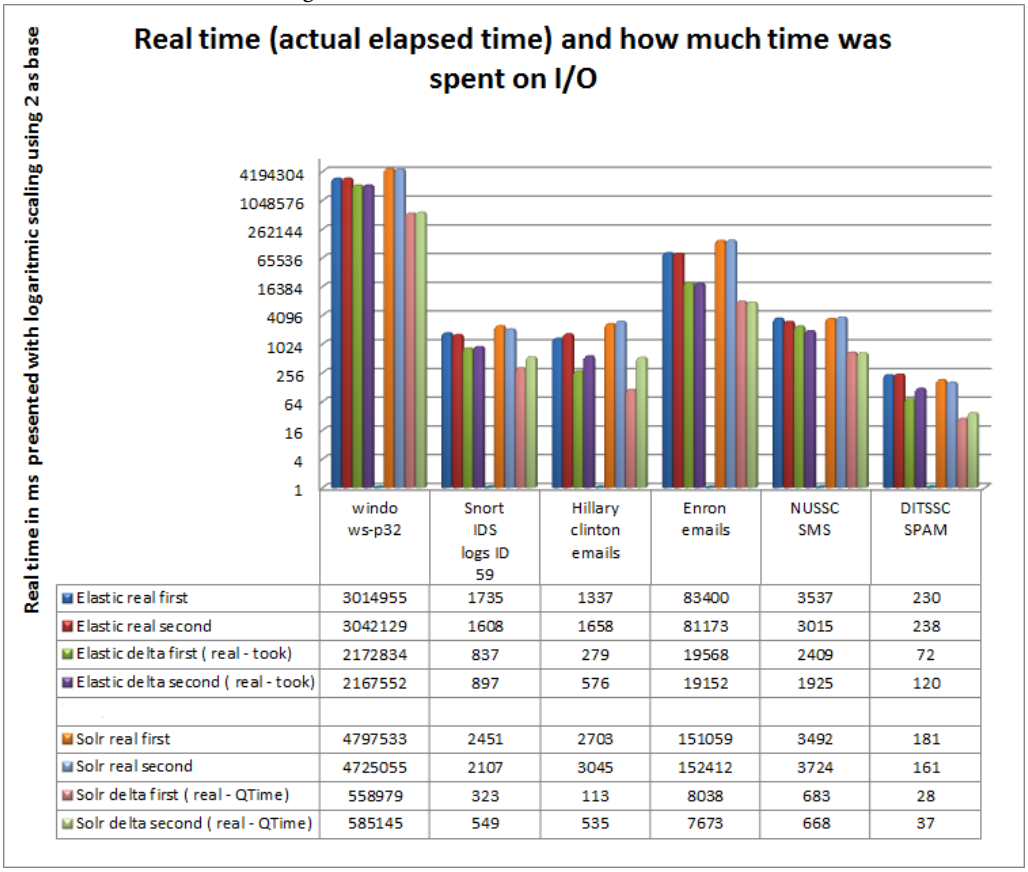


Figure 3: Index time real time and I/O time



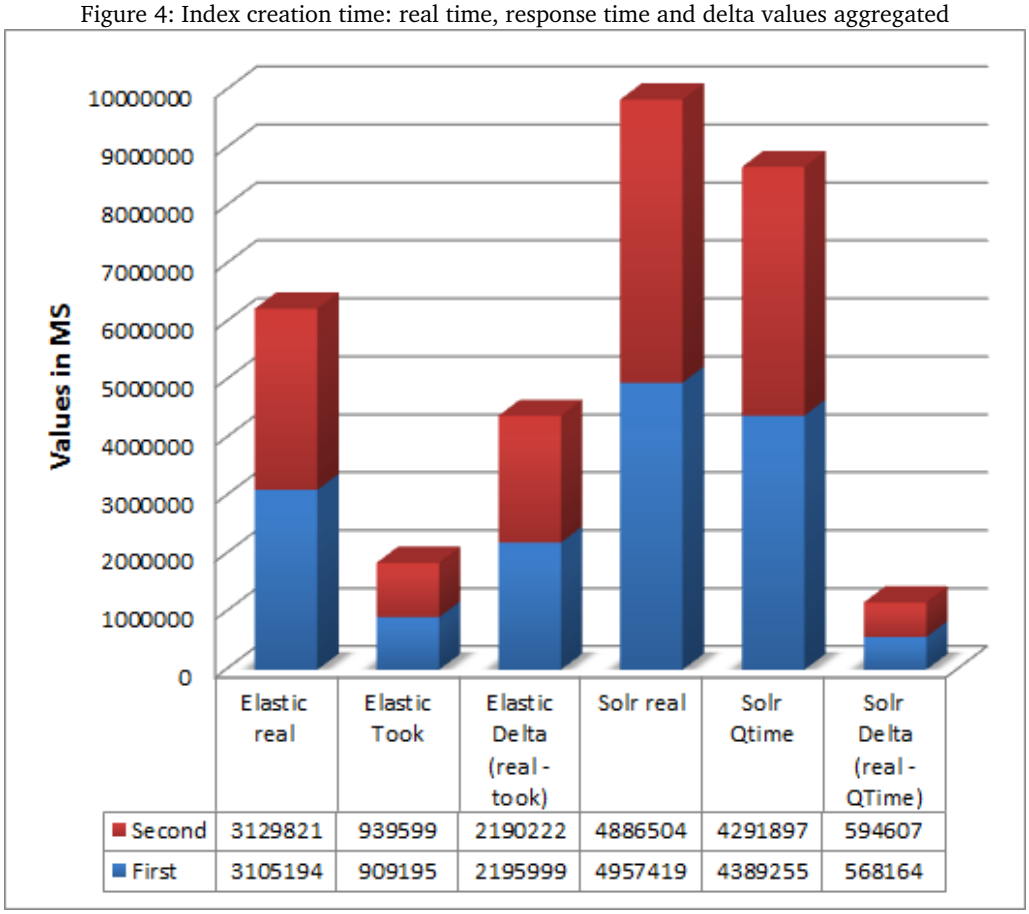
	windows-p32	Snort IDS logs ID 59	Hillary clinton emails	Enron emails	NUSSC SMS	DITSSC SPAM
Elastic real first	3014955	1735	1337	83400	3537	230
Elastic real second	3042129	1608	1658	81173	3015	238
Elastic delta first (real - took)	2172834	837	279	19568	2409	72
Elastic delta second (real - took)	2167552	897	576	19152	1925	120
Solr real first	4797533	2451	2703	151059	3492	181
Solr real second	4725055	2107	3045	152412	3724	161
Solr delta first (real - QTime)	558979	323	113	8038	683	28
Solr delta second (real - QTime)	585145	549	535	7673	668	37

Analysis of Findings

1. The index response time in figure 2 favours *Elasticsearch* greatly over *Solr* for both runs with exception of the DITSSC SPAM dataset
2. The The index response time in figure 2 is not fixed for both runs and are not a consistent reduction in response time from run 1 to run 2.
3. We can see in figure 3 that real time and response time are proportionally more close in *Solr* then in *Elasticsearch*. More time where therefore spent on delta in *Elasticsearch* than *Solr*. This is further discussed in section 7.2.
4. A clear difference between *Solr* and *Elasticsearch* with respect to index size can be seen in figure 5:
   - Consistently we see that the index size in *Solr* is increased from the preprocessed dataset size, while decreased in Elasticsearch.
   - When looking at both the figure 5 and the distributions of line lengths with regard to the datasets in table 10, we see that index sizes in *Solr* increases the most with high amount of lines of few characters.
   - Also given the same identical preprocessed dataset the index size is not fixed for run 1 or 2 in either *Solr* or *Elasticsearch*.

Discussion of findings is continued in section 7.2.

Figure 4: Index creation time: real time, response time and delta values aggregated



6.2.1 Query performance

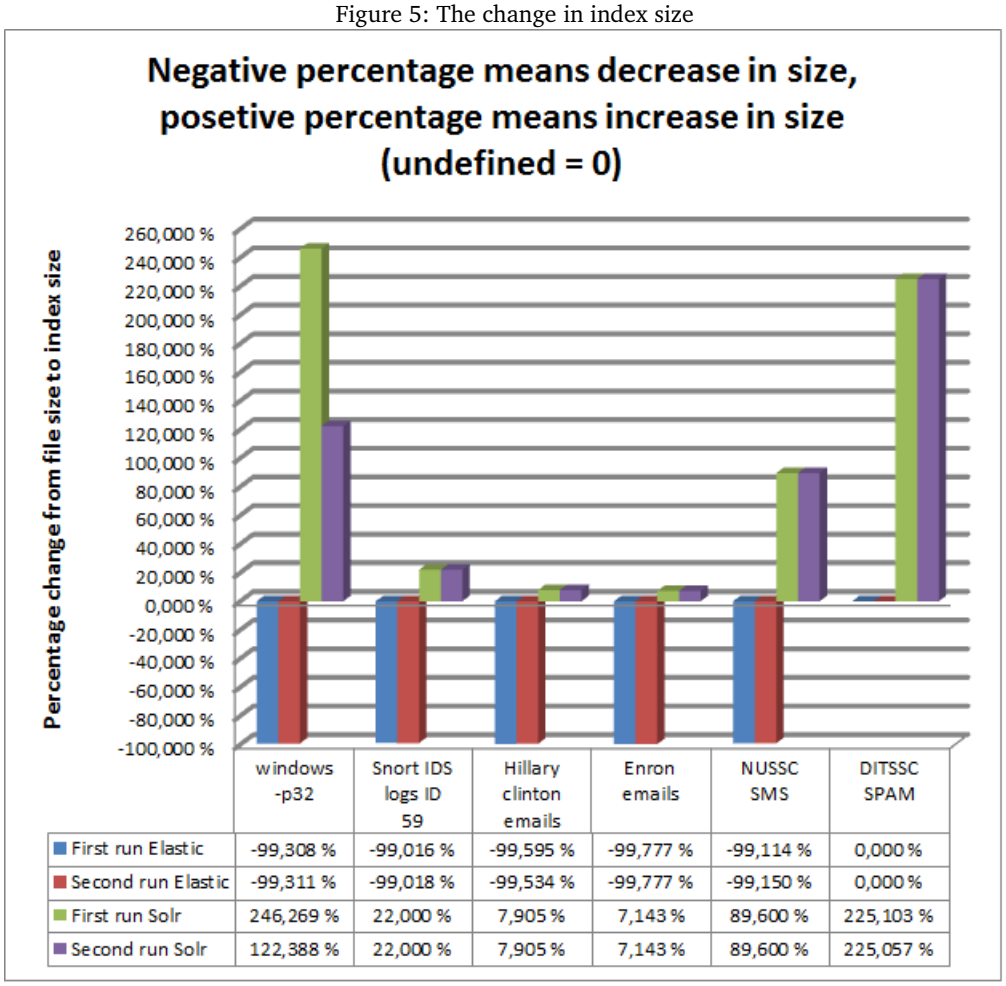
Figure 6 shows the response time for the first and second run of the same search query in ms. The figure is logarithmically scaled with 2 as base. The search queries where divided into the following categories:

- single term Elastic,
- single term Solr,
- multi term search with search function Elastic (OR),
- multi term search with search function *Solr* (OR),
- multi term search with search function Elastic (AND+),
- multi term search with search function *Solr* (AND+),
- not present Elastic,
- not present Solr,
- aggregate time Elastic,
- and aggregate time Solr.

Figure 6 also shows a table with the aggregated search response times.

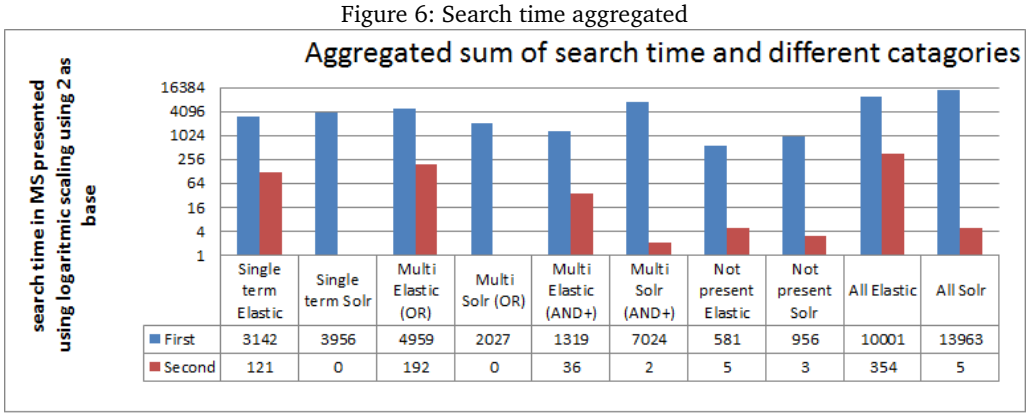
Figure 7 shows the aggregated search hits for the same search categories as figure 6. Figure 7 is scaled normally and shows a table view of the search hits as well.

Figure 5: The change in index size



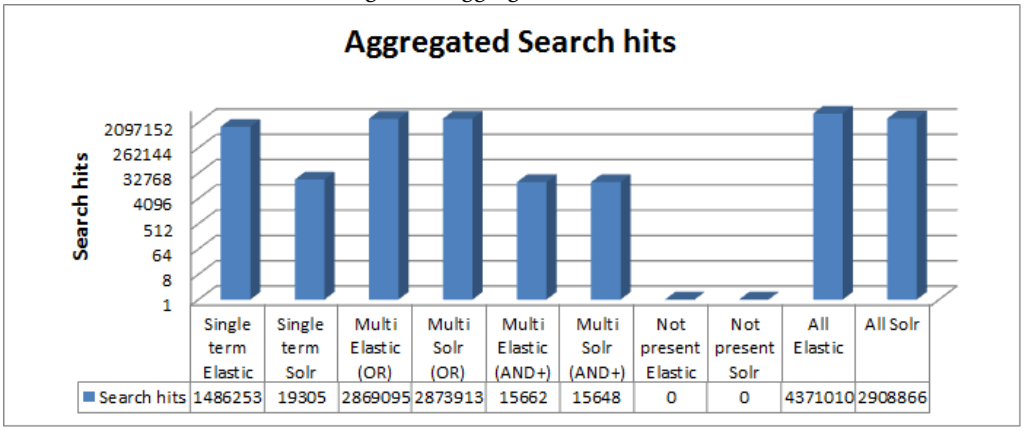
	windows -p32	Snort IDS logs ID 59	Hillary clinton emails	Enron emails	NUSSC SMS	DITSSC SPAM
First run Elastic	-99,308 %	-99,016 %	-99,595 %	-99,777 %	-99,114 %	0,000 %
Second run Elastic	-99,311 %	-99,018 %	-99,534 %	-99,777 %	-99,150 %	0,000 %
First run Solr	246,269 %	22,000 %	7,905 %	7,143 %	89,600 %	225,103 %
Second run Solr	122,388 %	22,000 %	7,905 %	7,143 %	89,600 %	225,057 %

Figure 6: Search time aggregated



	Single term Elastic	Single term Solr	Multi Elastic (OR)	Multi Solr (OR)	Multi Elastic (AND+)	Multi Solr (AND+)	Not present Elastic	Not present Solr	All Elastic	All Solr
First	3142	3956	4959	2027	1319	7024	581	956	10001	13963
Second	121	0	192	0	36	2	5	3	354	5

Figure 7: Aggregated search hits



Analysis of Findings

1. *Solr* and *Elasticsearch* has both 2 clear cut misses, meaning that no search hits was found when there is atleast one matching occurrence of the search string present in the dataset. See search strings:
   - "Error writhing temporary file. Make sure your temp folder is valid"
   - "AFGAN AID"

   in table 13 in appendix B.4 I note that all clear cut misses was from AND+ functions.
2. see figure 6:
   (a) *Solr* has 25.91% increase in search time over *Elasticserch* with respect to single term search.
   (b) *Elasticsearch* has a 144.65% increase in search time with the OR search function over *Solr*.
   (c) *Solr* has a 432.52% increase in search time with the AND+ search function over *Elasticsearch*.
   (d) *Solr* has 64.54% increase in search time with the "not present" search category over *Elasticsearch*.
   (e) Overall for the first searches *Solr* has a 39.62% increased search time over *Elasticsearch*.
   (f) Overall *Elasticsearch* saw an 96.46% reduction of search time (aggregate all categories) from first to second run.
   (g) Overall *Solr* saw an 99.96% reduction of search time (aggregate all categories) from first to second run.
   (h) *Elastic* has a 6980% increase of search time as a aggregate for the second run over *Solr*.
3. See figure 7
   (a) Largely with most categories has just a small difference between the number of search hits in *Solr* and *Elasticsearch*.
   (b) One big exception is with Single term search where *Elastic* had 7598.8% increased number of search hits over *Solr*.

Discussion of findings is continued in section 7.2.

6.2.2 Memory performance during searching

The memory stats in table 11 are captured just prior to initiating a search quary, during first and second attempt of the same search and the capturing process is ended just seconds after the searches are done. There was in total 84 searches (44 for *Solr* and 44 for *Elasticsearch*) that make up the stats in this table.

Table 11: Memory stats Elasticsearch and Solr during search

Elasticsearch					**Solr**				
Virtual memory(VIRT): Size in GiB					Virtual memory(VIRT): Size in GiB				
Average	Max	Min	Delta	Mode	Average	Max	Min	Delta	Mode
42,831	42,831	42,831	0	42,831	29,144	29,144	29,144	0	29,144
Physical memory (not swappable) - RES:size in GiB					Physical memory (not swappable) - RES: size in GiB				
2,807	2,898	2,666	0,232	2,898	2,936	2,964	2,881	0,083	2,964
shared memory (SHR): size in GiB (rounded up)					shared memory (SHR): size in GiB				
0,34	0,43	0,2	0,23	0,43	2,296	2,323	2,241	0,082	2,323

Analysis of Findings

1. see figure 11
   (a) *Elactisearch* used 13,687 GiB more VIRT then *Solr* (average).
   (b) *Solr* used 0,129204022 GiB more on RES then *Elasticsearch* (average).
   (c) *Solr* used 1,955766102 GiB more on Shared then *Elasticsearch*(average).

Discussion of findings is continued in section 7.2.

7 Conclusion and Discussion

7.1 Conclusion

Research questions and answers for the practical research questions:

2.a) What are the publicly available forensic related datasets for testing?
- The table 7 shows a small summary of the findings in chapter 4. To the best of my knowledge the summarized findings in chapter 4 is one of the most comprehensive list/survey of forensic related dataset available.

2.b) Which search engine performs the most efficient indexing with respect to the resulting index size and index creation time?
- *Elasticsearch* search time was faster than *Solr* for 11 out of 12 trials of the indexer. The one exception was the first run of the smallest dataset.
- In *Elasticsearch* the resulting index size is reduced from the original dataset by around 99%, while we see an increase in index size in *Solr* ranging from a few percentage to up too around 240%.
- The null hypothesis can be tossed away, as there is significant difference in favour of *Elasticsearch* over *Solr* with regard to both index size and index creation time.

2.c) What is the best search engine with respect to minimizing search time and the number of clear cut misses on searches with single, multi and non-existing search terms?
- *Elasticsearch* is better then *Solr* at 3 out of 4 search categories for the first run, but *Solr* was in a dominant position over *Elasticsearch* for the second run. There was also the same number of clear cut misses for both search engines. More importance where put on the second run as the second run should be more similar to a real operating environment with a warm cache. Therefore there was reason to toss the null hypothesis as *Solr* was significantly better than *Elasticsearch* for the second search round.

2.d) Which search engine used the least memory during search?
- Tossing the null hypothesis as there was a big difference between *Elasticsearch* and *Solr* usage of Virtual memory. *Elasticsearch* uses around 13 more GiB on Virtual memory then *Solr*.

7.2 Discussion of Findings

7.2.1 Indexing performance

While not being the focus of the paper, this part tries providing the argumentations the black box testing results.

Figure 8: One iteration of the indexer.

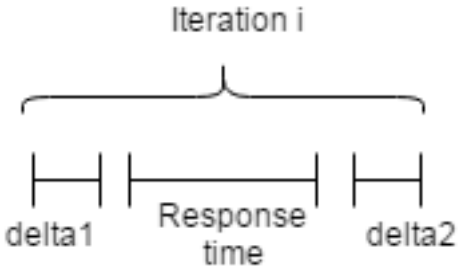


Figure 8 shows a single iteration of the Solr/Elasticsearch indexer. The indexer would run i=(number of bulk files for dataset d) times. Response time is the time the request handler used in millisecond from the time it received the the request and until completion of request handler tasks. The response time used as the measurement for indexing is the SUM response time over all iterations of the bulk index process (all bulk files for one dataset). The real time represents the total elapsed time for the completion of indexing all bulk files (bulk *Elastic* or bulk *Solr* for a dataset). Delta is defined as the difference between real time and response time. The indexer is still working but the actions is not part of the request handler. It is assumed that delta is some actions that is done both before and after the actions of the request handler [328, 329, 342]. The second assumption is that the actions we can be attributed to delta and response time is similar for Elasticsearch and Solr, with some minor exceptions. This is a list of actions that is reasonable to assume is part of the delta set of actions:

- Reading the bulk elastic/solr files to be indexed.
- For the first iteration of the indexer script the Curl command is establishing a connection with the Solr/Elastic server (i.e. localhost). It was verified that with the verbose parameter in curl this connection remains open when indexing multiple documents. This is also verified in the manpage for curl [343]. The manpage also says that using the same connection again only happens per curl command. Therefore a new connection/handshake is created at each iteration of the indexer script. It was assumed that the the difference in time it took to establish a connection with *Solr* over *Elasticsearch* using localhost was negligible.
- Transferring with curl
- Sending request to the request handler.
- Writing to console.

This is the set of actions that is believed to be part of the request handler:

- Creating index structure
- Creating ID automatically for the JSON document. IDs was not provided for the documents so the request handler had to either be given the ID by some other sub process or generate the ID on its own.
- commit (Solr specific command).
- Writing index to disk.

Why did Elasticsearch use more time with Delta over Solr regarding indexing?(see figure 4)

The two main reasons that where identified: 1) The *JSON* bulk index file format in *Elasticsearch* with the { "index":{} } lines, resulted in a "Elastic bulk index file" that was

about twice as large as in the case of *Solr* (see section 5.5 for *JSON* format examples). 2) It has been observed that *Elasticsearch* writes far more information to the console than *Solr* during indexing. I considers the latter the highest contributing factor.

Why did Elasticsearch out perform Solr on index creation time?

As far less time was used on the delta actions in Solr then Elasticsearch (see figure 4) more emphasis was put on the request handler actions for explaining this question.

1. Frequency of commit:
   - The commit operation was performed at every iteration of the indexer script, and may therefore be partially responsible for slowing down the overall indexing process. While the commit operations are needed in *Solr* in order to make the documents available for search, the operation is not free [342, 344]. It was recommended in [342,344] to limit the frequency of commits to improve indexer performance with respect to index creation time. The last dataset was processed in a similar time for both Solr and *Elasticsearch* with 2 bulk files = 2 iterations = 2 commits (see figure 2) . Either the influence commits have on creation time are not that significant, or the commits gets slower and slower as more documents are added to the *Solr* collection.
2. Document size and line length:
   - The response times in *Solr* (see figure 2) ranges from 2-5 times that of *Elasticsearch* (first run *Solr* divided by first run *Elastic*). More time have been spent on processing larger datasets, but there seem to be a larger gap between *Solr* and *Elastic* when there are many low character count lines such as the line range 11-100(see figure 2 and table 10). For example p32 dataset has the largest gap of 5 with many 11-100 lines, while *Enron* dataset with the second smallest gap of 2.2 and had only a few 11-100 lines. If *Elasticsearch* is slightly faster at indexing smaller documents/lines over *Solr*, and that there is many of these smaller documents, then that can explain the gap.
3. Index structure:
   - Given the big size difference between the indexes in *Solr* and *Elasticsearch* (see figure 5), the longer index creation time in *Solr* can partly be explained by having to write more to disk and more processing of the documents to create the index.
4. Segment count:
   - There is a consideration to be made in *Solr*, if the segment count is set to a low number, then more merges will occur when indexing, that negatively effects indexing performance [345]. On the upside, queries will be faster as there are fewer places to search. If on the other hand the segment count is high, then we get the opposite situation with improved indexed speed and degraded search performance. A low segment count can be the reason why *Solr* seems to perform badly with indexing, yet be good at searching.
5. *Zookeeper* and *Solr* stored on the same machine:
   - *Solr* cloud has a dependency on a server/program called *Apache ZooKeeper* and its database [346]. In our experiment where the *Zookeper* server and *Solr* was located on the same virtual machine, it might cause *Solr* and *Zookeper* to fight over I/O resources and therefore trigger a *Zookeper* timeout [346]. This

could explain the gap in indexing performance.

6. Sharding:
    - The default number of shards was used in both *Solr* (i.e. 1) and *Elasticsearch* (i.e. 5). Maybe the indexing process can be speed up with more shards, and therefore *Elasticsearch* performs better.

Why is the index size in Solr so much larger then Elasticsearch

Indexing structure, duplication of data and that the default *Solr* behavior is to emphasis query speed over degree of compression [347] are the 3 main likely reasons for the gap in index size. The default compression algorithm in *Elasticsearch* is LZ4 [348] and it seems to focus on search performance over compression also.

7.2.2 Query performance

Attempt to explain query search response time performance:

In section 7.2 it was previously stated that a high segment count (the experiment operated with default values) could negatively affect search performance and that both *Solr* and *Elasticsearch* are emphasizing search performance over compression, as the default behaviour. If multiple shards are presents on the same node/host/computer (not distributed) then search performance can be negatively affected according to [349]. Which is the case with *Elasticsearch* where 5 shards are on the same host machine. *Elasticsearch* out performs *Solr* 4/5 search categorizes (see section 6.2.1) but *Solr* performs better at the second run. I argue that the latter observation was because *Solr* is better (e.g. caching more items or caching the right set of items) at using the cached items from the previous search then *Elasticsearch* [350].

Attempting to explain search query hits

Solr and *Elasticsearch* perform quite similar. But gave slightly different number of search hits. The reasons for this was either the search queries do not behave exactly the same way, parsing index data was done differently in the search engines, or one search engine parse more information (more exhaustive search) than the other. It was also assumed that OR and AND operators did not affect single terms search, this assumption might be wrong and can be the reason for the big difference between the number of search hits in *Solr* and *Elasticsearch* with respect to single term search. From the relevancy list (see appendix B.2) we can see that *Solr* and *Elasticsearch* both matched mixed case words.

7.2.3 Memory performance during searching

There are differences in memory performance, but not that significant (see table 11). One question that had arisen is what was taken into account when the memory was measured. The code should have summed up all threads for the process under question. But what about forking short lived sub procces that are helping with a search task. And what part of *Solr* are captured. *Solr* can be divided into *Solr*, *Solr cloud* and *Zookeper*. So which one of these was parts of the memory summery?. *Solr* did perform better, but if not all of these process were taken into account, then the memory performance might not favor *Solr* after all.

7.3 Pros and Cons with "out of the box" Solr and Elasticsearch

While *Solr* excels at second run search and uses less memory, it is far worse on index creation time and index storage. *Elasticsearch* are good all around. The fact that the index

size increases rather than decreases like in *Elasticsearh* is problematic when dealing with large evidence collections. According to [346] there is also a known issue in *Solr* with collection/index counts that exceeds 100, and it just get progressively worse as more collections are added.

If the response time for the search query is 100 ms or 1 ms, then it does not matter that much in the case where there is a single human interacting with the search engine. This is because the human typing keywords cannot react fast enough to benefit from the fast performance. However if many repeated are performed, then the response time will add up/be significant. If on the other hand there is a script running queries in succession, or a software agent that is sending many request to the search engine then the performance gain is important. In addition, if the search engine is distributed with shards and are shared between many forensics practitioners using the service at the same time, then I would argue that the difference from 100 to 1 ms in response time is significant.

7.4 Theoretical implications

This thesis produced 3 literature reviews and a benchmark experiment on how well the open source search engines *Solr* and *Elasticsearch* perform at indexing and keyword searching. In the first review, scientific papers from 2014 to 2017 on the application of search in a forensic setting/task was summarized. This summary was a new contribution as no recent survey on the usage of search in Digital Forensics was identified. The second new contribution from review 1, was the summary of open search engines that were still being maintained and its main search capabilities. In review 2 the Digital forensics tools was identified and a in depth comparison of the search capabilities was made between a set of open search engines and forensics tool capable of keyword search. This in depth analysis is a new contribution as no up to date and equally extensive comparison was found. The last review the most extensive list [to my knowledge] of forensic related datasets was summarized. The dataset list is also a new contribution to the field of Digital forensics. The novelty of the experiments is benchmarking using:

- the same environment,
- same datasets,
- same preprocessing steps,
- same bulk size,
- importing from command line using curl command,
- using default configurations,
- and using the same measurements.

7.5 Practical implications

Who benefits:

1. Summary of recent applications of search in Digital forensics:
    - The summary have contributed to partly cover a missing gap in Digital forensic research. Combining this review with review 2, forensic practitioners and researchers can see how search have been used in a (experimental) forensic setting and the tools that implements these search features. This can be useful for forensic practitioners, so that they can figure out what search features they need in their tools.
2. A list of open source search engines and forensic tools that are capable of keyword

search and that are currently being maintained:
    - Finding the right open source search engines and forensics tools that fits your forensic task can be difficult. Therefore this addition to the thesis can be valuable for forensics practitioners trying to find the right set of tools or a forensic researcher that are researching the topic of "search and open source tools".
3. In depth comparison of search capabilities:
    - The comparison is useful for forensic practitioners to identify which tools have the desired search capabilities for their forensic problem. A quick summary of this information can be seen in figure 5.
4. A list of publicly available forensic related datasets:
    - This list can save many researchers time to find good datasets in network, network security, malware, spam, sms, email, fraud, files etc.
5. Benchmark experiments *Solr* and *Elasticsearch*:
    - The experiment environment can be similar to a small, but powerful forensic lab. The results revealed some major differences in indexing and search performance between the two most popular open search engines. The experiments can be useful for forensic practitioners when deciding which search engine to adopt, and for researchers researching Information retrieval or Digitalforensics.

7.6 Reliability and validity of experimental results

The measurements of indexing, searching and memory was consistent over multiple runs. The response time and search hits are taken from the search engines themselves, so validity (measuring the correct thing) should not be in doubt. The time command where used to get the total elapsed time of the index process. This measurement was validated manually as I also watched the system clock on my machine as the indexer started and ended. But some of the time command elapsed time could also be from counting threads run time and forked processes run time. No checks were made to rule this out. The validity of the memory measurement are in question as there are doubts on what are actually being measured (see section 7.2.3).

When it comes to reproducibility, the experiments are made on a newly created virtual machine with the OS type and version stated. The source for the tutorials used for installing the search engines have been provided and the search engine version. The experimental design also states which dependencies was installed. The experiment uses mostly publicly available datasets, and link for where to obtain these datasets have been provided. The scripts for preprocessing, indexing, searching and memory measurements are all in the thesis. All what the reader needs to know to replicate the experiments are provided.

7.7 Limitations

I could not obtain information on what low level algorithms that the applications are using (e.g. string matching algorithms like aho–corasick). This information could help explain why the software is performing well and how it could potentially be improved. By dismissing applications that had no recent updates and had little documentation, I might have discarded multiple good candidates like the forensics tool ELK [351].

The documentation used for the search capability comparison might be out of date,

and recent versions of the software might not have their old functionality working properly. Ideally, experiments should have been performed to verify the statements made in the software documentation, but there was not enough time to do that.

The experiment tested only 1 software configuration (i.e. default settings). If more configurations had been tested, then some of the variables such as sharding and segment count could be controlled for. And could see which search engine performed best at various optimizations. Also if the environments have multiple virtual machines, then we could have tested which search engine is best when shards, servers and databases are distributed. This could control for the Zookeper and sharding variable. I should also have tried to make sub datasets that have different number of lines (documents) and line length (document size). This would be done in order to control for document size and number of documents influence on the index performance and search performance. I would also have liked to had Terabyte sized dataset to check how the search engines scales with respect to indexing and searching.

There were also no practical experiments on forensics tools for keyword search comparison. Ideally, two forensic tools of every category would be selected for inspection, so that forensic practitioners could know what tool of which category would be most applicable for their needs. But this was not feasible due to time constraints.

7.8 Future work

To the best of my knowledge, there were no surveys from 2014-2017 in the application of search in Digital Forensics. An approach to better cover this topic could be to use older published papers and interviewing forensic practitioners on how they use search in their line of work.

The work done with table 5 could be extended by either adding to it, or by performing practical experiments to verify the contents of the software documentation. This would be useful for forensics practitioners and researchers, as the software might not work as stated in the documentation.

A lot of resources have been invested in finding details about what string matching search algorithms the open source search engines and open source forensic tools were using. This was a point of interest as this could partly explain why the applications were performing the way they do, and how they can be improved. This information was not easily accessible. To acquire this information I looked though:

- relevant scientific papers,
- white papers,
- web pages,
- github repositories to identify relevant comments and source code regarding search algorithms
- and manual inspection of the source code

. This searching and manual review of source code did not give many results, and the process of finding this information had to be terminated due to time constraints. A future paper could look into using benchmark scripts written specifically for the applications and application specific benchmark API that can potentially output details about underlying algorithms.

There are undoubtedly more datasets available online that can be of interest of forensics researchers. One methodology that I think could be promising is to 1) map universities

and colleges that hosts forensic related labs or courses and their websites. Then 2) go through the school website to find datasets. The reason why I think this methodology can give results is that some of the datasets found in this thesis was hosted on universities websites.

New experiments could test how well the search engines perform with virtual machine with low to high allocation of storage and RAM. Future experiments could have a multi virtual machine environment and spread the databases, shards and servers amongst the hosts. This environment would be suitable for testing how well the search engines perform at distributed search. The experiments could also test different import methods, importing using multiple threads and using different bulk sizes to test which approach minimizes index creation time. Another possibility is to look into which search engine perform best with structured data (e.g. multiple fields to search) and which is best at searching though multiple indexes and document types. Different optimizations (configurations) should also be considered to figure out the best performing search engine state. Could be interesting to explore how forensics tools like Sluthkit and ELK have functionality that is dependent on *Solr* and *Elasticsearch* [352, 353]. Multiple sets of different documents sizes and number of documents and their influence of index size and index creation time should be tested to better understand how the search engines scale. It could be interesting to compare relevancy lists for the same search and look into precision and recall.

Appendices

A Experimental design

A.1 Snort IDS logs ID 59 - General preprocessing

Listing A.1: All general preprocessing in 1 script for snort dataset

```
cat /home/search/Downloads/Datasets/snort |
dos2unix |
xargs |
sed 's/ \+/ /g' |
sed 's/\[\*\*\] \[1/\n\[\*\*\] \[1/g' |
sed -e 's/\],\\/ /g' -e 's/\},/ /g'
-e 's/\}\\/ /g' -e 's/{\\/ /g'
-e 's/\":/ /g' -e 's/\\\"/ /g'
-e 's/\]\\/ /g' -e 's/\[\\/ /g'
-e 's/,\\/ /g' -e 's/ \\/ /g'
-e 's/\]\"/ /g' -e 's/\]/ /g'
-e 's/\[/ /g' -e 's/"/ /g'
-e 's/://g' -e 's/{/ /g'
-e 's/}/ /g' |
sed '/^\s*$/d' |
sed 's/\*/ /g' |
sed 's/ \+/ /g'|
perl -pe 's/[^[:ascii:]]//g;' |
tr -cd '\12\40-\176' |
sed -e 's/\\/\\\\/g' |
split -l 1000 -d
- /home/search/Downloads/Datasets/batchSnort/snortFile
```

A.2 Hillary emails - General preprocessing

Listing A.2: General preprocessing hillary clinton emails

```
cat Emails.csv |
sed ':a;N;$!ba;s/\n/ /g'  |
perl -pe  "s/$(echo -en '\u000C')\"/\n/g" |
dos2unix| sed -e 's/\],\\/ /g'
-e 's/\},/ /g' -e 's/\}\\/ /g'
-e 's/{\\/ /g' -e 's/\":/ /g'
-e 's/\\\"/ /g' -e 's/\]\\/ /g'
-e 's/\[\\/ /g' -e 's/,\\/ /g'
-e 's/ \\/ /g' -e 's/\]\"/ /g'
-e 's/\]/ /g' -e 's/\[/ /g'
-e 's/"/ /g' -e 's/://g'
-e 's/{/ /g' -e 's/}/ /g' |
sed '/^\s*$/d' |
sed 's/\*/ /g' |
sed 's/ \+/ /g' |
perl -pe 's/[^[:ascii:]]//g;' |
tr -cd '\12\40-\176' |
sed -e 's/\\/\\\\/g' | split -l 1000 -d
- /home/search/Downloads/
Datasets/batchHillaryEmails/hillary
```

A.3 Enron emails - General preprocessing

Listing A.3: General preprocessing Enron emails

```
cat /home/search/Downloads
/Datasets/enron.csv |
sed ':a;N;$!ba;s/\n/ /g'  |
perl -pe "s/Date: /\nDate: /g" |
dos2unix| sed -e 's/\],\\/ /g'
-e 's/\},/ /g' -e 's/\}\\/ /g'
-e 's/{\\/ /g' -e 's/\":/ /g'
-e 's/\\\"/ /g' -e 's/\]\\/ /g'
-e 's/\[\\/ /g' -e 's/,\\/ /g'
-e 's/ \\/ /g' -e 's/\]\"/ /g'
-e 's/\]/ /g' -e 's/\[/ /g'
-e 's/"/ /g' -e 's/://g'
-e 's/{/ /g' -e 's/}/ /g' |
sed '/^\s*$/d' |
sed 's/\*/ /g' |
sed 's/ \+/ /g' |
perl -pe 's/[^[:ascii:]]//g;' |
tr -cd '\12\40-\176' |
sed -e 's/\\/\\\\/g' |
split -l 1000 -d -
/home/search/Downloads/
Datasets/batchEnron/enron
```

A.4 NUSSC sms - General preprocessing

Listing A.4: General preprocessing Enron emails

```
cat /home/search/Downloads/
Datasets/sms.xml |
sed 's/<\/message>/<\/message>aaaaaaa/g' |
sed -E 's/<[^>]+>/ /g'|
sed ':a;N;$!ba;s/\n/ /g'  |
perl -pe 's/aaaaaaa/\n/g' |
perl -pe 's/&amp//g' |
dos2unix| sed -e 's/\],\\/ /g'
-e 's/\},/ /g' -e 's/\}\\/ /g'
-e 's/{\\/ /g' -e 's/\":/ /g'
-e 's/\\\"/ /g' -e 's/\]\\/ /g'
-e 's/\[\\/ /g' -e 's/,\\/ /g'
-e 's/ \\/ /g' -e 's/\]\"/ /g'
-e 's/\]/ /g' -e 's/\[/ /g'
-e 's/"/ /g' -e 's/://g'
-e 's/{/ /g' -e 's/}/ /g'  |
sed 's/unknown/ /g' |
sed '/^\s*$/d' |
sed 's/\*/ /g' |
sed 's/ \+/ /g' |
perl -pe 's/[^[:ascii:]]//g;' |
tr -cd '\12\40-\176' |
sed -e 's/\\/\\\\/g' |
split -l 1000 -d -
/home/search/Downloads/Datasets/batchSms/sms
```

A.5 DITSSC Spam - General preprocessing

Listing A.5: General preprocessing Enron emails

```
cat /home/search/Downloads/
Datasets/smsSpam.xml |
sed 's/<\/sms>/<\/sms>aaaaaaa/g' |
sed -E 's/<[^>]+>//g'|
sed ':a;N;$!ba;s/\n/ /g'  |
perl -pe 's/aaaaaaa/\n/g' |
perl -pe 's/&amp//g' |
dos2unix| sed -e 's/\],\\/ /g'
-e 's/\},/ /g' -e 's/\}\\/ /g'
-e 's/{\\/ /g' -e 's/\":/ /g'
-e 's/\\\"/ /g' -e 's/\]\\/ /g'
-e 's/\[\\/ /g' -e 's/,\\/ /g'
-e 's/ \\/ /g' -e 's/\]\"/ /g'
-e 's/\]/ /g' -e 's/\[/ /g'
-e 's/"/ /g' -e 's/://g'
-e 's/{/ /g' -e 's/}/ /g' |
sed '/^\s*$/d' |
sed 's/\*/ /g' |
sed 's/ \+/ /g' |
perl -pe 's/[^[:ascii:]]//g;' |
tr -cd '\12\40-\176' |
sed -e 's/\\/\\\\/g' |
split -l 1000 -d -
/home/search/Downloads/Datasets/batchSpam/spam
```

A.5.1

B Result and analysis

B.1 Dataset line length

B.1.1 Code

Code was modified from [354] to fit the purpose of getting the frequency of line length intervals, in order to better describe the dataset getting indexed. The listings B.1 first starts by getting the full path to a file in the command command line argument 1 and then process the file. Awk first initialize the variables, then stores the length of the current line being read in variable thislen, increment the intervall that thislen falls under and then finally print the results.

Listing B.1: Enumerate line lengths

```
cat $1 | awk ' BEGIN{upTo10=0;
upTo100=0; upTo500=0;
upTo1000=0; upTo5000=0;
upTo10000=0; upTo50000=0;
upTo100000=0; upTo500000=0;
upToMore=0;totlen=0;}

{thislen=length($0); totlen+=thislen;
if (thislen <= 10) upTo10++;
else if (thislen >10 && thislen <= 100) upTo100++;
else if (thislen >100 && thislen <=500) upTo500++;
else if (thislen >500 && thislen <=1000) upTo1000++;
else if (thislen >1000 && thislen <=5000) upTo5000++;
else if (thislen >5000 && thislen <=10000) upTo10000++;
else if (thislen >10000 && thislen
 <=50000) upTo50000++;
else if (thislen >50000 &&
thislen <=100000) upTo100000++;
else if (thislen >100000
 && thislen <=500000) upTo500000++;
else upToMore++;
}

END { printf("average: %d\n", totlen/NR);
printf("Number of lines: %d\n",NR);
printf("Line length 0-9: %d\n",upTo10);
printf("Line length 11-100: %d\n",upTo100);
printf("Line length 101-500: %d\n",upTo500);
printf("Line length 501-1000: %d\n",upTo1000);
printf("Line length 1001-5000: %d\n",upTo5000);
printf("Line length 5001-10000: %d\n",upTo10000);
printf("Line length 10001-50000: %d\n",upTo50000);
printf("Line length 50001-100000: %d\n",upTo100000);
printf("Line length 100001-500000: %d\n",upTo500000);
printf("Longer line lengths: %d\n",upToMore);
} '
```

B.2 Output Solr and Elasticsearch

B.2.1 Solr Query:shoppin


```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":29,
    "params":{
      "q":"content:\"shoppin\""}},
  "response":{"numFound":12,"start":0,"docs":[
      {
        "content":[" m stuck in mango... Shoppin! 8 8 SG  "],
        "id":"cc97bc77-a870-43d6-83a7-831467e6fb9b",
        "_version_":1584915314292817961,
        "content_str":[" m stuck in mango... Shoppin! 8 8 SG  "]},
      {
        "content":[" m stuck in mango... Shoppin! 8 8 SG  "],
        "id":"2ca2c2ce-0230-4ce6-97b1-98cab25a8761",
        "_version_":1584915314372509706,
        "content_str":[" m stuck in mango... Shoppin! 8 8 SG  "]},
      {
        "content":[" m fine... Shoppin now at heeren... 0 0 SG  "],
        "id":"626baa1d-d717-47eb-aa63-ca5bd6d51a6a",
        "_version_":1584915314692325382,
        "content_str":[" m fine... Shoppin now at heeren... 0 0 SG  "]},
      {
        "content":[" s so funny... Enjoy shoppin gal... 8 8 SG  "],
        "id":"78212f7d-7c4c-4456-b62f-3ed18eab21ca",
        "_version_":1584915314292817957,
        "content_str":[" s so funny... Enjoy shoppin gal... 8 8 SG  "]},
      {
        "content":[" s so funny... Enjoy shoppin gal... 8 8 SG  "],
        "id":"6cb8f331-d83d-48fa-a6dd-dcfb88137892",
        "_version_":1584915314372509702,
        "content_str":[" s so funny... Enjoy shoppin gal... 8 8 SG  "]},
      {
        "content":[" Help me record betelnut beauty at 2, channel u... Ur
goin shoppin later? 8 8 SG  "],
        "id":"3dac02b6-81c6-4b28-aa72-caf070149ca7",
        "_version_":1584915314332663855,
        "content_str":[" Help me record betelnut beauty at 2, channel
u... Ur goin shoppin later? 8 8 SG  "]},
      {
        "content":[" Hmmm.... Mayb can try e shoppin area one, but forgot
e name of hotel... 51 51 SG  "],
        "id":"a4b83720-c848-4b1a-aa9b-6a8f2c82fc82",
        "_version_":1584915314203688972,
        "content_str":[" Hmmm.... Mayb can try e shoppin area one, but
forgot e name of hotel... 51 51 SG  "]},
      {
        "content":[" My fri ah... Okie lor,goin 4 my drivin den go
shoppin after tt... 51 51 SG  "],
        "id":"fa54bfa2-2d70-4f0a-abc0-19d7efe120c7",
        "_version_":1584915314203688979,
        "content_str":[" My fri ah... Okie lor,goin 4 my drivin den go
shoppin after tt... 51 51 SG  "]},
      {
        "content":[" Meet at tanjong pagar station at one? If dun have
den juz go shoppin, haha... 8 8 SG  "],
        "id":"8c40a271-d427-485d-8428-724cd7fc6137",
        "_version_":1584915314375655445,
```

B.2.2 Solr Query:shoppin


```
        "content_str":[" Meet at tanjong pagar station at one? If dun
have den juz go shoppin, haha... 8 8 SG  "]},
      {
        "content":[" Where r u ? U went shoppin wout me... Anythin la, u
like den buy, but u will wear meh... 0 0 SG  "],
        "id":"499931b2-bbe3-458b-b739-bc531f749cf4",
        "_version_":1584915314689179653,
        "content_str":[" Where r u ? U went shoppin wout me... Anythin
la, u like den buy, but u will wear meh... 0 0 SG  "]}]
  }}
```

B.2.3 Elasticsearch Query:shoppin


```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 12,
    "max_score" : 11.255844,
    "hits" : [
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mLi-7KAqxoxVJdy",
        "_score" : 11.255844,
        "_source" : {
          "content" : " m stuck in mango... Shoppin! 8 8 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mLi-7KAqxoxVJdu",
        "_score" : 10.456016,
        "_source" : {
          "content" : " s so funny... Enjoy shoppin gal... 8 8 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mRS-7KAqxoxVKpt",
        "_score" : 10.456016,
        "_source" : {
          "content" : " m fine... Shoppin now at heeren... 0 0 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mMu-7KAqxoxVJms",
        "_score" : 10.198918,
        "_source" : {
          "content" : " s so funny... Enjoy shoppin gal... 8 8 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mMu-7KAqxoxVJmw",
        "_score" : 10.198918,
        "_source" : {
          "content" : " m stuck in mango... Shoppin! 8 8 SG  "
        }
      },
      {
        "_index" : "batch-sms",
```

B.2.4 Elasticsearch Query:shoppin


```
        "_type" : "somedoctype",
        "_id" : "AV_r9mMu-7KAqxoxVJpe",
        "_score" : 9.806873,
        "_source" : {
          "content" : " Meet at tanjong pagar station at one? If dun have
den juz go shoppin, haha... 8 8 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mLi-7KAqxoxVJiV",
        "_score" : 9.354715,
        "_source" : {
          "content" : " Help me record betelnut beauty at 2, channel u...
Ur goin shoppin later? 8 8 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mKL-7KAqxoxVJIk",
        "_score" : 9.242589,
        "_source" : {
          "content" : " Hmmm.... Mayb can try e shoppin area one, but
forgot e name of hotel... 51 51 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mKL-7KAqxoxVJIr",
        "_score" : 8.376655,
        "_source" : {
          "content" : " My fri ah... Okie lor,goin 4 my drivin den go
shoppin after tt... 51 51 SG  "
        }
      },
      {
        "_index" : "batch-sms",
        "_type" : "somedoctype",
        "_id" : "AV_r9mRS-7KAqxoxVKnX",
        "_score" : 7.620589,
        "_source" : {
          "content" : " Where r u ? U went shoppin wout me... Anythin la,
u like den buy, but u will wear meh... 0 0 SG  "
        }
      }
    ]
  }
}
```

B.3 Indexing performance

Table 12 contains information regarding how long time it took to index the datasets and how big the resulting index was. Two runs of indexing each dataset was undertaken, in order to prevent outliers. Caching might improve the performance of the second run.

Table 12: Indexing performance
Source: Table appearance/latex code for table creation is from [355]

Dataset (index/collection name)	Elasticsearch		Solr	
	Index creation time	Index size (num of docs)	Index creation time	Index size
windows-p32 (batch-andrii-malware2 / mal3)	• First run: – took: 842121 ms – real: 50 m 14.955s – user: 2 m 21.208s – sys: 0 m 55.004s • Second run: – took: 874577ms – real: 50m42.129s – user: 2m21.480s – sys: 0m53.236s	• First run: – 47.5mb (45117458) • Second run: – 47.3mb (45117458)	• First run: – QTime: 4238554ms – Real: 79 M 57.533s – User: 2 M 22.256s – sys: 0 M 52.368s • second run: – QTime: 4139910ms – real: 78m45.055s – user: 2m21.124	• First run – 23.2GB • Second run – 14.9GB
Snort IDS logs ID 59 (snort-logs)	• First run: – took: 898 ms – real: 0m1.735s – user: 0m0.084s – sys: 0m0.032s • Second run: – took: 711ms – real: 0m1.608s – user: 0m0.084s – sys: 0m0.032s	• First run: – 98.4kb (25820) • Second run: – 98.2 kb	• First run: – QTime: 2128 – real: 0m2.451s – user: 0m0.100s – sys: 0m0.024s • second run: – QTime: 1558ms – real: 0m2.107s – user: 0m0.080s – sys: 0m0.028s	• First run: – 12.2MB • Second run: – 12.2MB
Hillary clinton emails (batch-hil-email)	• First run: – took: 1058ms – real: 0m1.337s – user: 0m0.032s – sys: 0m0.032s • Second run: – Took: 1082ms – real: 0m1.658s – user: 0m0.020s – sys: 0m0.048s	• First run: – 102.5kb (7944) • Second run: – 117.8 kb (7944)	• First run – QTime: 2590ms – real: 0m2.703s – user: 0m0.036s – sys: 0m0.028s • Second run: – QTime: 2510ms – real: 0m3.045s – user: 0m0.040s – sys: 0m0.024s	• First run: – 27.3 MB • Second run: – 27.3 MB
Enron emails (batch-enron)	• First run: – took: 63832ms – real: 1m23.400s – user: 0m2.172s – sys : 0m1.804s • Second run: – took: 62021ms – real: 1m21.173s – user: 0m2.136s – 0m2.284s	• First run: – 3.2mb (554800) • Second run: – 3.2mb (554800)	• First run: – QTime: 143021ms – real: 2m31.059s – user: 0m2.024s – sys: 0m2.072s • Second run: – QTime: 144739ms – real: 2m32.412s – user: 0m2.068s – sys: 0m2.072s	• First run: – 1.5 GB • Second run: – 1.5 GB
NUSSC SMS (batch-sms)	• First run: – Took: 1128ms – real: 0m3.537s – user: 0m0.188s – sys: 0m0.056s • Second run: – Took: 1090ms – real: 0m3.015s – user: 0m0.168s – sys: 0m0.068s	• First run: – 110.7 kb (55849) • Second run: – 106.2 kb (55849)	• First run: – QTime: 2809ms – real: 0m3.492s – user: 0m0.180s – sys: 0m0.064s • Second run: – QTime: 3056ms – real: 0m3.724s – user: 0m0.184s – sys: 0m0.052s	• First run: – 23.7 MB • Second run: – 23.7 MB
DITSSC SPAM (batch-spam)	• First run: – took: 158ms – real: 0m0.230s – user: 0m0.004s – sys: 0m0.004s • Second run: – Took: 118ms – real: 0m0.238s – user: 0m0.004s – sys: 0m0.004s	• First run: – N/A - too small (1353) • Second run: – N/A - too small (1353)	• Fist run: – QTime: 153ms (2 bulk files = 2 commits performed) – real: 0m0.181s – user: 0m0.004s – sys: 0m0.004s • Second run: – QTime: 124ms – real: 0m0.161s – user: 0m0.004s – sys: 0m0.004s	• First run: – 709.7 KB • Second run: – 709.6 KB

B.4 Query performance

The tables 13 and 14 shows he Query string, the search function used and search time in MS taken from *took* (*Elastic*) and *Qtime* (*Solr*). Elastic(OR) (see section 5.6) and Solr (AND+) was executed on all search queries. While Solr(OR) and Elastic (AND+) was executed on multi terms searches only. It was assumed that single term searches with the

OR operator or AND operator will not effect the end results. This assumption might not be true.

Table 13: Time to complete query request 1 of 2

Dataset name	Query string	Elastic (OR)	Solr (AND+)	Solr (OR)	Elastic (AND+)
Windows p32	Trojan-Clicker.Win32.Agent.acu	total: 485706 1. 752ms 2. 21ms	total: 1 1. 961ms 2. 0ms		
Windows p32	Trojan-Downloader.Win32.QQHelper.wo	total: 485787 1. 850ms 2. 44ms	total: 2 1. 663ms 2. 0ms		
Windows p32	Trojan-PSW.win32.Magania.nof.exe	total: 485705 1. 450ms 2. 24ms	total:1 1. 768ms 2. 0ms		
Windows p32	We are young Money	total: 76238 1. 738ms 2. 11ms	total:2 1. 1424ms 2. 0ms	total:76168 1. 114ms 2. 0ms	total: 2 1. 67ms 2. 3ms
Windows p32	The ordinal %u could not be located in the dynamic link library	total: 3604900 1. 2713 ms 2. 136 ms	N/A due to not able to handle %u	N/A	N/A
Windows p32	Error writhing temporary file. Make sure your temp folder is valid	total: 2226913 1. 1953ms 2. 84ms	total: 0 1. 1464ms 2. 0ms	total: 2222171 1. 1605ms 2. 0ms	total: 0 1. 5ms 2. 5ms
Windows p32	02edbd94746bc69677e969a89c4eb0d8	total: 0 1. 367ms 2. 1ms	total:0 1. 377ms 2. 0ms		
Snort IDS logs ID 59	02edbd94746bc69677e969a89c4eb0d8	total: 0 1. 54ms 2. 1ms	total: 0 1. 156ms 2. 1ms		
Snort IDS logs ID 59	Web App Scan in Progress	total: 14536 1. 145ms 2. 4ms	total: 6022 1. 237 ms 2. 0ms	total: 14521 1. 5ms 2. 0ms	total: 6029 1. 43ms 2. 4ms
Snort IDS logs ID 59	/etc/passwd	total: 459 1. 2ms 2. 2ms	total: 457 1. 75ms 2. 0ms		
Snort IDS logs ID 59	UNICODE CODEPOINT ENCODING	total: 3173 1. 54ms 2. 1ms	total:2332 1. 49ms 2. 0ms	total: 3170 1. 3ms 2. 0ms	total: 2334 1. 2ms 2. 1ms
Snort IDS logs ID 59	Priority 3	total: 25768 1. 44ms 2. 2ms	total: 7154 1. 73ms 2. 0ms	total: 25742 1. 10ms 2. 0ms	total: 7159 1. 60ms 2. 8ms
Snort IDS logs ID 59	http//www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/SCAN/SCAN_Nikto	total: 15759 1. 5ms 2. 9ms	total: 6022 1. 32ms 2. 0ms		
Snort IDS logs ID 59	Decoding	total: 208 1. 18ms 2. 1ms	total: 208 1. 1ms 2. 0ms		
Hillary Clinton emails	air base issue	total: 973 1. 151ms 2. 2ms	total:4 1. 275ms 2. 1ms	total: 972 1. 2ms 2. 0ms	total: 4 1. 63ms 2. 1ms
Hillary Clinton emails	anonymity	total: 42 1. 46ms 2. 1ms	total:42 1. 81ms 2. 0ms		
Hillary Clinton emails	AFGAN AID	total: 493 1. 51ms 2. 2ms	total: 0 1. 0ms 2. 0ms	total: 352 1. 1ms 2. 0ms	total: 0 1. 0ms 2. 0ms
Hillary Clinton emails	leaker	total: 1 1. 22ms 2. 1ms	total: 1 1. 0ms 2. 0ms		
Hillary Clinton emails	Independence	total: 48 1. 40ms 2. 1ms	total: 48 1. 0ms 2. 0ms		
Hillary Clinton emails	Syrian Ambassador	total: 537 1. 87ms 2. 2ms	total:5 1. 213ms 2. 0ms	total: 537 1. 9ms 2. 0ms	total: 5 1. 51ms 2. 1ms
Hillary Clinton emails	02edbd94746bc69677e969a89c4eb0d8	total: 0 1. 1ms 2. 1ms	total: 0 1. 0ms 2. 0ms		

Table 14: Time to complete query request 2 of 2

Dataset name	Query string	Elastic (OR)	Solr (AND+)	Solr (OR)	Elastic (AND+)
Enron emails	farewell party	total: 23452 1. 715ms 2. 21ms	total: 57 1. 1016ms 2. 0ms	total: 23421 1. 3ms 2. 0ms	total:57 1. 227ms 2. 2ms
Enron emails	Investigate	total: 2017 1. 368ms 2. 5ms	total: 2015 1. 551ms 2. 0ms		
Enron emails	settlement	total:7783 1. 281ms 2. 4ms	total: 7772 1. 374ms 2. 0ms		
Enron emails	imbalance penalties	total: 4778 1. 127ms 2. 3ms	total: 62 1. 479ms 2. 0ms	total: 4774 1. 2ms 2. 0ms	total: 62 1. 278ms 2. 2ms
Enron emails	Thanks for the tip last night	total: 467500 1. 578ms 2. 46ms	total: 4 1. 1460ms 2. 0ms	total: 467020 1. 131 ms 2. 0ms	total: 4 1. 462ms 2. 4ms
Enron emails	courts	total: 1822 1. 278ms 2. 2ms	total: 1821 1. 385ms 2. 0ms		
Enron emails	02edbd94746bc69677e969a89c4eb0d8	total: 0 1. 158ms 2. 1ms	total: 0 1. 353ms 2. 1ms		
NUSSC SMS	when you get back	total: 5 1. 156ms 2. 2ms	total: 1 1. 255ms 2. 1ms	total: 10359 1. 125ms 2. 0ms	total: 1 1. 55ms 2. 1ms
NUSSC SMS	I booked all already	total: 15506 1. 40ms 2. 5ms	total: 1 1. 1ms 2. 0ms	total: 15493 1. 4ms 2. 0ms	total: 1 1. 1ms 2. 1ms
NUSSC SMS	can go 4 dinner together	total: 8136 1. 28ms 2. 2ms	total: 1 1. 57ms 2. 0ms	total: 8126 1. 3ms 2. 0ms	total:1 1. 1ms 2. 1ms
NUSSC SMS	shoppin	total: 12 1. 1ms 2. 1ms	total: 12 1. 29ms 2. 0ms		
NUSSC SMS	browser	total: 1 1. 1ms 2. 1ms	total: 1 1. 0ms 2. 0ms		
NUSSC SMS	computers	total: 3 1. 18ms 2. 1ms	total: 3 1. 15ms 2. 0ms		
NUSSC SMS	02edbd94746bc69677e969a89c4eb0d8	total: 0 1. 0ms 2. 1ms	total: 0 1. 21ms 2. 0ms		
DITSSC SPAM	FREE membership	total: 315 1. 45ms 2. 2ms	total:1 1. 21ms 2. 0ms	total: 315 1. 0ms 2. 0ms	total: 1 1. 1ms 2. 1ms
DITSSC SPAM	click the WAP link	total: 363 1. 14ms 2. 1ms	total:1 1. 18ms 2. 0ms	total:363 1. 1ms 2. 0ms	total: 1 1. 1ms 2. 0ms
DITSSC SPAM	your subscription	total: 409 1. 2ms 2. 1ms	total: 1 1. 0ms 2. 0ms	total:409 1. 11ms 2. 0ms	total:1 1. 1ms 2. 1ms
DITSSC SPAM	charged	total: 19 1. 8ms 2. 1ms	total:19 1. 21ms 2. 0ms		
DITSSC SPAM	call	total: 566 1. 1ms 2. 1ms	total:565 1. 0ms 2. 0ms		
DITSSC SPAM	FREE	total: 315 1. 1ms 2. 1ms	total:315 1. 0ms 2. 0ms		
DITSSC SPAM	02edbd94746bc69677e969a89c4eb0d8	total: 0 1. 1ms 2. 0ms	total: 0 1. 49ms 2. 1ms		

Bibliography

[1] Zawoad S, Hasan R. Digital Forensics in the Age of Big Data: Challenges, Approaches, and Opportunities. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems; 2015. p. 1320–1325.

[2] Yafooz WMS, Abidin SZZ, Omar N, Idrus Z. Managing unstructured data in relational databases. In: 2013 IEEE Conference on Systems, Process Control (ICSPC); 2013. p. 198–203.

[3] Li Y, Liu Z, Zhu H. Enterprise Search in the Big Data Era: Recent Developments and Open Challenges. Proc VLDB Endow. 2014 Aug;7(13):1717–1718. Available from: http://dx.doi.org/10.14778/2733004.2733071.

[4] Palmer G, Corporation M. A Road Map for Digital Forensic Research; 2001. Accessed 29.04.17. Available from: http://dfrws.org/sites/default/files/session-files/a_road_map_for_digital_forensic_research.pdf.

[5] Armknecht F, Dewald A. Privacy-preserving email forensics. Digital Investigation. 2015;14, Supplement 1:S127 – S136. The Proceedings of the Fifteenth Annual {DFRWS} Conference. Available from: http://www.sciencedirect.com/science/article/pii/S1742287615000481.

[6] Martini B, Choo KKR. Distributed filesystem forensics: XtreemFS as a case study. Digital Investigation. 2014;11(4):295 – 313. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614000942.

[7] Yu S. Covert communication by means of email spam: A challenge for digital investigation. Digital Investigation. 2015;13:72 – 79. Available from: http://www.sciencedirect.com/science/article/pii/S1742287615000432.

[8] Garfinkel SL, McCarrin M. Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. Digital Investigation. 2015;14, Supplement 1:S95 – S105. The Proceedings of the Fifteenth Annual {DFRWS} Conference. Available from: http://www.sciencedirect.com/science/article/pii/S1742287615000468.

[9] Thongjul S, Tritilanunt S. Analyzing and searching process of internet username and password stored in Random Access Memory (RAM). In: 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE); 2015. p. 257–262.

[10] Bjelland PC, Franke K, Årnes A. Practical use of Approximate Hash Based Matching in digital investigations. Digital Investigation. 2014;11, Supplement 1:S18 – S26. Proceedings of the First Annual {DFRWS} Europe. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614000085.

[11] Farhadi MR, Fung BCM, Fung YB, Charland P, Preda S, Debbabi M. Scalable code clone search for malware analysis. Digital Investigation. 2015;15:46 – 60. Special Issue: Big Data and Intelligent Data Analysis. Available from: http://www.sciencedirect.com/science/article/pii/S1742287615000705.

[12] Wang WB, Huang ML, Lu L, Zhang J. Improving Performance of Forensics Investigation with Parallel Coordinates Visual Analytics. In: 2014 IEEE 17th International Conference on Computational Science and Engineering; 2014. p. 1838–1843.

[13] Sharif SA, Ali MA, Reqabi NA, Iqbal F, Baker T, Marrington A. Magec: An Image Searching Tool for Detecting Forged Images in Forensic Investigation. In: 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS); 2016. p. 1–6.

[14] Pollitt M. Chapter 2 - The key to forensic success: examination planning is a key determinant of efficient and effective digital forensics. In: Sammons J, editor. Digital Forensics. Boston: Syngress; 2016. p. 27 – 43. Available from: http://www.sciencedirect.com/science/article/pii/B9780128045268000022.

[15] Rogers MK. Chapter 3 - Psychological profiling as an investigative tool for digital forensics. In: Sammons J, editor. Digital Forensics. Boston: Syngress; 2016. p. 45 – 58. Available from: http://www.sciencedirect.com/science/article/pii/B9780128045268000034.

[16] Mascarnes S, Lopes P, Sakhare P. Search model for searching the evidence in digital forensic analysis. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT); 2015. p. 1353–1358.

[17] Pollitt MM. Triage: A practical solution or admission of failure. Digital Investigation. 2013;10(2):87 – 88. Triage in Digital Forensics. Available from: http://www.sciencedirect.com/science/article/pii/S1742287613000030.

[18] Overill RE, Silomon JAM, Roscoe KA. Triage template pipelines in digital forensic investigations. Digital Investigation. 2013;10(2):168 – 174. Triage in Digital Forensics. Available from: http://www.sciencedirect.com/science/article/pii/S1742287613000261.

[19] Moser A, Cohen MI. Hunting in the enterprise: Forensic triage and incident response. Digital Investigation. 2013;10(2):89 – 98. Triage in Digital Forensics. Available from: http://www.sciencedirect.com/science/article/pii/S1742287613000285.

[20] Case A, III GGR. Memory forensics: The path forward. Digital Investigation. 2017;20:23 – 33. Special Issue on Volatile Memory Analysis. Available from: http://www.sciencedirect.com/science/article/pii/S1742287616301529.

[21] III GGR, Case A. In lieu of swap: Analyzing compressed {RAM} in Mac {OS} X and Linux. Digital Investigation. 2014;11, Supplement 2:S3 – S12. Fourteenth Annual {DFRWS} Conference. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614000541.

[22] Sylve JT, Marziale V, III GGR. Pool tag quick scanning for windows memory analysis. Digital Investigation. 2016;16, Supplement:S25 – S32. {DFRWS} 2016 EuropeProceedings of the Third Annual {DFRWS} Europe. Available from: http://www.sciencedirect.com/science/article/pii/S1742287616000062.

[23] Lapso JA, Peterson GL, Okolica JS. Whitelisting system state in windows forensic memory visualizations. Digital Investigation. 2017;20:2 – 15. Special Issue on Volatile Memory Analysis. Available from: http://www.sciencedirect.com/science/article/pii/S1742287616301438.

[24] Mohamed AFAL, Marrington A, Iqbal F, Baggili I. Testing the forensic soundness of forensic examination environments on bootable media. Digital Investig-

ation. 2014;11, Supplement 2:S22 – S29. Fourteenth Annual {DFRWS} Conference. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614000589.

[25] Attoe R. Chapter 6 - Digital forensics in an eDiscovery world. In: Sammons J, editor. Digital Forensics. Boston: Syngress; 2016. p. 85 – 98. Available from: http://www.sciencedirect.com/science/article/pii/B978012804526800006X.

[26] Beebe NL, Liu L. Clustering digital forensic string search output. Digital Investigation. 2014;11(4):314 – 322. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614001108.

[27] Lees C. Determining removal of forensic artefacts using the {USN} change journal. Digital Investigation. 2013;10(4):300 – 310. Available from: http://www.sciencedirect.com/science/article/pii/S1742287613001084.

[28] Leimich P, Harrison J, Buchanan WJ. A {RAM} triage methodology for Hadoop {HDFS} forensics. Digital Investigation. 2016;18:96 – 109. Available from: http://www.sciencedirect.com/science/article/pii/S1742287616300780.

[29] Wagner J, Rasin A, Grier J. Database image content explorer: Carving data that does not officially exist. Digital Investigation. 2016;18, Supplement:S97 – S107. Available from: http://www.sciencedirect.com/science/article/pii/S1742287616300500.

[30] Anwar T, Abulaish M. A social graph based text mining framework for chat log investigation. Digital Investigation. 2014;11(4):349 – 362. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614001091.

[31] Minnaard W. Out of sight, but not out of mind: Traces of nearby devices' wireless transmissions in volatile memory. Digital Investigation. 2014;11, Supplement 1:S104 – S111. Proceedings of the First Annual {DFRWS} Europe. Available from: http://www.sciencedirect.com/science/article/pii/S1742287614000188.

[32] Mathew LM, R S, Kizhakkethottam JJ. A survey on different video restoration techniques. In: 2015 International Conference on Soft-Computing and Networks Security (ICSNS); 2015. p. 1–3.

[33] Stewart J, Uckelman J. Unicode search of dirty data, or: How I learned to stop worrying and love Unicode Technical Standard number 18. Digital Investigation. 2013;10, Supplement:S116 – S125. The Proceedings of the Thirteenth Annual {DFRWS} Conference13th Annual Digital Forensics Research Conference. Available from: http://www.sciencedirect.com/science/article/pii/S1742287613000595.

[34] karpet. karpet/Dezi; 2016. Accessed on 20.03.2017. Available from: https://github.com/karpet/Dezi.

[35] Karman P. Dezi::Config;. Accessed 23.04.17. Available from: https://metacpan.org/pod/Dezi::Config.

[36] Karman P. Dezi::Aggregator::DBI;. Accessed 23.04.17. Available from: https://metacpan.org/pod/Dezi::Aggregator::DBI.

[37] Apache. Index of /lucene/solr/6.4.2; 2017. Accessed on 22.03.2017. Available from: http://apache.uib.no/lucene/solr/6.4.2/.

[38] Targett C. Spatial Search; 2017. Accessed 23.04.17. Available from: https://cwiki.apache.org/confluence/display/solr/Spatial+Search.

[39] Bernstein J. Streaming Expressions; 2017. Accessed 23.04.17. Available

from: https://cwiki.apache.org/confluence/display/solr/Streaming+Expressions.

[40] Targett C. Faceting; 2017. Accessed 23.04.17. Available from: https://cwiki.apache.org/confluence/display/solr/Faceting.

[41] Sphinx. Sphinx 2.3.2-beta downloads; 2016. Accessed on 22.03.2017. Available from: http://sphinxsearch.com/downloads/beta/.

[42] Sphinx. Sphinx 2.3.2-beta reference manual; 2016. Accessed 23.04.17. Available from: http://sphinxsearch.com/docs/devel.html#searching.

[43] Lemur. The Lemur Project; 2017. Accessed on 22.03.2017. Available from: https://sourceforge.net/projects/lemur/.

[44] lemur project. Sifaka; 2016. Accessed 23.04.17. Available from: http://www.lemurproject.org/sifaka.php.

[45] Opensearchserver. Configuring facets;. Accessed 24.04.17. Available from: http://www.opensearchserver.com/documentation/clients/php_client/facets.md.

[46] emmanuel keller. OpenSearchServer; 2017. Accessed on 22.03.2017. Available from: https://github.com/jaeksoft/opensearchserver.

[47] OpenSearchServer. Downloads and documentation; 2017. Accessed on 22.03.2017. Available from: http://www.opensearchserver.com/.

[48] romseygeek. flaxsearch/luwak; 2017. Accessed on 26.03.2017. Available from: https://github.com/flaxsearch/luwak.

[49] Julien. Advanced search feature (Datafari 3.2 and above); 2017. Accessed 24.04.17. Available from: https://datafari.atlassian.net/wiki/pages/viewpage.action?pageId=61282866.

[50] julienFL. francelabs/datafari; 2017. Accessed on 26.03.2017. Available from: https://github.com/francelabs/datafari.

[51] ElasticSearch. Full text search;. Accessed 24.04.17. Available from: https://www.elastic.co/guide/en/elasticsearch/guide/current/full-text-search.html.

[52] ElasticSearch. Facets;. Accessed 24.04.17. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-facets.html.

[53] ElasticSearch. Geo Distance query;. Accessed 24.04.17. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-geo-distance-query.html.

[54] ElasticSearch. Fuzzy query;. Accessed 24.04.17. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html.

[55] ElasticSearch. Phonetic Matching;. Accessed 24.04.17. Available from: https://www.elastic.co/guide/en/elasticsearch/guide/current/phonetic-matching.html.

[56] elasticsearch. elasticsearch; 2017. Accessed 24.04.17. Available from: https://github.com/elastic/elasticsearch.

[57] Groonga. Characteristics of Groonga; 2017. Accessed 24.04.17. Available from: http://groonga.org/docs/characteristic.html#groonga-overview.

[58] Groonga. The latest release; 2017. Accessed 24.04.17. Available from: http://groonga.org/.

[59] tantivy. tantivy; 2017. Accessed 24.04.17. Available from: https://github.com/tantivy-search/tantivy.

[60] TNTsearch. TNTsearch; 2017. Accessed 24.04.17. Available from: https://github.com/teamtnt/tntsearch.

[61] pouchdb-quick search. pouchdb-quick-search; 2017. Accessed 24.04.17. Available from: https://github.com/nolanlawson/pouchdb-quick-search.

[62] Search OS. Open Semantic Search;. Accessed 25.04.17. Available from: https://www.opensemanticsearch.org/.

[63] Search OS. Open Semantic Search; 2017. Accessed 25.04.17. Available from: https://github.com/opensemanticsearch/open-semantic-search-apps.

[64] Krellenstein M. Starting a Search Application; 2009. Accessed 25.04.17. Available from: https://whitepapers.em360tech.com/wp-content/files_mf/white_paper/lucid2.pdf.

[65] Lucidworks. Full Text Search Engines vs. DBMS (whitepaper);. Accessed 25.04.17. Available from: https://lucidworks.com/2009/09/02/full-text-search-engines-vs-dbms/.

[66] Guarino A. In: Reimer H, Pohlmann N, Schneider W, editors. Digital Forensics as a Big Data Challenge. Wiesbaden: Springer Fachmedien Wiesbaden; 2013. p. 197–203. Available from: http://dx.doi.org/10.1007/978-3-658-03371-2_17.

[67] Cleverley PH, Burnett S. Retrieving haystacks: a data driven information needs model for faceted search. Journal of Information Science. 2015;41(1):97–113. Available from: http://dx.doi.org/10.1177/0165551514554522.

[68] Li J, Wang Q, Wang C, Cao N, Ren K, Lou W. Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In: 2010 Proceedings IEEE INFOCOM; 2010. p. 1–5.

[69] Ji S, Li G, Li C, Feng J. Efficient Interactive Fuzzy Keyword Search. In: Proceedings of the 18th International Conference on World Wide Web. WWW '09. New York, NY, USA: ACM; 2009. p. 371–380. Available from: http://doi.acm.org/10.1145/1526709.1526760.

[70] Solutions VI. Approximate Matching (whitepaper); 2008. Accessed 25.04.17. Available from: http://viewds.com/images/pdf/Whitepapers/approximate%20matching%202.pdf.

[71] Elmes GA, Roedl G, Conley J. Forensic GIS: The Role of Geospatial Technologies for Investigating Crime and Providing Evidence. Springer Publishing Company, Incorporated; 2014.

[72] Selvi RT, Raj EGDP. An Approach to Improve Precision and Recall for Ad-hoc Information Retrieval Using SBIR Algorithm. In: 2014 World Congress on Computing and Communication Technologies; 2014. p. 137–141.

[73] Cai J, Shao X, Ma W. Ontology Driven Semantic Search over Structure P2P Network. In: 2009 Ninth International Conference on Hybrid Intelligent Systems. vol. 3; 2009. p. 29–34.

[74] Bonino D, Corno F, Farinetti L, Bosca A. Ontology Driven Semantic Search. In: WSEAS International Journal Multimedia and Image Processing (IJMIP), Volume 2, Issues 1/2, March/June 2012 Copyright © 2012, Infonomics Society 148 Transaction on Information Science and Application, Issue 6; 2004. p. 1597–1605.

[75] Kleppmann M. real time full text search with luwak and samza; 2015. Ac-

cessed 29.04.17. Available from: https://www.confluent.io/blog/real-time-full-text-search-with-luwak-and-samza/.

[76] Data I. Building a Streaming Search Platform; 2016. Accessed 29.04.17. Available from: https://blog.insightdatascience.com/building-a-streaming-search-platform-61a0d5a323a8.

[77] Lillis D, Scanlon M. In: Park JJJH, Jin H, Jeong YS, Khan MK, editors. On the Benefits of Information Retrieval and Information Extraction Techniques Applied to Digital Forensics. Singapore: Springer Singapore; 2016. p. 641–647. Available from: http://dx.doi.org/10.1007/978-981-10-1536-6_83.

[78] forensicswiki. Tools; 2017. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: http://www.forensicswiki.org/wiki/Tools#Open_Source_Tools.

[79] forensicswiki. Tools:Data Recovery; 2017. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: http://www.forensicswiki.org/wiki/Tools:Data_Recovery.

[80] forensicswiki. Tools:File Analysis; 2015. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: http://www.forensicswiki.org/wiki/Tools:File_Analysis.

[81] cugu. awesome-forensics; 2016. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: https://github.com/cugu/awesome-forensics#live-forensics.

[82] rshipp. awesome-malware-analysis; 2017. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: https://github.com/rshipp/awesome-malware-analysis#file-carving.

[83] wikipedia. List of digital forensics tools; 2017. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: https://en.wikipedia.org/wiki/List_of_digital_forensics_tools.

[84] encasefinal. Digital Forensic Analysis, EnCase; 2011. Last accessed (DD/MM/YYYY) 06/10/2017. Available from: http://encasefinal.blogspot.no/2011/07/open-source-tools.html.

[85] Sluethkit. Download; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://www.sleuthkit.org/sleuthkit/download.php.

[86] vstinner. hachoir3; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/vstinner/hachoir3.

[87] volatilityfoundation. Releases; 2016. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: http://www.volatilityfoundation.org/releases.

[88] google/grr. google/grr; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/google/grr.

[89] cgsecurity. testdisk; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/cgsecurity/testdisk.

[90] forensicswiki. Bulk extractor; 2015. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: http://www.forensicswiki.org/wiki/Bulk_extractor.

[91] simsong. bulk_extractor; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/simsong/bulk_extractor.

[92] mozilla. mig; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/mozilla/mig.

[93] gvoncken. guymager; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Avail-

able from: https://sourceforge.net/projects/guymager/?source=navbar.

[94] google. rekall; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/google/rekall.

[95] fireeye. flare-floss; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/fireeye/flare-floss.

[96] ShaneK2. inVtero.net; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://github.com/ShaneK2/inVtero.net.

[97] wireshark. About Wireshark; 2017. Last accessed (DD/MM/YYYY) 12/12/2017. Available from: https://www.wireshark.org/#download.

[98] Carrier B. Analysis Features;. Last accessed (DD/MM/YYYY) 08/10/2017. Available from: https://www.sleuthkit.org/autopsy/features.php.

[99] Carrier B. Keyword Search and Indexing;. Last accessed (DD/MM/YYYY) 08/10/2017. Available from: https://www.sleuthkit.org/autopsy/keyword.php.

[100] Carrier B. The sluth kit and open source digital Forensic conferance - Autopsy 3.0; 2012. Last accessed (DD/MM/YYYY) 08/10/2017. Available from: https://www.osdfcon.org/presentations/2012/OSDF-2012-Autopsy-3-0-Brian-Carrier.pdf.

[101] sleuthkit. About File Search; 2015. Last accessed (DD/MM/YYYY) 08/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/3.1/file_search_page.html#how_to_open_file_search.

[102] sleuthkit. Keyword Search; 2015. Last accessed (DD/MM/YYYY) 08/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/3.1/keyword_search.html#keyword_search_configuration_dialog.

[103] sleuthkit. Overview;. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://www.sleuthkit.org/autopsy/help/srch_mode.html.

[104] sleuthkit. Ad Hoc Keyword Search; 2017. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/4.3/ad_hoc_keyword_search_page.html.

[105] Carrier B. CONTENTS;. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://www.sleuthkit.org/informer/sleuthkit-informer-15.txt.

[106] sleuthkit. Reporting; 2015. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/3.1/reporting_page.html.

[107] sleuthkit. Autopsy User's Guide; 2015. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/3.1/index.html.

[108] sleuthkit. UI Layout; 2016. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: https://sleuthkit.org/autopsy/docs/user-docs/4.0/uilayout_page.html.

[109] sleuthkit/autopsy. Different character encodings #129; 2013. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: https://github.com/sleuthkit/autopsy/issues/129.

[110] sleuthkit. Hash Database Help;. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://www.sleuthkit.org/autopsy/help/hash_db.html.

[111] sleuthkit. Keyword Search Module; 2017. Last accessed (DD/MM/YYYY)

09/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/4.3/keyword_search_page.html.

[112] sleuthkit. Embedded File Extraction Module; 2015. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/3.1/embedded_file_extractor_page.html.

[113] sleuthkit. Interesting Files Module; 2015. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: http://sleuthkit.org/autopsy/docs/user-docs/3.1/interesting_page.html.

[114] wiki sleuthkit. PTK; 2013. Last accessed (DD/MM/YYYY) 09/10/2017. Available from: https://wiki.sleuthkit.org/index.php?title=PTK.

[115] fossies. contrib.plugins.psdispscan.PSDispScan Class Reference;. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://fossies.org/dox/volatility-2.6/classcontrib_1_1plugins_1_1psdispscan_1_1PSDispScan.html.

[116] volatilityfoundation. volatility/volatility/poolscan.py; 2015. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility/blob/master/volatility/poolscan.py.

[117] volatilityfoundation. volatility/volatility/scan.py; 2014. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility/blob/master/volatility/scan.py.

[118] volatilityfoundation. Command Reference Mal; 2017. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Mal.

[119] volatility labs. Automating Detection of Known Malware through Memory Forensics; 2016. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://volatility-labs.blogspot.no/2016/08/automating-detection-of-known-malware.html.

[120] volatilityfoundation. volatilityfoundation/volatility; 2017. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility.

[121] volatilityfoundation. volatility/volatility/win32/tasks.py; 2016. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility/blob/69142099447a5248ebdb9e3ba636738d509b7055/volatility/win32/tasks.py.

[122] volatilityfoundation. Volatility; 2015. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://www.aldeid.com/wiki/Volatility#psscan.

[123] volatilityfoundation. volatility/volatility/plugins/malware/malfind.py; 2017. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility/blob/1ace3c4e0e1b85e97f5357a3b1f35b198868ac75/volatility/plugins/malware/malfind.py.

[124] JamesHabben. JamesHabben/evolve; 2015. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/JamesHabben/evolve.

[125] volatilityfoundation. Unified Output; 2016. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/volatilityfoundation/volatility/wiki/Unified-Output.

[126] mig. mig/doc/concepts.rst; 2017. Last accessed (DD/MM/YYYY) 11/10/2017.

Available from: https://github.com/mozilla/mig/blob/master/doc/concepts.rst.

[127] mig. Mozilla InvestiGator: File module; 2016. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/mozilla/mig/blob/master/modules/file/doc.rst#search-paths.

[128] MIG. mig/modules/memory/doc.rst; 2015. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/mozilla/mig/blob/master/modules/memory/doc.rst.

[129] MIG. mig/conf/mig-agent.cfg.inc; 2017. Last accessed (DD/MM/YYYY) 11/10/2017. Available from: https://github.com/mozilla/mig/blob/a2fe0fed53fb75d6c1ece5f917268c79774ca0ef/conf/mig-agent.cfg.inc.

[130] hachoir. Docs/hachoir-metadata program;. Last accessed (DD/MM/YYYY) 12/10/2017. Available from: http://hachoir3.readthedocs.io/metadata.html.

[131] hachoir. hachoir-subfile program;. Last accessed (DD/MM/YYYY) 12/10/2017. Available from: http://hachoir3.readthedocs.io/subfile.html.

[132] hachoir. Hachoir3 for developers;. Last accessed (DD/MM/YYYY) 12/10/2017. Available from: http://hachoir3.readthedocs.io/developer.html#why-using-hachoir-parsers.

[133] hachoir. hachoir.regex module;. Last accessed (DD/MM/YYYY) 12/10/2017. Available from: http://hachoir3.readthedocs.io/regex.html.

[134] elastic. Search;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-search.html#search-multi-index-type.

[135] elastic. Search APIs;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search.html.

[136] elastic. URI Search;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-uri-request.html.

[137] elastic. Sort;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-sort.html.

[138] elastic. Source filtering;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-source-filtering.html.

[139] Elastic. Script Fields;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-script-fields.html.

[140] Elastic. Post filter;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-post-filter.html.

[141] Elastic. Highlighting;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-highlighting.html.

[142] Elastic. Scroll;. Last accessed (DD/MM/YYYY) 15/10/2017. Available

from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-scroll.html.

[143] Elastic. Elasticsearch: The Definitive Guide [2.x] / Search in Depth / Controlling Relevance / Query-Time Boosting;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/guide/current/query-time-boosting.html.

[144] Elastic. Elasticsearch Reference [5.6] / Search APIs / Request Body Search / Index Boost;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-index-boost.html.

[145] Elastic. Elasticsearch: The Definitive Guide [2.x] / Getting Started / Sorting and Relevance / What Is Relevance?;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html.

[146] Elastic. Elasticsearch Reference [5.6] / Search APIs / Request Body Search / min_score;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-min-score.html.

[147] Elastic. Elasticsearch Reference [5.6] / Search APIs / Request Body Search / Search After;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-search-after.html.

[148] Elastic. Elasticsearch Reference [5.6] / Search APIs / Request Body Search / Field Collapsing;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-collapse.html#CO27-2.

[149] Elastic. Elasticsearch Reference [5.6] / Search APIs / Search Shards API;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-shards.html.

[150] Elastic. Elasticsearch Reference [5.6] / Search APIs / Profile API;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-profile.html.

[151] Elastic. Elasticsearch Reference [5.6] / Aggregations;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html.

[152] Elastic. Elasticsearch Reference [5.6] / Indices APIs / Clear Cache;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-clearcache.html.

[153] Elastic. Elasticsearch Reference [5.6] / Query DSL;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html.

[154] Elastic. Elasticsearch Reference [5.6] / API Conventions / Common options;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/common-options.html#fuzziness.

[155] Elastic. Elasticsearch Reference [5.6] / Query DSL / Full text queries /

Match Query;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query.html.

[156] Elastic. Elasticsearch Reference [5.6] / Query DSL / Full text queries / Simple Query String Query;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-simple-query-string-query.html.

[157] Elastic. Elasticsearch Reference [5.6] / Query DSL / Term level queries;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/term-level-queries.html.

[158] Elastic. Elasticsearch Reference [5.6] / Query DSL / Compound queries / Boosting Query;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-boosting-query.html.

[159] Elastic. Elasticsearch Reference [5.6] / Query DSL / Joining queries;. Last accessed (DD/MM/YYYY) 15/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/joining-queries.html.

[160] Grand A. Store compression in Lucene and Elasticsearch; 2015. Last accessed (DD/MM/YYYY) 16/10/2017. Available from: https://www.elastic.co/blog/store-compression-in-lucene-and-elasticsearch.

[161] elastic. Elasticsearch Reference [5.6] / Modules / Thread Pool;. Last accessed (DD/MM/YYYY) 16/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-threadpool.html.

[162] Mohan V. Eliminating Duplicate Documents in Elasticsearch; 2015. Last accessed (DD/MM/YYYY) 16/10/2017. Available from: https://qbox.io/blog/minimizing-document-duplication-in-elasticsearch.

[163] Elastic. Elasticsearch Reference [5.6] / Analysis / Token Filters / Stemmer Token Filter;. Last accessed (DD/MM/YYYY) 16/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-stemmer-tokenfilter.html.

[164] Elastic. Elasticsearch: The Definitive Guide [2.x] / Dealing with Human Language / Normalizing Tokens / In That Case;. Last accessed (DD/MM/YYYY) 16/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/guide/current/lowercase-token-filter.html.

[165] ElasticSearch. Elasticsearch Reference [5.6] » Mapping / Mapping parameters / boost;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-boost.html.

[166] Alger S. Elastic Team Member - Export data to csv file; 2017. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: https://discuss.elastic.co/t/export-data-to-csv-file/80250.

[167] Solr. Overview of Searching in Solr;. Last accessed (DD/MM/YYYY) 16/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/overview-of-searching-in-solr.html.

[168] Solr. Common Query Parameters;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/common-query-p

arameters.html#fq-filter-query-parameter.

[169] Solr. The Standard Query Parser;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/the-standard-query-parser.html.

[170] Solr. The DisMax Query Parser;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/the-dismax-query-parser.html.

[171] Solr. Highlighting;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/the-dismax-query-parser.html.

[172] Solr. Transforming Result Documents;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/transforming-result-documents.html.

[173] Solr. Exporting Result Sets;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/exporting-result-sets.html.

[174] Solr. Suggester;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/suggester.html.

[175] Solr. MoreLikeThis;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/morelikethis.html.

[176] Solr. Result Clustering;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/result-clustering.html.

[177] Solr. De-Duplication;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: https://lucene.apache.org/solr/guide/6_6/de-duplication.html.

[178] Solr. Result Grouping;. Last accessed (DD/MM/YYYY) 20/10/2017. Available from: http://lucene.apache.org/solr/guide/7_0/result-grouping.html.

[179] wikiApache. General - What is Solr?; 2016. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: https://wiki.apache.org/solr/FAQ#Why_don.27t_International_Characters_Work.3F.

[180] WikiApache. Specifying a Query Parser; 2015. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: https://wiki.apache.org/solr/SolrQuerySyntax.

[181] WikiApache. Analyzers, Tokenizers, and Token Filters; 2016. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: https://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters#solr.EdgeNGramFilterFactory.

[182] WikiApache. Solr Relevancy FAQ; 2017. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: https://wiki.apache.org/solr/SolrRelevancyFAQ.

[183] Solr. Detecting Languages During Indexing;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: https://lucene.apache.org/solr/guide/6_6/detecting-languages-during-indexing.html.

[184] sphinx. Boolean query syntax;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: http://sphinxsearch.com/docs/latest/boolean-syntax.html.

[185] Sphinx. Quick summary of the ranking factors;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: http://sphinxsearch.com/docs/latest/ranking

-factors.html.

[186] Sphinx. Field-level ranking factors;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: http://sphinxsearch.com/docs/latest/field-factors.html.

[187] Sphinx. Sorting modes;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: http://sphinxsearch.com/docs/latest/sorting-modes.html.

[188] Sphinx. Grouping (clustering) search results;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: http://sphinxsearch.com/docs/latest/clustering.html.

[189] Sphinx. Distributed searching;. Last accessed (DD/MM/YYYY) 21/10/2017. Available from: http://sphinxsearch.com/docs/latest/distributed.html.

[190] Sphinx. regexp_filter;. Last accessed (DD/MM/YYYY) 22/10/2017. Available from: http://sphinxsearch.com/docs/current/conf-regexp-filter.html.

[191] Sphinx. 12.2.7. dict;. Last accessed (DD/MM/YYYY) 22/10/2017. Available from: http://sphinxsearch.com/docs/current.html#conf-enable-star.

[192] Sphinx. Sphinx features;. Last accessed (DD/MM/YYYY) 22/10/2017. Available from: http://sphinxsearch.com/docs/latest/features.html.

[193] Sphinx. BuildExcerpts;. Last accessed (DD/MM/YYYY) 22/10/2017. Available from: http://sphinxsearch.com/docs/latest/api-func-buildexcerpts.html.

[194] Yannikos Y, Graner L, Steinebach M, Winter C. In: Peterson G, Shenoi S, editors. Data Corpora for Digital Forensics Education and Research. Berlin, Heidelberg: Springer Berlin Heidelberg; 2014. p. 309–325. Available from: https://doi.org/10.1007/978-3-662-44952-3_21.

[195] Grajeda C, Breitinger F, Baggili I. Availability of datasets for digital forensics. And what is missing. Digital Investigation. 2017;22(Supplement):S94 – S105. Available from: http://www.sciencedirect.com/science/article/pii/S1742287617301913.

[196] Grajeda C, Breitinger F, Baggili I. DATASETS FOR CYBER FORENSICS; 2017. Last accessed (DD/MM/YYYY) 19/09/2017. Available from: http://datasets.fbreitinger.de/datasets/.

[197] Abt S, Baier H. Are We Missing Labels? A Study of the Availability of Ground-Truth in Network Security Research. In: Proceedings of the 2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security. BADGERS '14. Washington, DC, USA: IEEE Computer Society; 2014. p. 40–55. Available from: http://dx.doi.org/10.1109/BADGERS.2014.11.

[198] Schler J, Koppel M, Argamon S, Pennebaker J. In: Effects of age and gender on blogging. vol. SS-06-03; 2006. p. 191–197. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: http://u.cs.biu.ac.il/~schlerj/schler_springsymp06.pdf.

[199] u cs biu ac il. The Blog Authorship Corpus;. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: http://u.cs.biu.ac.il/~koppel/BlogCorpus.htm.

[200] Nunes E, Shakarian P, Simari GI, Ruef A. Argumentation Models for Cyber Attribution. CoRR. 2016;abs/1607.02171. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: https://arxiv.org/pdf/1607.02171v1.pdf.

[201] Nunes E. CTF data;. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: https://www.dropbox.com/sh/17d4eyg0cwoxg8s/AAA5g1NvQw-tUoZPvldloddRa?dl=0.

[202] Marcinczuk M, Zasko-Zielinska M, Piasecki M. Structure Annotation in the Polish Corpus of Suicide Notes. In: Text, Speech and Dialogue - 14th International Conference, TSD 2011, Pilsen, Czech Republic, September 1-5, 2011. Proceedings; 2011. p. 419–426. Available from: https://doi.org/10.1007/978-3-642-23538-2_53.

[203] Zaśko-Zielińska M, Piasecki M, Marcińczuk M. Polski korpus listów pożegnalnych samobójców; 2015. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.pcsn.uni.wroc.pl/.

[204] Brennan M, Greenstadt R. Practical Attacks Against Authorship Recognition Techniques. In: Proceedings of the 21st Innovative Applications of Artificial Intelligence Conference, IAAI-09; 2009. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: https://www.cs.drexel.edu/~greenie/brennan_paper.pdf.

[205] psal cs drexel edu. JStylo-Anonymouth; 2013. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: https://psal.cs.drexel.edu/index.php/JStylo-Anonymouth.

[206] Luyckx K, Daelemans W. Personae: a Corpus for Author and Personality Prediction from Text. In: LREC. European Language Resources Association; 2008. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.lrec-conf.org/proceedings/lrec2008/pdf/759_paper.pdf.

[207] Luyckx K, Daelemans W. Personae Corpus; 2017. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: https://www.clips.uantwerpen.be/datasets/personae-corpus.

[208] Argamon S, Juola P. Overview of the International Authorship Identification Competition at PAN-2011. In: Petras V, Forner P, Clough P, editors. Notebook Papers of CLEF 2011 Labs and Workshops, 19-22 September, Amsterdam, Netherlands; 2011. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.uni-weimar.de/medien/webis/events/pan-11/pan11-papers-final/pan11-author-identification/argamon11-overview.pdf.

[209] pan webis de. Evaluation Data; 2016. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://pan.webis.de/data.html.

[210] Juola P. An Overview of the Traditional Authorship Attribution Subtask. In: Forner P, Karlgren J, Womser-Hacker C, editors. CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy; 2012. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.uni-weimar.de/medien/webis/events/pan-12/pan12-papers-final/pan12-author-identification/juola12-overview.pdf.

[211] Juola P, Stamatatos E. Overview of the Author Identification Task at PAN 2013. In: Forner P, Navigli R, Tufis D, editors. CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September, Valencia, Spain; 2013. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.uni-weimar.de/medien/webis/events/pan-13/pan13-papers-final/pan13-authorship-verification/juola13-overview.pdf.

[212] Stamatatos E, Daelemans W, Verhoeven B, Potthast M, Stein B, Juola P,

et al. Overview of the Author Identification Task at PAN 2014. In: Cappellato L, Ferro N, Halvey M, Kraaij W, editors. CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers, 15-18 September, Sheffield, UK. CEUR-WS.org; 2014. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.uni-weimar.de/medien/webis/events/pan-14/pan14-papers-final/pan14-authorship-verification/stamatatos14-overview.pdf.

[213] Stamatatos E, amd Ben Verhoeven WD, Juola P, López-López A, Potthast M, Stein B. Overview of the Author Identification Task at PAN 2015. In: Cappellato L, Ferro N, Jones G, San Juan E, editors. CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers, 8-11 September, Toulouse, France. CEUR-WS.org; 2015. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://www.uni-weimar.de/medien/webis/events/pan-15/pan15-papers-final/pan15-authorship-verification/stamatatos15-overview.pdf.

[214] Halvani O, Winter C, Graner L. On the Usefulness of Compression Models for Authorship Verification. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. ARES '17. New York, NY, USA: ACM; 2017. p. 54:1–54:10. Available from: http://doi.acm.org/10.1145/3098954.3104050.

[215] Halvani O, Winter C, Graner L. ARES_WSDF2017; 2017. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: https://www.dropbox.com/sh/f2mlp6u5vervx9b/AABr_c7qrmahCqUviIu3ORz6a?dl=0.

[216] schonlau. Masquerading User Data;. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: http://www.schonlau.net/intrusion.html.

[217] Wang k, Stolfo S. One-Class Training for Masquerade Detection. 2003 01;Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.researchgate.net/publication/247054265_One-Class_Training_for_Masquerade_Detection.

[218] netresec. Publicly available PCAP files; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.netresec.com/?page=PcapFiles.

[219] malware-traffic analysis. A source for pcap files and malware samples...; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://malware-traffic-analysis.net/.

[220] pcapr. Welcome to pcapr, where pcaps come alive.;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.pcapr.net/home.

[221] evilfingers. PCAP Repository; 2010. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.evilfingers.com/repository/index.php.

[222] caida. CAIDA Data - Overview of Datasets, Monitors, and Reports; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.caida.org/data/overview/.

[223] mining group C. Datasets;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://csmining.org/index.php/data.html.

[224] azsecure data. Get Data; •. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.azsecure-data.org/get-data.html.

[225] Corpora D. Corpora;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://digitalcorpora.org/corpora.

[226] Harrison S. The Global Inteligence Files;. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://wikileaks.org/the-gifiles.html.

[227] wlstorage net. Index of /torrent/gifiles/;. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://wlstorage.net/torrent/gifiles/.

[228] Kaggle. Hillary Clinton's Emails; 2016. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.kaggle.com/kaggle/hillary-clinton-emails.

[229] Tatman R. Fraudulent E-mail Corpus CLAIR collection of "Nigerian" fraud emails; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.kaggle.com/rtatman/fraudulent-email-corpus.

[230] Ruano-Ordas D. Corpus 200 Emails. 2015 3;Available from: https://figshare.com/articles/Corpus_200_Emails/1326662.

[231] Enrondata. Data;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://enrondata.readthedocs.io/en/latest/references/data/.

[232] Dang-Nguyen DT, Pasquini C, Conotter V, Boato G. RAISE: A Raw Images Dataset for Digital Image Forensics. In: Proceedings of the 6th ACM Multimedia Systems Conference. MMSys '15. New York, NY, USA: ACM; 2015. p. 219–224. Available from: http://doi.acm.org/10.1145/2713168.2713194.

[233] Dang-Nguyen DT, Pasquini C, Conotter V, Boato G. Introducing RAISE dataset;. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: http://mmlab.science.unitn.it/RAISE/.

[234] Mirsky Y, Shabtai A, Rokach L, Shapira B, Elovici Y. SherLock vs Moriarty: A Smartphone Dataset for Cybersecurity Research. In: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security. AISec '16. New York, NY, USA: ACM; 2016. p. 1–12. Available from: http://doi.acm.org/10.1145/2996758.2996764.

[235] Mirsky Y, Shabtai A, Rokach L, Shapira B, Elovici Y. DOWNLOADS; 2016. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: http://bigdata.ise.bgu.ac.il/sherlock/#/download.

[236] Allix K, Bissyandé TF, Klein J, Traon YL. AndroZoo: Collecting Millions of Android Apps for the Research Community. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR); 2016. p. 468–471.

[237] androzoo. androzoo; 2016. Last accessed (DD/MM/YYYY) 24/09/2017. Available from: https://androzoo.uni.lu/.

[238] Pozzolo AD, Caelen O, Johnson RA, Bontempi G. Calibrating Probability with Undersampling for Unbalanced Classification. In: IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7-10, 2015; 2015. p. 159–166. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: https://www3.nd.edu/~dial/publications/dalpozzolo2015calibrating.pdf.

[239] Andrea. Credit Card Fraud Detection: Anonymized credit card transactions labeled as fraudulent or genuine; 2016. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: https://www.kaggle.com/dalpozz/creditcardfraud.

[240] K R S, Zareapoor M. FraudMiner: A Novel Credit Card Fraud Detection Model Based on Frequent Itemset Mining. 2014 09;2014:252797. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.researchgate.net/publication/266746615_FraudMiner_A_Novel_Credit_Card_Fraud_Detection_Model_Based_on_Frequent_Itemset_Mining.

[241] purdue. Index of /data/credit_card;. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.cs.purdue.edu/commugrate/data/credit_card/.

[242] cms. Dataset Downloads; 2017. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.cms.gov/OpenPayments/Explore-the-Data/Dataset-Downloads.html.

[243] CMS. Open Payments Public Use Files: Methodology Overview & Data Dictionary; 2017. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.cms.gov/OpenPayments/Downloads/OpenPaymentsDataDictionary.pdf.

[244] Lopez-Rojas E. Synthetic Financial Datasets For Fraud Detection - Synthetic datasets generated by the PaySim mobile money simulator; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.kaggle.com/ntnu-testimon/paysim1.

[245] Lopez-Rojas EA. Applying Simulation to the Problem of Detecting Financial Fraud. , Department of Computer Science and Engineering; 2016.

[246] Lopez-Rojas EA, Axelsson S. BankSim: A Bank Payment Simulation for Fraud Detection Research; 2014. Available from: https://www.researchgate.net/publication/265736405_BankSim_A_Bank_Payment_Simulation_for_Fraud_Detection_Research.

[247] Lopez-Rojas EA, Axelsson S. Synthetic data from a financial payment system - Synthetic datasets generated by the BankSim payments simulator; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.kaggle.com/ntnu-testimon/banksim1.

[248] Amerini I, Ballan L, Caldelli R, Del Bimbo A, Serra G. A SIFT-Based Forensic Method for Copy–Move Attack Detection and Transformation Recovery. Trans Info For Sec. 2011 Sep;6(3):1099–1110. Available from: http://dx.doi.org/10.1109/TIFS.2011.2129512.

[249] lambertoballan. sift-forensic; 2015. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://github.com/lambertoballan/sift-forensic/blob/master/README.md.

[250] Carrier B. Digital Forensics Tool Testing Images; 2010. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: http://dftt.sourceforge.net/.

[251] Corpora D. Real Data Corpus; 2017. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: https://digitalcorpora.org/corpora/disk-images/real-data-corpus.

[252] NIST. The CFReDS Project; 2016. Last accessed (DD/MM/YYYY) 20/09/2017. Available from: https://www.cfreds.nist.gov/.

[253] VirusShare. VirusShare.com - Because Sharing is Caring; 2017. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://virusshare.com/.

[254] Kaggle. Microsoft Malware Classification Challenge (BIG 2015);. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://www.kaggle.com/c/malware-classification/data.

[255] Arp D, Spreitzenbarth M, Gascon H, Rieck K. Drebin: Effective and explainable detection of android malware in your pocket; 2014. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://www.tu-braunschweig.de/Medien-DB/sec/pubs/2014-ndss.pdf.

[256] Arp D, Spreitzenbarth M, Gascon H, Rieck K. The Drebin Dataset; 2016. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://www.sec.cs.tu-bs.de/~danarp/drebin/.

[257] d Costa KAP, d Silva LA, Martins GB, Rosa GH, Pereira CR, Papa JP. Malware Detection in Android-Based Mobile Environments Using Optimum-Path Forest. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA); 2015. p. 754–759.

[258] RECOVI. DroidWare; 2015. Last accessed (DD/MM/YYYY) 23/09/2017. Available from: https://github.com/RECOVI/DroidWare.

[259] Bowen T, Poylisher A, Serban C, Chadha R, Chiang CYJ, Marvel LM. Enabling reproducible cyber research - four labeled datasets. In: MILCOM 2016 - 2016 IEEE Military Communications Conference; 2016. p. 539–544.

[260] McDaniel P. Data Sets;. Last accessed (DD/MM/YYYY) 23/09/2017. Available from: https://cybervan.appcomsci.com:9000/datasets.

[261] Kiss N, Lalande JF, Leslous M, Tong VVT. Kharon Dataset: Android Malware under a Microscope. In: The LASER Workshop: Learning from Authoritative Security Experiment Results (LASER 2016). San Jose, CA: USENIX Association; 2016. p. 1–12. Available from: https://www.usenix.org/conference/laser2016/program/presentation/kiss.

[262] Kharon-project. Kharon Malware Dataset; 2016. Last accessed (DD/MM/YYYY) 23/09/2017. Available from: http://kharon.gforge.inria.fr/dataset/.

[263] Avdiienko V, Kuznetsov K, Gorla A, Zeller A, Arzt S, Rasthofer S, et al. Mining Apps for Abnormal Usage of Sensitive Data. In: Proceedings of the 37th International Conference on Software Engineering. ICSE 2015; 2015. .

[264] Avdiienko V, Kuznetsov K, Gorla A, Zeller A. About MUDFLOW;. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: https://www.st.cs.uni-saarland.de/appmining/mudflow/.

[265] Lab IR. Datasets; 2010. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.uvic.ca/engineering/ece/isot/datasets/index.php#section0-0.

[266] ISOT. ISOT Dataset Overview;. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: https://www.uvic.ca/engineering/ece/isot/assets/docs/isot-datase.pdf.

[267] Saad S, Traoré I, Ghorbani AA, Sayed B, Zhao D, Lu W, et al. Detecting P2P botnets through network behavior analysis and machine learning. In: PST. IEEE; 2011. p. 174–180. Available from: http://ieeexplore.ieee.org/abstract/document/5971980/.

[268] lirmm. Analyzing Web Traffic ECML/PKDD 2007 Discovery Challenge; 2007. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: http://www.lirmm.fr/pkdd2007-challenge/index.html#dataset.

[269] isi csic es. HTTP DATASET CSIC 2010; 2012. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: http://www.isi.csic.es/dataset/.

[270] contagiodump. Collection of Pcap files from malware analysis; 2015. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://contagiodump.blogspot.no/2013/04/collection-of-pcap-files-from-malware.html.

[271] dropbox. PCAPS_TRAFFIC_PATTERNS fra DeepEnd Research (DeepEnd Re-

search);. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://www.dropbox.com/sh/7fo4efxhpenexqp/AACmuri_l-LDiVDUDJ3hVLqPa?dl=0.

[272] Dolan-Gavitt B. (Sys)Call Me Maybe: Exploring Malware Syscalls with PANDA; 2015. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://moyix.blogspot.no/search?q=dataset.

[273] Garcia S. Dataset; 2015. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://stratosphereips.org/category/dataset.html.

[274] Samani EBB, Jazi HH, Stakhanova N, Ghorbani AA. Towards effective feature selection in machine learning-based botnet detection approaches. In: IEEE Conference on Communications and Network Security, CNS 2014, San Francisco, CA, USA, October 29-31, 2014; 2014. p. 247–255. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6997492.

[275] UNB. Botnet dataset;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.unb.ca/cic/research/datasets/botnet.html.

[276] Abdul Kadir AF, Stakhanova N, Ghorbani AA. In: Qiu M, Xu S, Yung M, Zhang H, editors. Android Botnets: What URLs are Telling Us. Cham: Springer International Publishing; 2015. p. 78–91. Available from: https://doi.org/10.1007/978-3-319-25645-0_6.

[277] UNB. Android Botnet dataset;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.unb.ca/cic/research/datasets/android-botnet.html.

[278] ll mit edu. DARPA Intrusion Detection Data Sets;. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://ll.mit.edu/ideval/data/index.html.

[279] Haines JW, Lippman RP, Fried DJ, Zissman MA, Tran E, Boswell SB. 1999 DARPA INTRUSION DETECTION EVALUATION DESIGN AND PROCEDURES; 2001. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://ll.mit.edu/ideval/files/TR-1062.pdf.

[280] ll mit edu. 2000 DARPA Intrusion Detection Scenario Specific Data Sets;. Last accessed (DD/MM/YYYY) 21/09/2017. Available from: https://ll.mit.edu/ideval/data/2000data.html.

[281] Romain F, Pierre B, Patrice A, Kensuke F. MAWILab Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In: ACM CoNEXT 10. Philadelphia PA;. .

[282] fukuda lab. MAWILab v1.1; 2017. Last accessed (DD/MM/YYYY) 23/09/2017. Available from: http://www.fukuda-lab.org/mawilab/data.html.

[283] "kdd ics uci edu". KDD Cup 1999 Data; 1999. Last accessed (DD/MM/YYYY) 23/09/2017. Available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[284] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A Detailed Analysis of the KDD CUP 99 Data Set. In: Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications. CISDA'09. Piscataway, NJ, USA: IEEE Press; 2009. p. 53–58. Available from: http://dl.acm.org/citation.cfm?id=1736481.1736489.

[285] Moustafa N, Slay J. The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems. In: 2015 4th International

Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS); 2015. p. 25–31.

[286] unsw-adfa-edu au". The UNSW-NB15 data set description; 2016. Last accessed (DD/MM/YYYY) 24/09/2017. Available from: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/.

[287] states military academy west point U. Data Sets; 2009. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: Unitedstatesmilitaryacademywestpoint.

[288] Creech EITUCU Gideon. Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks. Awarded by:University of New South Wales. Engineering and Information Technology; 2014. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://unsworks.unsw.edu.au/fapi/datastream/unsworks:11913/SOURCE02?view=true.

[289] unsw. The ADFA Intrusion Detection Datasets; 2013. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-IDS-Datasets/.

[290] takakura. Traffic Data from Kyoto University's Honeypots; 2015. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: http://www.takakura.com/Kyoto_data/.

[291] SONG J, Takakura H, Okabe Y. Description of Kyoto University Benchmark Data;. Last accessed (DD/MM/YYYY) 01/10/2017. Available from: http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf.

[292] RAWDAD. All datasets and tools: sorted by name; 2017. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://crawdad.org/all-byname.html.

[293] automayt. A collection of ICS/SCADA PCAPs; 2016. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://github.com/automayt/ICS-pcap.

[294] amazon. Common Crawl on AWS;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: https://aws.amazon.com/public-datasets/common-crawl/.

[295] UNB. NSL-KDD dataset;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.unb.ca/cic/research/datasets/nsl.html.

[296] Lashkari AH, Gil GD, Mamun MSI, Ghorbani AA. Characterization of Tor Traffic using Time based Features. In: Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,. INSTICC. SciTePress; 2017. p. 253–262.

[297] UNB. Tor-nonTor dataset;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.unb.ca/cic/research/datasets/tor.html.

[298] Draper-Gil G, Lashkari AH, Mamun MSI, Ghorbani AA. Characterization of Encrypted and VPN Traffic using Time-related Features. In: Proceedings of the 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,. INSTICC. SciTePress; 2016. p. 407–414.

[299] UNB. VPN-nonVPN dataset;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.unb.ca/cic/research/datasets/vpn.html.

[300] Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic

approach to generate benchmark datasets for intrusion detection. Computers & Security. 2012;31(3):357 – 374. Available from: http://www.sciencedirect.com/science/article/pii/S0167404811001672.

[301] UNB. Intrusion detection evaluation dataset;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.unb.ca/cic/research/datasets/ids.html.

[302] Bonneau J. Yahoo Password Frequency Corpus. 2015 12;Available from: https://figshare.com/articles/Yahoo_Password_Frequency_Corpus/2057937.

[303] Blocki J, Datta A, Bonneau J. Differentially Private Password Frequency Lists. IACR Cryptology ePrint Archive. 2016;2016:153. Available from: http://eprint.iacr.org/2016/153.

[304] Granville V. Password and hijacked email dataset for you to test your data science skills; 2012. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://www.datasciencecentral.com/forum/topics/password-dataset-for-you-to-test-your-data-science-skills.

[305] Wood T, Tarasuk-Levin G, Shenoy P, Desnoyers P, Cecchet E, Corner MD. Memory Buddies: Exploiting Page Sharing for Smart Colocation in Virtualized Data Centers. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. VEE '09. New York, NY, USA: ACM; 2009. p. 31–40. Available from: http://doi.acm.org/10.1145/1508293.1508299.

[306] umass. Index of /traces/cpumem/memtraces; 2009. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://skuld.cs.umass.edu/traces/cpumem/memtraces/.

[307] umass. readme.txt;. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://skuld.cs.umass.edu/traces/cpumem/memtraces/readme.txt.

[308] Chen T, Kan MY. Creating a live, public short message service corpus: the NUS SMS corpus. Language Resources and Evaluation. 2013 Jun;47(2):299–335. Available from: https://doi.org/10.1007/s10579-012-9197-9.

[309] kite1988. nus-sms-corpus; 2016. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: https://github.com/kite1988/nus-sms-corpus.

[310] Wang D, Irani D, Pu C. Evolutionary Study of Web Spam: Webb Spam Corpus 2011 Versus Webb Spam Corpus 2006. In: Proceedings of the 2012 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012). COLLABORATECOM '12. Washington, DC, USA: IEEE Computer Society; 2012. p. 40–49. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: http://de-wang.org/download/webbspamcorpus2011.pdf.

[311] Wang D, Irani D, Pu C. Webb Spam Corpus 2011;. Last accessed (DD/MM/YYYY) 22/09/2017. Available from: https://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html.

[312] Delany SJ, Buckley M, Greene D. SMS spam filtering: Methods and data. Expert Systems with Applications. 2012;39(10):9899 – 9908. Available from: http://www.sciencedirect.com/science/article/pii/S0957417412002977.

[313] dublin institute of technology. DIT SMS Spam Dataset;. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://www.dit.ie/computing/research/resources/smsdata/.

[314] NIST. Spam Track; 2017. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://trec.nist.gov/data/spam.html.

[315] UCI. Spambase Data Set; 1999. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://archive.ics.uci.edu/ml/datasets/Spambase?ref=datanews.io.

[316] WEBSPAM-UK2007. "Web Spam Collections"; 2007. Crawled by the Laboratory of Web Algorithmics, University of Milan, http://law.di.unimi.it/.Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://chato.cl/webspam/datasets/uk2007/.

[317] Jun Liu(liukeen '@' mail xjtu cn) MZJMYL Hao Chen(lechenhao '@' gmail com). microblogPCU Data Set;. MOEKLINNS Lab, Department of Computer Science ,Xi'an Jiaotong University, China. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: https://archive.ics.uci.edu/ml/datasets/microblogPCU.

[318] NIST. Tweets2011; 2014. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://trec.nist.gov/data/tweets/.

[319] Dredze M, Gevaryahu R, Elias-Bachrach A. Learning Fast Classifiers for Image Spam.; 2007. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.8417&rep=rep1&type=pdf.

[320] jhu. Image Spam Dataset; 2007. Last accessed (DD/MM/YYYY) 02/10/2017. Available from: http://www.cs.jhu.edu/~mdredze/datasets/image_spam/.

[321] PhishTank. FAQ;. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://www.phishtank.com/faq.php#whatisphishing.

[322] millersmiles. about us; 2017. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://www.millersmiles.co.uk/aboutus.php.

[323] Mohammad R, McCluskey TL, Thabtah FA. Intelligent Rule based Phishing Websites Classification. IET Information Security. 2014 May;8(3):153–160. Available from: http://eprints.hud.ac.uk/id/eprint/17994/.

[324] UCI. Phishing Websites Data Set;. Last accessed (DD/MM/YYYY) 26/09/2017. Available from: http://archive.ics.uci.edu/ml/datasets/Phishing+Websites#.

[325] Cukierski W. The Enron Email Dataset - 500,000+ emails from 150 employees of the Enron Corporation; 2016. Last accessed (DD/MM/YYYY) 21/11/2017. Available from: https://www.kaggle.com/wcukierski/enron-email-dataset.

[326] Shalaginov A, Grini LS, Franke K. Understanding Neuro-Fuzzy on a class of multinomial malware detection problems. In: Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE; 2016. p. 684–691.

[327] Grini L, Shalaginov A, Franke K. Study of Soft Computing methods for large-scale multinomial malware types and families detection; 2016. .

[328] Apache W. General; 2013. Last accessed (DD/MM/YYYY) 18/11/2017. Available from: https://wiki.apache.org/solr/SolrTerminology.

[329] Elasticsearch. The Search API;. Last accessed (DD/MM/YYYY) 18/11/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/_the_search_api.html.

[330] Vlaswinkel K. How To Install Java with Apt-Get on Ubuntu 16.04; 2016. Last accessed (DD/MM/YYYY) 4/12/2017. Available from:

https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-get-on-ubuntu-16-04.

[331] Elasticsearch. Install Elasticsearch;. Last accessed (DD/MM/YYYY) 4/12/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html#deb-repo.

[332] howtoforge. How to install and configure Solr 6 on Ubuntu 16.04;. Last accessed (DD/MM/YYYY) 4/12/2017. Available from: https://www.howtoforge.com/tutorial/how-to-install-and-configure-solr-on-ubuntu-1604/.

[333] Solr. Getting Started with SolrCloud;. Last accessed (DD/MM/YYYY) 4/12/2017. Available from: https://lucene.apache.org/solr/guide/6_6/getting-started-with-solrcloud.html.

[334] Luracast. Import Multiple .sql dump files into mysql database from shell; 2012. Last accessed (DD/MM/YYYY) 16/11/2017. Available from: https://stackoverflow.com/questions/4708013/import-multiple-sql-dump-files-into-mysql-database-from-shell.

[335] isedwards. Error Code: 1290. The MySQL server is running with the –secure-file-priv option so it cannot execute this statement; 2015. Last accessed (DD/MM/YYYY) 16/11/2017. Available from: https://stackoverflow.com/questions/31951468/error-code-1290-the-mysql-server-is-running-with-the-secure-file-priv-option.

[336] devnull. How to remove lines shorter than XY?; 2014. Last accessed (DD/MM/YYYY) 16/11/2017. Available from: https://unix.stackexchange.com/questions/123243/how-to-remove-lines-shorter-than-xy.

[337] Botykai Z. How can I replace a newline using sed?; 2009. Last accessed (DD/MM/YYYY) 21/11/2017. Available from: https://stackoverflow.com/questions/1251999/how-can-i-replace-a-newline-n-using-sed.

[338] Rhyous. Removing all xml or html tags using Notepad++; 2012. Last accessed (DD/MM/YYYY) 24/11/2017. Available from: https://www.rhyous.com/2012/12/11/removing-all-xml-or-html-tags-using-notepad/.

[339] ElasticSearch. cat indices;. Last accessed (DD/MM/YYYY) 18/11/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/cat-indices.html.

[340] elastic. Create Index;. Last accessed (DD/MM/YYYY) 21/11/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html.

[341] Solr. RequestHandlers and SearchComponents in SolrConfig;. Last accessed (DD/MM/YYYY) 29/11/2017. Available from: https://lucene.apache.org/solr/guide/6_6/requesthandlers-and-searchcomponents-in-solrconfig.html.

[342] wikiapache. How can indexing be accelerated; 2016. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://wiki.apache.org/solr/FAQ#Why_does_the_request_time_out_sometimes_when_doing_commits.3F.

[343] manpagehaxx. URL;. Last accessed (DD/MM/YYYY) 6/12/2017. Available from: https://curl.haxx.se/docs/manpage.html.

[344] lucidworks. Understanding Transaction Logs, Soft Commit and Commit in SolrCloud;. Last accessed (DD/MM/YYYY) 7/12/2017. Available

from: https://lucidworks.com/2013/08/23/understanding-transaction-logs-softcommit-and-commit-in-sorlcloud/.

[345] Solr. IndexConfig in SolrConfig;. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://lucene.apache.org/solr/guide/6_6/indexconfig-in-solrconfig.html.

[346] Solrwiki. SolrCloud; 2017. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://wiki.apache.org/solr/SolrPerformanceProblems.

[347] Solr. Codec Factory;. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://lucene.apache.org/solr/guide/6_6/codec-factory.html.

[348] Elastic. index.codec;. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html.

[349] Hundley B. Small Static Dataset, 2-3 GB; 2015. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://qbox.io/blog/optimizing-elasticsearch-how-many-shards-per-index.

[350] wikiApache. Cache autoWarm Count Considerations; 2014. Last accessed (DD/MM/YYYY) 7/12/2017. Available from: https://wiki.apache.org/solr/SolrPerformanceFactors.

[351] cvandeplas. ELK-forensics; 2016. Last accessed (DD/MM/YYYY) 10/12/2017. Available from: https://github.com/cvandeplas/ELK-forensics.

[352] sleuthkit. Install and Configure Solr; 2016. Last accessed (DD/MM/YYYY) 10/12/2017. Available from: https://sleuthkit.org/autopsy/docs/user-docs/4.0/install_solr.html.

[353] Vandeplas C. BlueCoat Proxy log search and analytics with ELK; 2014. Last accessed (DD/MM/YYYY) 10/12/2017. Available from: http://christophe.vandeplas.com/search/label/elk.

[354] admin. awk line length and average; 2007. Last accessed (DD/MM/YYYY) 20/11/2017. Available from: https://coding-school.com/awk-line-length-and-average/.

[355] alexwlchan. Drawing multicolumn table in latex; 2014. Last accessed (DD/MM/YYYY) 04/11/2017. Available from: https://tex.stackexchange.com/questions/166263/drawing-multicolumn-table-in-latex.