

Fuzzy-Import Hashing: A Static Analysis Technique for Malware Detection

Nitin Naik^{a,*}, Paul Jenkins^b, Nick Savage^c, Longzhi Yang^d, Tossapon Boongoen^e and Natthakan Iam-On^e

^a*School of Informatics and Digital Engineering, Aston University, United Kingdom*

^b*Cardiff School of Technologies, Cardiff Metropolitan University, United Kingdom*

^c*School of Computing, University of Portsmouth, United Kingdom*

^d*Department of Computer and Information Sciences, Northumbria University, United Kingdom*

^e*Center of Excellence in AI and Emerging Technologies, School of Information Technology, Mae Fah Luang University, Thailand*

ARTICLE INFO

Keywords:

Malware Analysis
Fuzzy-Import Hashing
Fuzzy Hashing
Import Hashing
YARA Rules
Fuzzy C-Means Clustering
Ransomware

ABSTRACT

The advent of new malware types and their attack vectors poses serious challenges for security experts in discovering effective malware detection and analysis techniques. The preliminary step in malware analysis is filtering out samples of counterfeit malware from the suspicious samples by classifying them into most likely and unlikely malware categories. This will enable effective utilisation of resources and expertise for the most likely category of samples in subsequent stages and avoid nugatory effort. This process requires a very fast and resource-optimised method as it is applied on a large sample size. Fuzzy hashing and import hashing methods satisfy these requirements of malware analysis, though, with some limitations. Therefore, the proper integration of these methods, may overcome some of the limitations and improve the detection accuracy without affecting the overall performance of analysis. Hence, this paper proposes a fuzzy-import hashing technique, which is the integration of two methods, namely, fuzzy hashing and import hashing. This integration can offer several benefits such as an improved detection rate by complementing each other when one method cannot detect malware, then the other method can; and the generation of fuzzified results for subsequent clustering or classification, as the import hashing result can be easily merged with the fuzzy hashing result. The success of this proposed fuzzy-import hashing method is demonstrated through several experiments namely: on the collected malware and goodware corpus; a comparative evaluation against the established YARA rules and application in fuzzy c-means clustering.

1. Introduction


The volume of malware and number of attacks are increasing on a daily basis, which challenges security experts to develop more effective malware analysis techniques continuously (Naik, Shang, Jenkins and Shen, 2020h). In developing any malware analysis, one of the most important and initial activities is to eliminate a large number of unlikely malware samples from the corpus of suspicious samples. If this detection strategy is effective then it allows security analysts to concentrate on the most likely malware samples, thus improving the efficiency and outcomes of the analysis. This step in the analysis can be accomplished through either dynamic or static malware analysis (Harrell, 2013). Dynamic analysis can offer relatively detailed and precise information; however, static malware analysis is a relatively simple, quick, safe and lightweight activity to analyse suspicious samples. Several dynamic and static malware analysis techniques are in use, however, there is no “silver bullet” to apply for all types of malware successfully (Naik, Jenkins, Savage and Yang, 2019c).

There is always a requirement for an effective malware analysis technique as attackers are discovering novel and sophisticated means to evade existing detection methods (Naik,

Shang, Jenkins and Shen, 2020g; Naik, Jenkins, Savage and Yang, 2020b). Hashing techniques are fast and compact which suggests that they could be an effective technique for accomplishing basic static analysis to determine likely and unlikely malware, for example, fuzzy hashing and import hashing (Breitinger, Stivaktakis and Baier, 2013b; Naik et al., 2019c; Naik, Jenkins and Savage, 2019b). Nonetheless, these two hashing techniques have some limitations; for instance, fuzzy hashing provides better results only when the structure of files possess similarity, and import hashing provides better results only when import address tables of files possess similarity. This suggests that they may not be effective on several types of malware individually, however, they may be combined in such a way that improves the overall detection rate and performance. Therefore, this paper proposes a fuzzy-import hashing technique, which is the integration of fuzzy hashing and import hashing as a means to improve the overall detection rate and performance.

The fuzzy-import hashing operates by firstly applying import hashing on samples due to its speed and performance, and subsequently applying fuzzy hashing which is only applied on the remaining undetected samples, which could save significant computational overheads in comparison with applying fuzzy hashing to the complete sample. Additionally, fuzzy-import hashing can offer a number of other benefits for instance, both underlying methods can complement each others miss outs and improve the overall detection rate; im-

*Corresponding author

 n.naik1@aston.ac.uk (N. Naik)

ORCID(s): 0000-0002-0659-9646 (N. Naik)

port hashing results can be easily aligned with fuzzy hashing to produce common similarity scores for advanced analysis (e.g., clustering or classification). Thus, this proposed technique could improve the detection accuracy while maintaining the speed and overall performance. These benefits are demonstrated through several experiments performed on the collected malware and goodware samples, an application in fuzzy c-means clustering and a comparative evaluation against the established YARA rules.

The paper is organised into the following sections: Section 2 presents the selected malware analysis techniques import hashing, fuzzy hashing and YARA rules. Section 3 presents the malware (ransomware) related work and discusses some of the recently proposed static and dynamic detection method for ransomware. Section 4 explains the collection and verification process of chosen malware (ransomware) samples as well goodware samples. Section 5 presents the experimental evaluation of the selected methods: fuzzy hashing and import hashing on the collected malware and goodware. Section 6 discusses the proposed fuzzy-import hashing technique and its testing on the collected malware and goodware. Section 7 presents the comparative evaluation of the proposed fuzzy-import hashing with established malware analysis method YARA rules. Section 8 discusses the application of fuzzy-import hashing in Fuzzy C-Means (FCM) clustering and compares the FCM results of fuzzy-import hashing with the FCM results of their relative fuzzy hashing method. Section 9 discusses some of the main advantages and limitations of the proposed technique. Finally, Section 10 summarises the research work and suggests some future work.

2. Background: Employed Malware Analysis Techniques

Malware (MALicious softWARE) is a locution used to describe all software developed for disrupting, damaging or gaining access to data and systems in an unauthorised manner. Malware can be classified into several groups, dependent on several factors such as creator, variant, code, activity and severity. Malware analysis is the process of ascertaining the source, behaviour, and possible consequences of infection by an investigated malware sample. The analysis can be considered as either *static* or *dynamic*, where the former examines suspicious samples without the sample being executed or run. This essentially determines whether the suspicious sample is a piece of malware or not, with subsequent behaviour, technical indicators and possible consequences of infection determined later. This information may be further utilised to create detection signatures for malware discovery.

Alternatively, dynamic malware analysis examines suspicious samples by their execution in a controlled environment. Similar to static malware analysis, it determines all the features of the suspicious sample, although it is a more comprehensive analysis, observing the behaviour, actions and stages of the malware sample while in execution. For reverse engineering of complex malware, it is beneficial to

perform both static and dynamic analysis to complement each other instead of as an alternative to each other (Scott, 2017). Dynamic analysis can offer relatively detailed and precise information; however, static malware analysis is a relatively simple, quick, safe and lightweight activity to analyse suspicious samples. This work is mainly focused on developing an effective static malware analysis technique and therefore utilises static analysis techniques fuzzy hashing and import hashing.

2.1. Fuzzy Hashing

The use of cryptographic hash functions are not effective in every scenario in the field of cybersecurity, as they offer a Boolean result as to whether two files are identical. However, in malware analysis it the degree of similarity of files that are of interest. Furthermore, malware samples may not be identical but they often possess some similarity, as many malware writers utilise the same code to develop different variants of malware (Naik, Jenkins, Savage, Yang, Naik, Song, Boongoen and Iam-On, 2020f). Therefore, effective hashing techniques in malware analysis, are those which are able to find similarity between malware samples, and fuzzy hashing is one such technique that can be utilised to accomplish this goal (Kornblum, 2006). A fuzzy hashing technique divides the file of interest into several blocks and each block is treated separately for calculating its hash, finally, hashes of all the blocks are concatenated to obtain the fuzzy hash of that file (see [Figure 1](#)). A number of factors affect the size of the fuzzy hash of a file, comprising of the block size, the size of the file and the output size of the chosen hash function (Tridgell, 1999). Fuzzy hashing methods are divided into different types namely: Context-Triggered Piecewise Hashing (CTPH), Statistically-Improbable Features (SIF), Block-Based Hashing (BBH) and Block-Based Rebuilding (BBR) (Breitinger and Baier, 2012; Sadowski and Levin, 2007; Gayoso Martínez, Hernández Álvarez and Hernández Encinas, 2014). Forensic analysis of malware requires a thorough knowledge of the degree of similarity between known malware and inert files to assess files for their threat potential. This is vital when considering the analysis and clustering of suspected malware in order to discover new variants. Therefore, the use of the similarity preserving property of fuzzy hashing is useful in malware analysis while comparing unknown files with known malware families during malware analysis, where samples possess similar functionality, yet different cryptographic hash values (Naik et al., 2019c).

2.1.1. SSDEEP

The SSDEEP fuzzy hashing technique was specially created to distinguish spam or junk emails (Kornblum, 2006). It splits a file into several blocks depending on the data given in the file. These blocks and their endpoints are created by employing an Adler32 function involved in a rolling hash method (Tridgell, 1999). Subsequently, a hash is created for each block and finally, hashes of all the blocks are concatenated to obtain the fuzzy hash of that file. The Damerau-Levenshtein distance measure is used to compute

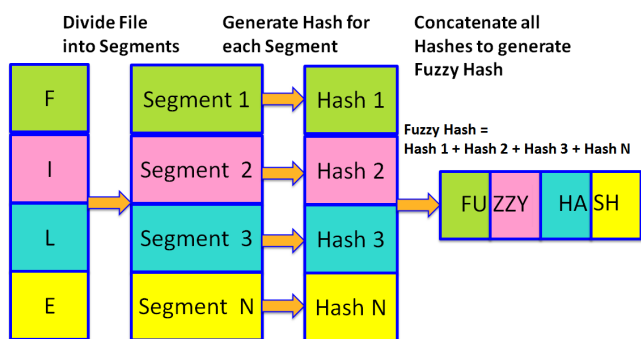


Figure 1: Fuzzy Hash generation process in a Fuzzy Hashing Method

the similarity distance of concerning files.

2.1.2. SDHASH

The SDHASH fuzzy hashing technique discovers common and uncommon attributes in a file and matches the uncommon attributes with those in another file to find the degree of similarity of interest (Roussev, 2010). Normally an attribute is a 64-byte string and is detected based on the calculation of entropy. The SDHASH fuzzy hash of a file is computed by employing SHA-1 hash function and Bloom filters. A Bloom filter is a probabilistic and space-efficient data structure used to establish that an element is a member or not a member of the set. The Hamming distance measure is used to compute the similarity distance of concerning files.

2.1.3. mvHASH-B

The mvHASH-B fuzzy hashing technique focuses on preserving the data unchanged in the case where there is a minor change between files, it ensures the same hash value while preserving the similarity. Nonetheless, mvHASH-B uses the concept of majority voting to transform the input data, encoding the majority vote bit sequence with Run-Length Encoding (RLE), and finally generating the mvHASH-B fuzzy hash employing Bloom filters (Breitinger, Astebøl, Baier and Busch, 2013a). Furthermore, it employs its own outlined hash function which is comparable with the standard SHA-1 function and having better run time efficiency than it.

2.1.4. TLSH

The TLSH fuzzy hashing technique is a Locality Sensitive Hashing (LSH) technique that hashes similar input items into the same buckets, and can be considered as a method to reduce the dimensionality of high-dimensional data (Oliver, Cheng and Chen, 2013). It constructs a TLSH hash value from a byte string using a sliding window of size 5 to populate an array of bucket counts (Oliver et al., 2013). Next, it calculates the quartile points, q_1 , q_2 and q_3 , constructing the digest/hash header values (first 3 bytes of the hash) and the digest/hash body by processing the bucket array (Oliver et al., 2013). The final TLSH digest/hash is the combination of the digest/hash header and the digest/hash body. The TLSH technique is different from other hash techniques which provide a similarity score between the range of 0 and 100, where 0 is

a mismatch and 100 is a perfect match. However, the TLSH technique uses a distance score between two digests/hashes of files, for comparison purposes, where a distance score of 0 represents that the files are identical and higher distance scores represent more differences between those files (Oliver et al., 2013).

2.2. Portable Executable (PE) File Based Hashing

The Portable Executable (PE) file format is a Microsoft Windows file format for executable/DLL files. The PE file format is a data structure that informs Windows OS loader what information is required for managing the wrapped executable code. This file format (PE), is derived from the Microsoft Common Object File Format (COFF). A PE file format has two main components the Header and Sections, which can be subdivided into several other small components as shown in Figure 2. All valid PE files are started with the memory address of 0x54AD, which represents the ASCII characters MZ (named after a former Microsoft architect Mark Zbikowski). PE file-based hashing techniques mainly generate their hashes from the various sections of a PE file.

2.2.1. IMPHASH

It is a PE file-based and one of the fastest hashing techniques used to determine the similarity of two malicious programs (Mandiant, 2014). Unlike other hashing techniques, which generate hash of a complete file, this import hashing technique only generates a hash of a small part (i.e., Imports or Import Libraries) of a PE file (see Figure 2). Imports/Import Libraries are simply functions called by a program (here, malware) from other programs, which are to be bound and linked with the program to build the final executable program (Mandiant, 2014). The details of Imports/Import Libraries (i.e., DLLs) are maintained in the Import Address Table (IAT), which is the basis of the generation of IMPort HASH (IMPHASH) of a program, where the order in which the Import Libraries are called determines the value of the generated IMPHASH. Thus, the two programs contain the same code but their order of functions/Import Libraries are different then they will produce distinct IMPHASH values. Import hashing is similar to fuzzy hashing except it provides binary outcomes (samples are matched or not), but not the degree of similarity scores.

2.2.2. peHash

It is another PE file-based hashing technique that calculates hash values of various parts of the PE format. It considers several PE properties into account while calculating hash values such as image characteristics, subsystem, stack commit size, heap commit size, virtual address, raw size and section characteristics (Wicherski, 2009). Hash values are calculated for binaries in the PE format that transform structural information about a sample into a hash value. Grouping binaries by hash values calculated with the new function allows for detection of multiple instances of the same polymorphic sample in addition to samples that are broken (Wicherski, 2009). The peHash technique can accomplish correct clustering for large groups based on very

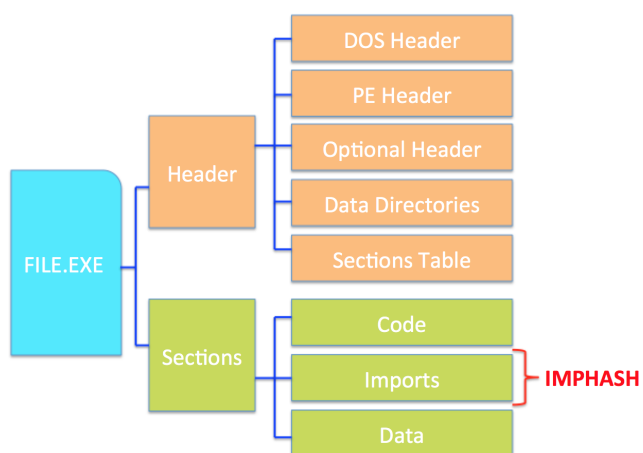


Figure 2: Import Hash (IMPHASH) is generated based on the Import Address Table (IAT) in a Portable Executable (PE) File

basic information from the PE files without any assumptions about the contents of the various sections, but only relying on their abundance of information. However, peHash cannot be used to cluster variants of malware families, as this would require the code structure itself to be analysed which due to the usage of packers is totally invisible to peHash (Wicherski, 2009).

2.3. YARA Rules

YARA rules are the most popular and de facto technique of malware analysis at present. YARA rules are utilised to discover malware, where each YARA rule contains predetermined strings which are matched against the strings of the targeted samples or processes (Naik, Jenkins, Savage, Yang, Naik and Song, 2020e; VirusTotal, 2019). The strings which are included in YARA rules are derived from the reverse engineering of known malware (Naik, Jenkins, Savage, Yang, Naik and Song, 2019e). These strings of the YARA rules are divided into three categories: text strings, hexadecimal strings and regular expression strings (see Figure 3). Text strings are generally a readable text complemented with some modifiers (e.g. nocase, ASCII, wide, and fullword) to manage the process more effectively (Alvarez, 2019). Hexadecimal strings are a sequence of raw bytes complemented with three flexible formats: wild-cards, jumps, and alternatives (Alvarez, 2019). Regular expression strings are similar to text strings as a readable text complemented with some modifiers; which are available since version 2.0 and increases the capability of YARA rules (Alvarez, 2019). Text strings and regular expression strings can be used to express a sequence of raw bytes through the use of escape sequences. Another crucial segment of a YARA rule is a rule condition that states the criteria for generating an alert by that rule, i.e. a number of strings must be matched with the target sample to alert that sample as malware (Readthedocs, 2019). YARA conditions determine whether to trigger the rule or not, nonetheless, rule conditions are Boolean expressions which are similar to those used in all other programming languages

```

rule RuleName
{
  meta:
    description = "descriptions of rule"
    author = "name"
    date = "dd/mm/yyyy"
    reference = "url"

  strings:
    $text_string1 = "text1 you wish to find in malware"
    $text_string2 = "text2 you wish to find in malware"

    $hex_string1 = {hex1 you wish to find in malware}
    $hex_string2 = {hex2 you wish to find in malware}

    $reg_exp_string1 = /regular expressions1 you wish to find in malware/
    $reg_exp_string2 = /regular expressions2 you wish to find in malware/

  condition:
    $text_string1 or $text_string2 or
    $hex_string1 or $hex_string2 or
    $reg_exp_string1 or $reg_exp_string2
}

rule WannaCry
{
  meta:
    description = "Generic Signature of WannaCry"
    author = "Nitin Naik"
    date = "01/06/2018"
    reference = "www.mydomain.com"

  strings:
    $text_string1 = "encrypt"
    $text_string2 = "bitcoin"

    $hex_string1 = {B6 D3 56 A5 78 43}
    $hex_string2 = {E8 27 F9 83 C4 82}

    $reg_exp_string1 = /md5: [0-9a-fA-F]{32}/
    $reg_exp_string2 = /state: (on|off)/

  condition:
    $text_string1 or $text_string2 or
    $hex_string1 or $hex_string2 or
    $reg_exp_string1 or $reg_exp_string2
}
  
```

Figure 3: YARA Rules: Syntax and Example

(Alvarez, 2019; Naik, Jenkins, Savage, Yang, Boongoen, Iam-On, Naik and Song, 2020d).

3. Related Work: Recent Malware Analysis Techniques Applied to Ransomware

Malware analysis is an established research area in cybersecurity and a number of malware detection techniques have been proposed in the past, however, only a small number of detection techniques were developed for ransomware. Here, some static and dynamic analysis methods which have been recently proposed to detect ransomware are discussed. The dynamic analysis method UNVEIL checks any tampering activity related to files and data (Kharraz, Arshad, Mulliner, Robertson and Kirda, 2016). Similarly, a dynamic machine learning method EldeRan monitors a set of actions performed by applications as signs of ransomware (Sgandurra, Muñoz-González, Mohsen and Lupu, 2016). Another dynamic analysis method HelDroid is particularly designed for mobile devices that detects if an app is attempting to lock or encrypt the device without the user’s consent (Andronio, Zanero and Maggi, 2015). The dynamic analysis method R-Locker is developed for detecting and preventing ransomware by employing honeypiles (a FIFO file structure) to block ransomware once it starts reading the file (Gómez-Hernández, Álvarez-González and García-Teodoro, 2018).

These dynamic methods are more effective than static methods in detection, however, their disadvantage in detecting ransomware is that they require the successful loading and running of ransomware to demonstrate their expected behaviour or characteristics. If the infected ransomware is able to evade the method due to its silent behaviour for some time or waiting for some specific activity from users, then it is already too late for those affected users. Moreover, depending on the requirement of emulation or virtualization for the dynamic method, several efficient ransomware variants will not perform any action and thus cannot be detected by that method. Furthermore, if the detection policy of the dynamic method is matched with the activity of goodware then that

dynamic method may generate several false positives.

Alternatively, a static analysis method can safely examine the ransomware samples without running them and affecting the data and user, leading to a safer analysis of malware. However, most static analysis and detection methods (Richardson and North, 2017; Cabaj, Gawkowski, Grochowski and Osojca, 2015; Chen and Bridges, 2017) require greater computational overheads and suffer from false positives and negatives. This proposed ransomware detection method is a static analysis method, which requires fewer computational overheads and performs rapid comparative analysis, in comparison to other static analysis methods.

4. Data Collection: Collection of Malware (Ransomware) and Goodware Samples

The Malware collection process is the principal part of malware analysis as it determines the significance and success of any malware analysis. There are several important aspects of the malware collection process such as: collecting the relevant and diverse malware samples; and assessing the validity and correctness of malware samples (Plohmann, Clauss, Enders and Padilla, 2018; Breitingner et al., 2013b; Upchurch and Zhou, 2015). Therefore, in this research study, following the analysis of various types of malware, one of the most significant and predominant malware, ransomware was selected for the examination and evaluation of the performance of the proposed fuzzy-import hashing method. Ransomware is a major concern for ordinary users and business organisations, as it exploits victims for financial gain, disrupts business activities and encrypts sensitive data (Constantin, 2020). Selecting the most relevant categories of ransomware was an important task as some ransomware categories were more severe in terms of attacks and financial loss, and these should be considered for analysis purpose, therefore, four ransomware categories were selected for this research: WannaCry, Locky, Cerber and CryptoWall (Richardson and North, 2017; Spadafora, 2020; Raconteur.net, 2017; Savage, Coogan and Lau, 2015; Klijnsma, 2019; Malwarebytes, 2019).

Malware samples were collected from the two well known repositories *Hybrid Analysis* (Hybrid-Analysis, 2019) and *Malshare* (Malshare, 2019). Thereupon these samples were verified for their credibility as several samples were simply counterfeit samples. It was critical to select only credible samples of each category as a reference to test all selected malware analysis techniques and the proposed fuzzy-import hashing technique successfully. These samples were investigated based on the information available on *VirusTotal* (VirusTotal, 2019). To determine that every sample was indeed genuine malware or ransomware and a member of a specific ransomware category, the criteria was set that it must be identified as malware by at least 40 or more detection engines on *VirusTotal*. To confirm the ransomware category of collected samples, their category from WannaCry, Locky, Cerber and CryptoWall was verified manually on the recognized detection engines on *VirusTotal*. This whole process of

malware collection and verification was a prolonged process, as a result, 1000 ransomware samples were selected for experiments from the earlier collected several thousand samples, then these samples were evenly divided into 250 samples of four ransomware categories WannaCry, Locky, Cerber and CryptoWall. The four distinct categories of ransomware were selected to examine how each selected method works on the distinct categories of ransomware. This malware testing and verification process can employ some of the existing frameworks to expedite the overall operation such as Malpedia (Plohmann et al., 2018), FRASH (Breitingner et al., 2013b) and Variant (Upchurch and Zhou, 2015).

In addition to the collection of malware (ransomware) samples, an equal number of goodware samples were collected to balance this analysis. These 1000 goodware samples were the files collected from ten commonly used software: JAVA, MS OFFICE, Google Chrome, MySQL, R, NMAP, McAfee, MATLAB, Python and Snort. These 10 software were chosen in such a way that it could encompass a wide range of benign programs and to evaluate how each malware analysis method works on the different types of benign programs. Finally, a total 2000 samples were utilised to perform all the experiments applying different malware analysis methods.

5. Experimental Evaluation of Selected Techniques: Malware Analysis Using Fuzzy Hashing and Import Hashing

Originally, the selected malware analysis techniques fuzzy hashing and import hashing are utilised to perform the malware analysis on the collected and verified samples of four ransomware categories of WannaCry, Locky, Cerber and CryptoWall and goodware samples. Fuzzy hashing and import hashing are compact, fast and resource-optimised malware analysis methods (Naik et al., 2019c). In addition to the accuracy of malware analysis results, these criteria are decisive in determining the appropriate method for the analysis of a large volume of malware. This section explains the malware detection process and its outcome of fuzzy hashing and import hashing for the collected ransomware and goodware samples. The experiment is intended at finding the similarity detection rate of each analysis method for each ransomware category separately and collectively. It is expected that most probably each sample of the same category holds some similarity to other samples in that category. Therefore, experiments evaluate how many samples within one category are matched with at least one other sample of the same category by each analysis method.

5.1. Malware Detection Process: Fuzzy Hashing

While detecting malware, fuzzy hashing is applied on an unpacked ransomware sample, where it generates a fuzzy hash value for that ransomware sample. The generated fuzzy hash value is compared with either previously known ransomware samples or the database of fuzzy hash values of ransomware. If the fuzzy hash of an examined sample matches

with any of the previously known ransomware samples or fuzzy hash in the database then the degree of similarity of the two matched samples is produced as a fuzzy hash result. The fuzzy similarity or degree of similarity is represented in the range of 1% (least matched) to 100% (exactly matched), nonetheless, their interpretation is dependent on malware analysts based on their criteria and requirements. Malware analysts normally set a threshold value to accept or ignore the fuzzy similarity score and to determine as matched or not matched scenarios respectively. This method can be utilised as a preliminary method in malware analysis that may help in any advanced analysis (Naik, Jenkins, Savage and Yang, 2019d).

5.2. Malware Detection Results: Fuzzy Hashing

The three previously discussed fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B were employed to detect the similarity for each ransomware category individually. The comparison of their detection rates was based on the different similarity thresholds to check their effectiveness and consistent performance in all the set threshold conditions. Four similarity threshold conditions were set for their evaluation: 1) when all the fuzzy similarity scores were considered (1-100%), 2) when those fuzzy similarity scores were considered which are greater than 10%, 3) when those fuzzy similarity scores were considered which are greater than 20%, and 4) when those fuzzy similarity scores were considered which are greater than 30%. The detection results of all the three fuzzy hashing methods in four different threshold conditions are presented in Table 1. The evaluation of these three fuzzy hashing methods revealed that the detection result of SDHASH and mvHASH-B was gradually declined when the similarity threshold value was gradually increased, and in certain cases it was declined considerably. The detection rate of the SSDEEP fuzzy hashing method is relatively lower, though, consistent in all four experiments. For the highest similarity threshold limit of 30%, most SSDEEP results exceed the other two fuzzy hashing methods (Naik et al., 2020f). This outcome is critical when utilising such similarity detection results in developing the proposed method and advanced analysis (e.g., clustering or classification) (Naik, Jenkins, Gillett, Mouratidis, Naik and Song, 2019a), as it will determine their success rate.

5.3. Malware Detection Process: Import Hashing

While detecting malware, import hashing is applied on an unpacked ransomware sample, where it generates an IMPHASH value for that ransomware sample. The generated IMPHASH value is compared with either previously known ransomware samples or the database of IMPHASH values of ransomware. If the IMPHASH of an examined sample matches with any of the previously known ransomware samples or IMPHASH in the database then the import hashing result is produced as matched. Nonetheless, it only provides binary outcomes (samples are matched or not), but not the degree of similarity scores. This method can be utilised as a preliminary method in malware analysis that may assist in any advanced analysis (Naik et al., 2019c).

5.4. Malware Detection Results: Import Hashing

The import hashing method was employed to detect similarity for each ransomware category individually, and their results are shown in Table 2. For some ransomware categories, the detection result of import hashing was relatively better than fuzzy hashing, however, in some categories it was not. Additionally, import hashing generates IMPHASH value of IAT of PE file format, therefore, their effectiveness is also dependent on the type of file format of samples investigated.

After employing both fuzzy hashing and import hashing individually on the ransomware samples, their detection rates were analysed, and it indicated that the detection rates of three fuzzy hashing (SSDEEP, SDHASH and mvHASH-B) and import hashing methods were not satisfactory. Consequently, they are not effective malware analysis technique as a standalone method and require additional improvement. Nonetheless, both methods are a fast and compact method and they can be combined together as an integrated method to apply two different detection mechanisms for improving the overall detection rate without affecting the performance considerably.

6. Proposed Technique: Malware Analysis Using the Proposed Fuzzy-Import Hashing

6.1. Malware Detection Process: Fuzzy-Import Hashing

No matter what malware analysis method is chosen to perform the analysis, it cannot always guarantee the best outcome for all malware types in all circumstances. Each method has their strengths and weaknesses, therefore, sometimes it may be effective to utilise an integrated method by combining the methods, to improve the overall detection rate given that the integration does not affect the overall performance considerably. The proposed fuzzy-import hashing is the integration of fuzzy hashing and import hashing which applies both techniques to detect the similarity between two files (Naik, Jenkins, Savage, Yang, Boongoen and Iam-On, 2020c). Both fuzzy hashing and import hashing are a compact, fast and resource-optimised method employed for analysis which may not be effective by their own, nonetheless, they can complement each other and may improve the overall detection accuracy without affecting the overall performance considerably (Naik et al., 2019c). Fuzzy hashing attempts to find structural similarity between the two files in their entirety, whereas import hashing attempts to find similarity between import address tables of files. Therefore, they can complement each other in finding a missed opportunity by one of the methods (Naik et al., 2020c). Thus, the combined detection result can increase the detection accuracy and confidence level of the overall analysis process.

The output of fuzzy hashing and import hashing is slightly different, fuzzy hashing delivers the output in the form of degree of similarity of each matched sample and import hashing only reveals whether the sample is matched or not. Therefore, the important condition of their integration is to align both outputs in a way that can be represented as one integrated

Table 1
Similarity Detection Results of the SSDEEP, SDHASH and mvHASH-B Fuzzy Hashing Methods for WannaCry, Locky, Cerber and CryptoWall Ransomware Corpora

Fuzzy Hashing Matching Criteria	WannaCry Ransomware			Locky Ransomware			Cerber Ransomware			Cryptowall Ransomware		
	SSDEEP Detection Rate	SDHASH Detection Rate	mvHASH-B Detection Rate	SSDEEP Detection Rate	SDHASH Detection Rate	mvHASH-B Detection Rate	SSDEEP Detection Rate	SDHASH Detection Rate	mvHASH-B Detection Rate	SSDEEP Detection Rate	SDHASH Detection Rate	mvHASH-B Detection Rate
Fuzzy Similarity Scores (1-100%)	91.2%	93.6%	90%	42%	58.4%	72.4%	33.6%	71.2%	94.8%	28%	52.4%	83.6%
Fuzzy Similarity Scores >10%	91.2%	93.6%	90%	42%	38.4%	64%	33.6%	62.8%	90.4%	28%	32.8%	56.8%
Fuzzy Similarity Scores >20%	91.2%	90%	84.4%	41.6%	35.6%	36.4%	33.6%	37.6%	36.8%	28%	24%	20.8%
Fuzzy Similarity Scores >30%	90.8%	90%	84.4%	41.6%	30.4%	33.6%	33.6%	28.4%	36%	28%	20.4%	20.4%

Table 2
Similarity Detection Results of Import Hashing for WannaCry, Locky, Cerber and CryptoWall Ransomware Corpora

Ransomware Category	Import Hashing Detection Rate
WannaCry Ransomware	87.6%
Locky Ransomware	31.6%
Cerber Ransomware	61.6%
CryptoWall Ransomware	27.2%

output and successfully utilised in the advanced analysis (Naik et al., 2020c). In the proposed fuzzy-import hashing technique, for aligning the two hashing results, the binary outcomes of import hashing, matched and not matched are treated as greatest and lowest value of fuzzy hashing respectively. It means a matched result is measured as 1 or 100% of the fuzzy hashing result and a not matched result is measured as 0 of fuzzy hashing result. The rationale for assuming the matched result as 1 or 100% is that import hashing will only generate matched result if it finds the IATs of two files are same, meaning both files are same or almost similar (Naik et al., 2020c). Based on this assumption, the two hashing results can be aligned and the final results of fuzzy-import hashing can be produced as a degree of similarity in the range of 1 to 100%, which is the same as the fuzzy hashing result. The actual integrated operation is based on the fact that import hashing is relatively faster so it will be applied on samples first and if it could not find any match then fuzzy hashing is applied. This sequence of operations will avoid applying relatively slower fuzzy hashing on all samples and save computational overheads (Naik et al., 2020c). The logical approach for the implementation of fuzzy-import hashing is shown using the pseudocode in Algorithm 1.

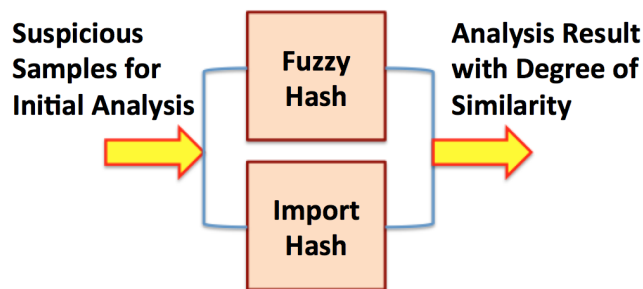


Figure 4: Fuzzy-Import Hashing: A Static Analysis Technique for Malware Detection

6.2. Malware Detection Results: Fuzzy-Import Hashing

The proposed fuzzy-import hashing method was employed to detect similarity for each ransomware category individually. Subsequently, its detection results were compared with the detection results of each of the fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B, and import hashing method. This comparative assessment was critical to confirm whether fuzzy-import hashing outperformed the other two underlying methods or not. If fuzzy-import hashing performed better than other two methods, then which fuzzy hashing method produced the best results among three chosen fuzzy hashing methods. Fuzzy-import hashing detection results based on the three underlying fuzzy hashing methods are shown in Table 3 (Naik et al., 2020f). All results are based on the fuzzy similarity scores greater than 30% for all fuzzy hashing methods. These results show fuzzy-import hashing improved the detection results for all three fuzzy hashing methods (at least >11%), however, the fuzzy-import hashing utilising SSDEEP generated relatively better overall results than the fuzzy-import hashing utilising SDHASH and mvHASH-B (see Figure 5). These experiments demonstrated that the proposed fuzzy-import hashing moderately improved detec-

Algorithm 1: Pseudocode of Fuzzy-Import Hashing to determine Malware Similarity by combining Fuzzy Hash with Import Hash

```

S, Set of Samples for Investigation
I, Set of Import Hashes of Known Malware
F, Set of Fuzzy Hashes of Known Malware
S, Similarity Score
I, Import Hash Value
F, Fuzzy Hash Value
 $\delta_T$ , Fuzzy Hash Similarity Threshold
 $\Delta$ , Degree of Similarity
for (i = 1; i < |S|; i++) do
    for (j = 1; j < |I|; j++) do
        if  $I_{S_i} == I_j$  then
             $S_{i,j} = 1$ 
        if  $I_{S_i} \notin I$  then
            for (k = 1; k < |F|; k++) do
                if  $\Delta(F_{S_i}, F_k) \geq \delta_T$  then
                     $S_{i,k} = \Delta(F_{S_i}, F_k)$ 
return S [ ]
    
```

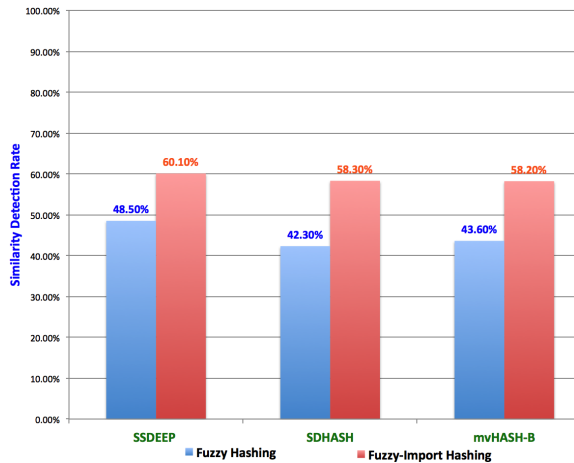


Figure 5: Overall Similarity Detection Rate of Fuzzy-Import Hashing and their related Fuzzy Hashing Methods SSDEEP, SDHASH, mvHASH-B for the collected Ransomware Corpora

tion results using all the three fuzzy hashing methods (Naik et al., 2020f).

7. Comparative Evaluation: Comparing the Proposed Fuzzy-Import Hashing Technique With Established YARA Rules

It is crucial to compare the similarity detection results and performance of the proposed fuzzy-import hashing with currently established and successful malware analysis technique. YARA rules technique is one of the most popular and

successful techniques used in malware analysis, therefore, the fuzzy-hashing result and performance is compared with YARA rules.

7.1. Malware Detection Process: YARA Rules

Fuzzy hashing and import hashing methods are similar in terms of generating a hash of a sample when they are applied. YARA rules are different from hashing as rule generation requires a reverse engineering process. Strings are extracted from malware to generate YARA rules after analysing particular malware and their family. Consequently, this rule generation process requires great effort and skill, unlike both hashing methods, where inexperienced personnel can perform the hash generation process to generate hashes of samples and accomplish the basic analysis. YARA rules can be generated manually or automatically, while the automatic rule generation process is relatively easy, nonetheless, it may necessitate some post-processing operations to improve these rules. Here *yarGen* tool (Roth, 2018) is employed to generate YARA rules for all ransomware samples, which is an effective tool to generate YARA rules (Naik, Jenkins, Cooke, Gillett and Jin, 2020a). This tool generates two types of rules ordinary rules and super rules depending on malware types by utilising some intelligent techniques such as Fuzzy Regular Expressions, Naive Bayes Classifier and Gibberish Detector (Roth, 2017). The generated basic YARA rules comprise up to 20 strings depending on their highest scores and do not comprise IMPHASH values, which is a separate method here (Naik et al., 2020a).

7.2. Malware Detection Results: YARA Rules

After generating YARA rules for all four ransomware categories separately, they are used to detect the similarity for each ransomware category individually. The detection results of YARA rules for all four ransomware categories are shown in Table 4. These detection results are mixed results in comparison with fuzzy and import hashing results; for two categories, it is somewhat improved, but for the other two categories, it is not. Nonetheless, there is a caveat here as these basic YARA rules were generated by *yarGen* with its default settings, it means different YARA tools may generate different rules which might produce different results. Additionally, if the number of strings in YARA rules are increased or decreased then it may affect the analysis results. If the number of strings are increased considerably then it may affect the performance of YARA rules adversely as it is expected to perform malware analysis on a large sample size.

7.3. Fuzzy-Import Hashing vs. YARA Rules: Detection Rate and Performance

The overall similarity detection results of different fuzzy-import hashing methods (SSDEEP, SDHASH and mvHASH-B) are presented and compared with YARA rules in Figure 6, where fuzzy-import hashing similarity detection results could not exceed the results of YARA rules, however, it is competitive and in particular, SSDEEP based fuzzy-import hashing result is very close to YARA rules. Noticeably, the

Table 3

Comparison of Similarity Detection Results of Fuzzy-Import Hashing with SSDEEP, SDHASH, mvHASH-B, IMPHASH for the collected Ransomware Corpora

Ransomware Category	Import Hashing (IMPHASH) Detection Rate	Fuzzy Hashing (SSDEEP) Detection Rate	Fuzzy-Import Hashing (SSDEEP) Detection Rate	Fuzzy Hashing (SDHASH) Detection Rate	Fuzzy-Import Hashing (SDHASH) Detection Rate	Fuzzy Hashing (mvHASH-B) Detection Rate	Fuzzy-Import Hashing (mvHASH-B) Detection Rate
WannaCry Ransomware	87.6%	90.8%	92.8%	90%	92.8%	84.4%	92.8%
Locky Ransomware	31.6%	41.6%	48.8%	30.4%	45.2%	33.6%	44.4%
Cerber Ransomware	61.6%	33.6%	61.6%	28.4%	61.6%	36%	61.6%
CryptoWall Ransomware	27.2%	28%	37.2%	20.4%	33.6%	20.4%	34%
Overall Detection Rate of Each Hashing Method	52%	48.5%	60.1%	42.3%	58.3%	43.6%	58.2%

Table 4

Similarity Detection Results of YARA Rules for WannaCry, Locky, Cerber and CryptoWall Ransomware Corpora

Ransomware Category	YARA Rules* Detection Rate
WannaCry Ransomware	89.6%
Locky Ransomware	54.4%
Cerber Ransomware	77.2%
CryptoWall Ransomware	27.6%
Overall Detection Rate of YARA Rules	62.20%

YARA Rules*: These rules are generated by **yarGen** tool utilising machine learning methods Fuzzy Regular Expressions, Naive Bayes Classifier and Gibberish Detector, where simple rules contain up to the 20 highest scored strings.

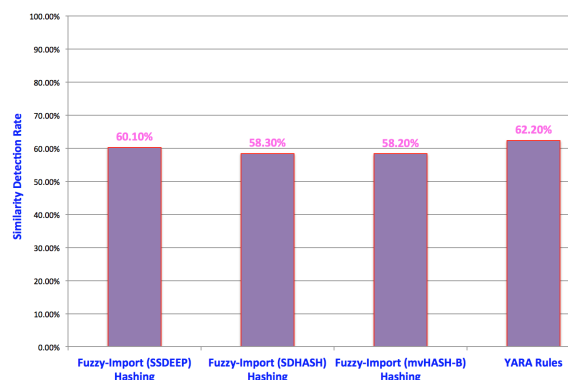


Figure 6: Comparison of Similarity Detection Rate of Fuzzy-Import Hashing (based on SSDEEP, SDHASH, mvHASH-B Fuzzy Hashing methods) and YARA Rules for the collected Ransomware Corpora

overall performance (including speed, memory and other resources) of fuzzy-import hashing is better than YARA rules as shown in Table 6. The performance of a method is crucial in malware analysis as it is always used for a large sample size. Therefore, a balance of detection accuracy and performance is very important when selecting any malware analysis approach. If the performance is a major criteria then fuzzy-import hashing may be more suitable option here.

For further rigorous comparison of fuzzy-import hashing and YARA rules, some other evaluation metrics (Accuracy, Precision, Recall and F1-Score) were calculated as shown in Table 5. Here, the two results fuzzy-import hashing (SSDEEP) and YARA rules are very close and comparable. In particular, the accuracy of fuzzy-import hashing (SSDEEP) is marginally higher than YARA rules (see Accuracy Values in Table 5). YARA rules generated a very small number of false positives, while fuzzy-import hashing (SSDEEP) generated zero false positive (see Precision Values in Table 5). However, YARA rules generated fewer false negatives in comparison to the fuzzy-import hashing (SSDEEP) method (see Recall Values in Table 5). In evaluating the efficacy of a

method, a balance of Precision and Recall is vital, therefore, the F1-Score may be more helpful in determining a suitable analysis method. Here, the F1-Score of fuzzy-import hashing (SSDEEP) is 75.08% and YARA rules is 75.49%, which is again very similar and shows fuzzy-import hashing can produce equally good results as YARA rules.

8. Application in Advanced Analysis: Utilising the Proposed Fuzzy-Import Hashing in Fuzzy C-Means Clustering

The proposed fuzzy-import hashing technique provides the useful input for advanced analysis such as clustering. Its results can be utilised in clustering, and particularly in fuzzy c-means clustering (Bezdek, Ehrlich and Full, 1984; Krishnapuram and Keller, 1993; Pal, Pal, Keller and Bezdek, 2005; Yang and Wu, 2006), due to its fuzzy similarity scores. Here, the results of fuzzy-import hashing based on three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B were utilised for FCM to compare if fuzzy-import hashing could improve the clustering results. Evaluation of the clustering

Table 5

Comparison of Evaluation Metrics for Fuzzy-Import Hashing with SSDEEP, SDHASH, mvHASH-B, IMPHASH and YARA Rules

Evaluation Metric	Import Hashing (IMPHASH)	Fuzzy Hashing (SSDEEP)	Fuzzy-Import Hashing (SSDEEP)	Fuzzy Hashing (SDHASH)	Fuzzy-Import Hashing (SDHASH)	Fuzzy Hashing (mvHASH-B)	Fuzzy-Import Hashing (mvHASH-B)	YARA Rules
Accuracy	76.00%	74.25%	80.05%	71.15%	79.15%	71.80%	79.10%	79.80%
Precision	100%	100%	100%	90.19%	92.69%	90.08%	92.38%	95.99%
Recall	52.00%	48.50%	60.10%	42.30%	58.3%	43.60%	58.20%	62.20%
F1-Score	68.42%	65.32%	75.08%	57.59%	71.58%	58.76%	71.41%	75.49%

Table 6

Comparison of Performance Metrics for IMPHASH, SSDEEP, SDHASH and YARA Rules

Performance Metric	IMPHASH Import Hashing	SSDEEP Fuzzy Hashing	SDHASH Fuzzy Hashing	mvHASH-B Fuzzy Hashing	YARA Rules
Similarity Measure	Similarity in import libraries and their order in files	Similarity in structure and order of code of files	Similarity in structure and order of code of files	Similarity in structure and order of code of files	Similarity in strings and op-codes of files
Speed	Fastest among all	Slower than Import Hashing	Slower than SSDEEP	Slower than SDHASH	Mostly slower among all, however, it is determined by the nature of execution (direct or compiled)
Memory	Least among all	Slightly higher than Import Hashing	Slightly higher than SSDEEP	Slightly higher than SDHASH	Mostly higher among all, however, it is determined by the size and number of rules
Size of File	None	≥ 4 KB	≥ 512 Bytes	≥ 4 KB	None

process was undertaken using the fuzzy c-means clustering results and their four fuzzy indexes *Fuzzy Silhouette Index*, *Partition Coefficient*, *Modified Partition Coefficient* and *Partition Entropy*, which were computed and collectively compared for both fuzzy-import hashing and its corresponding fuzzy hashing for the optimal clustering value of that category. Here, the higher value of the first three evaluation metrics signifies better clustering results and the lower value of the fourth evaluation metric signifies better clustering results. This computation was based on *fclust* package of **R** (Giordani and Ferraro, 2015). The comparative results are shown in Tables 7 to 10 for four different ransomware categories: WannaCry, Locky, Cerber and CryptoWall respectively (Naik et al., 2020c). The majority of fuzzy-import based FCM results (8 out of 12) were improved, nonetheless, a few FCM results were not, reflecting the necessity of additional analysis of unimproved results and their potential reasons (Naik et al., 2020c).

9. Advantages and Limitations of the Proposed Technique

9.1. Advantages of the Proposed Fuzzy-Import Hashing

The proposed fuzzy-import hashing offers several advantages, here some of the most notable advantages are (Naik et al., 2020c):

- **Performance Sustainability:** Import hashing is one of the fastest analysis methods as it only generates the

hash of a part of a file (i.e., IAT) and does not affect the overall performance of the combined analysis process.

- **Detection Rate Improvement:** Both hashing methods can sometimes complement each other when one hashing method fails to find similarity due to its particular limitations. Therefore, fuzzy-import hashing can detect more malware samples than any single method alone.
- **Overheads Minimisation:** Import hashing is faster than fuzzy hashing, therefore, if it is applied on samples prior to fuzzy hashing then all the matched samples would not require re-processing through fuzzy hashing as only unmatched samples require checking by fuzzy hashing. This avoids performing fuzzy hashing on all the samples thus reducing overheads required by fuzzy hashing alone.
- **Result Alignment:** Import hashing results (binary outcomes) can be readily aligned with fuzzy hashing results where the matching result could be treated similarly to 1 or 100% of a fuzzy hashing result (exact match of fuzzy hashing) and an unmatched result could be treated similarly to 0 of a fuzzy hashing result (no match of fuzzy hashing). Therefore, the two results can be easily aligned together in the form of fuzzy similarity scores.
- **Accuracy Improvement:** In case of import hashing found matched sample(s), the strong similarity score 1 or 100% is added to the final similarity result of

Table 7

Comparison of FCM Results based on Similarity Scores of Fuzzy-Import Hashing and their corresponding Fuzzy Hashing for the collected WannaCry Ransomware Corpora

Cluster Validity Index	Fuzzy Hashing (SSDEEP)	Fuzzy-Import Hashing (SSDEEP)	Fuzzy Hashing (SDHASH)	Fuzzy-Import Hashing (SDHASH)	Fuzzy Hashing (mvHASH-B)	Fuzzy-Import Hashing (mvHASH-B)
Fuzzy Silhouette Index	0.78324	0.793863	0.6958656	0.8228494	0.7994093	0.826524
Partition Coefficient	0.6433042	0.6691064	0.7055717	0.8488053	0.4005461	0.7275929
Modified Partition Coefficient	0.5719651	0.6029277	0.646686	0.8185663	0.2806553	0.6731114
Partition Entropy	0.8016553	0.7308573	0.5792176	0.3254954	1.278439	0.6301614

Table 8

Comparison of FCM Results based on Similarity Scores of Fuzzy-Import Hashing and their corresponding Fuzzy Hashing for the collected Locky Ransomware Corpora

Cluster Validity Index	Fuzzy Hashing (SSDEEP)	Fuzzy-Import Hashing (SSDEEP)	Fuzzy Hashing (SDHASH)	Fuzzy-Import Hashing (SDHASH)	Fuzzy Hashing (mvHASH-B)	Fuzzy-Import Hashing (mvHASH-B)
Fuzzy Silhouette Index	0.9085124	0.9300699	0.8325851	0.8326483	0.8816986	0.8443886
Partition Coefficient	0.8376619	0.838944	0.7522258	0.7536781	0.9988531	0.9408663
Modified Partition Coefficient	0.8051943	0.8053328	0.702671	0.7044138	0.9986237	0.9290395
Partition Entropy	0.3675082	0.347518	0.5703733	0.5602319	0.005407051	0.1397954

fuzzy-import hashing, which increases the accuracy of the overall result and the further processing results of clustering or classification.

9.2. Limitations of the Proposed Fuzzy-Import Hashing

Despite offering several advantages, the proposed fuzzy-import hashing has some limitations, here some of the most notable limitations are:

- **Similarity Scores:** Similarity scores provided by fuzzy-import hashing could be analysed differently by differ-

ent security analysts, resulting different conclusions based on the same similarity scores.

- **Structural Similarity:** Fuzzy-import hashing could only discover structural or syntactic similarity, however, not behavioural or semantic similarity due to the limitations of both underlying techniques.
- **File/Function Size and Order:** Fuzzy-import hashing similarity scores are dependent on the size of files, blocks and order of functions, therefore, any change in any of these parameters may affect the similarity score.

Table 9

Comparison of FCM Results based on Similarity Scores of Fuzzy-Import Hashing and their corresponding Fuzzy Hashing for the collected Cerber Ransomware Corpora

Cluster Validity Index	Fuzzy Hashing (SSDEEP)	Fuzzy-Import Hashing (SSDEEP)	Fuzzy Hashing (SDHASH)	Fuzzy-Import Hashing (SDHASH)	Fuzzy Hashing (mvHASH-B)	Fuzzy-Import Hashing (mvHASH-B)
Fuzzy Silhouette Index	0.8559951	0.6945895	0.6917668	0.6937094	0.7052969	0.7406375
Partition Coefficient	0.7772775	0.6930008	0.7838876	0.7951774	0.6131531	0.656417
Modified Partition Coefficient	0.732733	0.631601	0.7406651	0.7542129	0.5357837	0.5877004
Partition Entropy	0.4904145	0.6716732	0.4853877	0.4553734	0.8616182	0.7794824

Table 10

Comparison of FCM Results based on Similarity Scores of Fuzzy-Import Hashing and their corresponding Fuzzy Hashing for the collected CryptoWall Ransomware Corpora

Cluster Validity Index	Fuzzy Hashing (SSDEEP)	Fuzzy-Import Hashing (SSDEEP)	Fuzzy Hashing (SDHASH)	Fuzzy-Import Hashing (SDHASH)	Fuzzy Hashing (mvHASH-B)	Fuzzy-Import Hashing (mvHASH-B)
Fuzzy Silhouette Index	0.9863146	0.4704871	0.9988991	0.4587643	0.7775503	0.7939686
Partition Coefficient	0.7826071	0.6088555	0.9091084	0.1666667	0.5108229	0.822717
Modified Partition Coefficient	0.7391285	0.5306266	0.89093	0.5232862	0.4129875	0.7872604
Partition Entropy	0.4428341	0.8592991	0.1518233	1.791759	1.058915	0.4041951

- **Packed Samples:** Fuzzy-import hashing may not be very effective on packed samples and unable to discover their similarity due to the limitations of both underlying techniques.

10. Conclusion

This paper proposed a static analysis technique called a fuzzy-import hashing for malware detection to improve the overall detection rate and performance. The fuzzy-import hashing was the integration of fuzzy hashing and import hashing, which are fast methods but not always efficient in producing acceptable results as a standalone method. These two hashing techniques were integrated in such a manner that the performance of the proposed technique was optimised, in addition to improving the detection accuracy. In achieving this efficiency, faster import hashing was applied on samples before fuzzy hashing which avoided the use of fuzzy hashing on the complete corpus and only applied on the remainder of undetected samples, thus saved the computational overheads of fuzzy hashing.

The similarity detection results of the proposed fuzzy-import hashing was compared against individual fuzzy hashing methods (SSDEEP, SDHASH and mvHASH-B) and import hashing method, which demonstrated an improvement in similarity detection rates (>11%) for each fuzzy hashing method. The detection results and performance of the proposed technique was compared against the established malware analysis technique YARA rules, which demonstrated that this proposed technique could produce equally respectable results to that of YARA rules. Later, the FCM clustering result based on fuzzy-import hashing was compared against the related fuzzy hashing methods (SS-DEEP, SDHASH and mvHASH-B) to determine its success for advanced clustering analysis, demonstrating some positive results. Nonetheless, further investigation is necessary to analyse the clustering results. This proposed fuzzy-import hashing technique produced some enhancements in overall detection rates and performance. If the performance is a major criteria then fuzzy-import hashing may be more suitable option for malware analysis. However, the detection rate of this technique is still requires improvements for its development as a standard static analysis technique for malware detection.

Acknowledgement

The authors gratefully acknowledge the support of *Hybrid-Analysis.com*, *Malshare.com* and *VirusTotal.com* for this research work.

References

Alvarez, V., 2019. Writing YARA rules. URL: <https://yara.readthedocs.io/en/v3.4.0/writingrules.html>.
 Andronio, N., Zanero, S., Maggi, F., 2015. Heldroid: Dissecting and detecting mobile ransomware, in: International Workshop on Recent Advances in Intrusion Detection, Springer. pp. 382–404.

Bezdek, J.C., Ehrlich, R., Full, W., 1984. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences* 10, 191–203.
 Breiting, F., Astebøl, K.P., Baier, H., Busch, C., 2013a. mvhash-b- a new approach for similarity preserving hashing, in: 2013 Seventh International Conference on IT Security Incident Management and IT Forensics, IEEE. pp. 33–44.
 Breiting, F., Baier, H., 2012. A fuzzy hashing approach based on random sequences and hamming distance, in: Annual ADFSL Conference on Digital Forensics, Security and Law. 15. URL: <https://commons.erau.edu/adfsl/2012/wednesday/15>.
 Breiting, F., Stivaktakis, G., Baier, H., 2013b. FRASH: A framework to test algorithms of similarity hashing. *Digital Investigation* 10, S50–S58.
 Cabaj, K., Gawkowski, P., Grochowski, K., Osojca, D., 2015. Network activity analysis of Cryptowall ransomware. *Przeład Elektrotechniczny* 91, 201–204.
 Chen, Q., Bridges, R.A., 2017. Automated behavioral analysis of malware a case study of wannacry ransomware. arXiv preprint arXiv:1709.08753.
 Constantin, L., 2020. More targeted, sophisticated and costly: Why ransomware might be your biggest threat. URL: <https://www.csoonline.com/article/3518864/more-targeted-sophisticated-and-costly-why-ransomware-might/-be-your-biggest-threat.html>.
 Gayoso Martínez, V., Hernández Álvarez, F., Hernández Encinas, L., 2014. State of the art in similarity preserving hashing functions. URL: http://digital.csic.es/bitstream/10261/135120/1/Similarity_preserving_Hashing_functions.pdf.
 Giordani, P., Ferraro, M.B., 2015. Package FCLUST: Fuzzy Clustering. CRAN R studio.
 Gómez-Hernández, J., Álvarez-González, L., García-Teodoro, P., 2018. R-locker: Thwarting ransomware action through a honeyfile-based approach. *Computers & Security* 73, 389–398.
 Harrell, C., 2013. Finding Malware: Like Iron Man. URL: https://digital-forensics.sans.org/summit-archives/DFIR_Summit/Finding-Malware-Like-Iron-Man-Corey-Harrell.pdf.
 Hybrid-Analysis, 2019. Hybrid Analysis. URL: <https://www.hybrid-analysis.com/>.
 Kharraz, A., Arshad, S., Mulliner, C., Robertson, W.K., Kirda, E., 2016. Unveil: A large-scale, automated approach to detecting ransomware., in: USENIX Security Symposium, pp. 757–772.
 Klijnsma, Y., 2019. The history of Cryptowall: a large scale cryptographic ransomware threat. URL: <https://www.cryptowalltracker.org/>.
 Kornblum, J., 2006. Identifying almost identical files using context triggered piecewise hashing. *Digital investigation* 3, 91–97.
 Krishnapuram, R., Keller, J.M., 1993. A possibilistic approach to clustering. *IEEE transactions on fuzzy systems* 1, 98–110.
 Malshare, 2019. A free Malware repository providing researchers access to samples, malicious feeds, and YARA results. URL: <https://malshare.com/index.php>.
 Malwarebytes, 2019. Ransomware. URL: <https://www.malwarebytes.com/ransomware/>.
 Mandiant, 2014. Tracking malware with import hashing. URL: <https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html>.
 Naik, N., Jenkins, P., Cooke, R., Gillett, J., Jin, Y., 2020a. Evaluating automatically generated YARA rules and enhancing their effectiveness, in: IEEE Symposium Series on Computational Intelligence (SSCI), IEEE.
 Naik, N., Jenkins, P., Gillett, J., Mouratidis, H., Naik, K., Song, J., 2019a. Lockout-Tagout Ransomware: A detection method for ransomware using fuzzy hashing and clustering, in: IEEE Symposium Series on Computational Intelligence (SSCI), IEEE.
 Naik, N., Jenkins, P., Savage, N., 2019b. A ransomware detection method using fuzzy hashing for mitigating the risk of occlusion of information systems, in: 2019 IEEE International Symposium on Systems Engineering (ISSE), IEEE.
 Naik, N., Jenkins, P., Savage, N., Yang, L., 2019c. Cyberthreat Hunting- Part 1: Triaging Ransomware using Fuzzy Hashing, Import Hashing

- and YARA Rules, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE.
- Naik, N., Jenkins, P., Savage, N., Yang, L., 2019d. Cyberthreat Hunting-Part 2: Tracking Ransomware Threat Actors using Fuzzy Hashing and Fuzzy C-Means Clustering, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE.
- Naik, N., Jenkins, P., Savage, N., Yang, L., 2020b. A computational intelligence enabled honeypot for chasing ghosts in the wires. *Complex & Intelligent Systems* .
- Naik, N., Jenkins, P., Savage, N., Yang, L., Boongoen, T., Iam-On, N., 2020c. Fuzzy-Import Hashing: A malware analysis approach, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE.
- Naik, N., Jenkins, P., Savage, N., Yang, L., Boongoen, T., Iam-On, N., Naik, K., Song, J., 2020d. Embedded yara rules: strengthening YARA rules utilising fuzzy hashing and fuzzy rules for malware analysis. *Complex & Intelligent Systems* .
- Naik, N., Jenkins, P., Savage, N., Yang, L., Naik, K., Song, J., 2019e. Augmented YARA rules fused with fuzzy hashing in ransomware triaging, in: IEEE Symposium Series on Computational Intelligence (SSCI), IEEE.
- Naik, N., Jenkins, P., Savage, N., Yang, L., Naik, K., Song, J., 2020e. Embedding fuzzy rules with YARA rules for performance optimisation of malware analysis, in: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE.
- Naik, N., Jenkins, P., Savage, N., Yang, L., Naik, K., Song, J., Boongoen, T., Iam-On, N., 2020f. Fuzzy hashing aided enhanced YARA rules for malware triaging, in: IEEE Symposium Series on Computational Intelligence (SSCI), IEEE.
- Naik, N., Shang, C., Jenkins, P., Shen, Q., 2020g. Building a cognizant honeypot for detecting active fingerprinting attacks using dynamic fuzzy rule interpolation. *Expert Systems* , e12557.
- Naik, N., Shang, C., Jenkins, P., Shen, Q., 2020h. D-FRI-Honeypot: A secure sting operation for hacking the hackers using dynamic fuzzy rule interpolation. *IEEE Transactions on Emerging Topics in Computational Intelligence* .
- Oliver, J., Cheng, C., Chen, Y., 2013. TLSH—A locality sensitive hash, in: 2013 Fourth Cybercrime and Trustworthy Computing Workshop, IEEE. pp. 7–13.
- Pal, N.R., Pal, K., Keller, J.M., Bezdek, J.C., 2005. A possibilistic fuzzy c-means clustering algorithm. *IEEE transactions on fuzzy systems* 13, 517–530.
- Plohmann, D., Clauss, M., Enders, S., Padilla, E., 2018. Malpedia: A Collaborative Effort to Inventorize the Malware Landscape. *The Journal on Cybercrime & Digital Investigations* 3.
- Raconteur.net, 2017. Wannacry: the biggest ransomware attack in history. URL: <https://www.raconteur.net/infographics/wannacry-the-biggest-ransomware-attack-in-history>.
- Readthedocs, 2019. Writing YARA rules. URL: <https://yara.readthedocs.io/en/v3.5.0/writingrules.html>.
- Richardson, R., North, M., 2017. Ransomware: Evolution, mitigation and prevention. *International Management Review* 13, 10–21.
- Roth, F., 2017. How to post-process YARA rules generated by yarGen. URL: <https://medium.com/@cyb3rops/how-to-post-process-yara-rules-generated-by-yargen-121d29322282>.
- Roth, F., 2018. yarGen is a generator for YARA rules. URL: <https://github.com/Neo23x0/yarGen>.
- Roussev, V., 2010. Data fingerprinting with similarity digests, in: IFIP International Conference on Digital Forensics, Springer. pp. 207–226.
- Sadowski, C., Levin, G., 2007. Simhash: Hash-based similarity detection. URL: www.webrankinfo.com/dossiers/wp-content/uploads/simhash.pdf.
- Savage, K., Coogan, P., Lau, H., 2015. The evolution of ransomware - Symantec , 1–57.
- Scott, J., 2017. Detecting malware through static and dynamic techniques. URL: <https://webcache.googleusercontent.com/search?q=cache:Bo7aPh5t3h4J:https://technical.ntlsecurity.com/post/102efk4/detecting-malware-through-static-and-dynamic-techniques+źcd=13źhl=enźct=clnkźgl=uk>.
- Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C., 2016. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020 .
- Spadafora, A., 2020. Wannacry was the most common crypto ransomware attack last year. URL: <https://www.techradar.com/uk/news/wannacry-was-the-most-common-crypto-ransomware-attack-last-year>.
- Tridgell, A., 1999. Efficient algorithms for sorting and synchronization. Ph.D. thesis. Australian National University Canberra.
- Upchurch, J., Zhou, X., 2015. Variant: a malware similarity testing framework, in: 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), IEEE. pp. 31–39.
- VirusTotal, 2019. Virustotal. URL: <https://www.virustotal.com/#/home/upload>.
- VirusTotal, 2019. YARA in a nutshell. URL: <https://virustotal.github.io/yara/>.
- Wicherski, G., 2009. peHash: A novel approach to fast malware clustering. *LEET* 9, 8.
- Yang, M.S., Wu, K.L., 2006. Unsupervised possibilistic clustering. *Pattern Recognition* 39, 5–21.