Augmented YARA Rules Fused With Fuzzy Hashing in Ransomware Triaging

Nitin Naik[1,2], Paul Jenkins[1,2], Nick Savage[1], Longzhi Yang[3], Kshirasagar Naik[4] and Jingping Song[5]

[1]School of Computing, University of Portsmouth, United Kingdom
[2]Faculty of Science and Technology, Bournemouth University, United Kingdom
[3]Department of Computer and Information Sciences, Northumbria University, United Kingdom
[4]Department of Electrical and Computer Engineering, University of Waterloo, Canada
[5]Software College, Northeastern University, China
Email: {nitin.naik, paul.jenkins, nick.savage}@port.ac.uk, {nnaik, pjenkins}@bournemouth.ac.uk,
longzhi.yang@northumbria.ac.uk, snaik@uwaterloo.ca, songjp@swc.neu.edu.cn

***Abstract*—Triaging is an initial stage of malware analysis to assess whether a sample is malware or not and the degree of similarity it holds with known malware. It can be applied to any malware category such as ransomware, which is a type of malware that blocks access to a system or data, usually by encrypting it. It has become the main modus operandi for cybercriminals to extort monies from victims due to the growth of cryptocurrencies. Consequently, it severely affects all types of users whether they be from corporates or ordinary home users. Ransomware can be prevented in several different ways, however, the simple and initial step in prevention is its triaging without execution. Several triaging methods are in use such as fuzzy hashing, import hashing and YARA rules, amongst all, YARA rules are one of the most popular and widely used methods. Nonetheless, its success or failure is dependent on the quality of rules employed for malware triaging. This paper performs ransomware triaging using fuzzy hashing, import hashing and YARA rules and demonstrates how YARA rules can be improved using fuzzy hashing to obtain relatively better triaging results. Subsequently, it proposes the augmented YARA rules fused with fuzzy hashing to obtain improved triaging results and performance efficiency in comparison to all three triaging methods individually. Finally, the paper demonstrates how the use of the fused YARA rules can improve triaging results irrespective of the type of malware.**

***Index Terms*—YARA Rules; Fuzzy Hashing; Import Hashing; Ransomware; SSDEEP; SDHASH; mvHASH-B; WannaCry; WannaCryptor.**

I. Introduction

Cyberattacks have become more sophisticated over the years and cybercriminals are inventing novel attack techniques to cause maximum damage and gain. With the invention of cryptocurrencies, ransomware has now become the primary modus operandi for cybercriminals as their preferred extortion tool, whilst maintaining their anonymity. Due to the financial incentive, cybercriminals are developing a large volume of ransomware daily, which is increasing at an exponential rate. Nonetheless, the large volume of ransomware samples requires an effective assessment technique to determine the authenticity of samples and its prioritisation for the further analysis. This assessment can generate significant savings in computational resources by filtering out a large number of hoax samples, leading to the precise analysis of likely malware samples, saving significant time of cyber analysts [1]. This process of preliminary assessment and prioritisation of samples is known as triaging [2]. Triaging can be performed as a static process or dynamic process [3], however, the static process is relatively safe as it investigates samples without their execution. Several methods are used as a means of static assessment such as fuzzy hashing, import hashing and YARA rules, amongst all, YARA rules are one of the most popular and widely used methods.

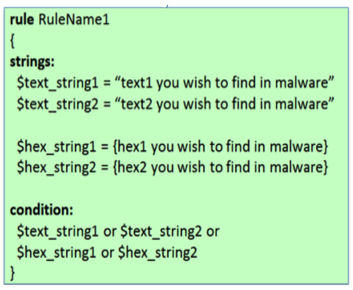
YARA rules are created based on the descriptions of malware families and text or binary strings contained within malware samples. YARA Rules can be used to assess static files or running processes to determine whether a sample belongs to a particular malware family. However, the success or failure of YARA rules is dependent on the quality of rules employed for malware triaging. This paper performs ransomware triaging using fuzzy hashing, import hashing and YARA rules and demonstrates how YARA rules can be improved using fuzzy hashing to obtain relatively better triaging results. It employs the three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B, import hashing method IMPHASH and standard YARA rules for ransomware triaging. Later, based on all triaging results and their analysis, it proposes the augmented YARA rules fused with fuzzy hashing to perform triaging that can generate optimal triaging results in comparison to all three triaging methods individually. The proposed fused YARA rules can produce improved triaging results irrespective of the type of malware with almost similar operational efficiency to that of standard YARA rules, as demonstrated through the ransomware triaging experimentation.

The paper is organised into the subsequent sections: Section II describes different triaging methods: YARA Rules, Import Hashing and Fuzzy Hashing with its different methods SSDEEP, SDHASH and mvHASH-B; Section III explains the process of gathering WannaCry/WannaCryptor ransomware samples for performing triaging operations using the selected triaging methods. Section IV explains the ransomware triaging process using the selected triaging methods. Section V describes the proposed fused YARA rules for triaging operation. Section VI presents the experimental evaluation of all selected triaging methods and their comparative analysis with the proposed fused YARA rules. Finally, Section VII presents the summary of the paper and suggests some future enhancements.

II. Triaging Methods

A. *YARA Rules*

YARA rules based technique is a pattern matching technique developed for the research community to discover and classify malware [4]. It is also known as a *swiss knife* for malware analysis. It offers a simple and effective way of creating customised rules (called YARA rules), comprising descriptions of aimed malware dependent on strings or byte sequences discovered in it, which are used to find malevolent files or processes. YARA syntaxes and semantics are very similar to the C programming language [4]. It can be used through its command-line interface or Python scripts with the *yara-python* extension. YARA rules are an adaptable and effective approach to deal with the swiftly rising issue of malware. It can be used on all main operating systems Windows, Linux and Mac OS X [5].

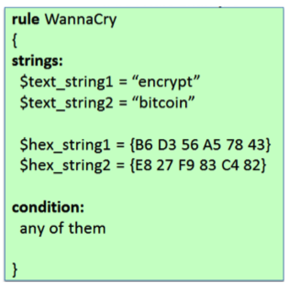


```
rule RuleName1
{
strings:
  $text_string1 = "text1 you wish to find in malware"
  $text_string2 = "text2 you wish to find in malware"

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

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

Fig. 1. Structure of YARA Rules



```
rule WannaCry
{
strings:
  $text_string1 = "encrypt"
  $text_string2 = "bitcoin"

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

condition:
  any of them

}
```

Fig. 2. Example of YARA Rules

B. *Import Hashing - IMPHASH*

Import hashing is a very simple and efficient approach of triaging developed by Mandiant [6]. It can be used to discover the similarity of unknown samples with known malware based on the hash generated from the import address table (IAT) of a portable executable (PE) [6]. This hashing technique is different from most other hashing techniques, where hash is generated from only one section of PE file (called *Imports*) rather than from an entire file. Imports are mainly functions that a program/file calls from other programs/files (usually numerous DLL, EXE, SYS files that provide functionality to the Windows operating system). Therefore, the hash is generated based on the *Imports/Functions* and dependent on their particular order within the executable file. The hash generated by this technique is known as IMPHASH. The generation of IAT and subsequently the IMPHASH from the IAT is profoundly dependent on the sequence of functions written in the source file and the sequence of source files during compile time. Therefore, when the two files based on the same source code are compiled, they will generate an identical IAT and an identical IMPHASH value. Likewise, when the two files have different source code, they will generate distinct IATs and distinct IMPHASH values.

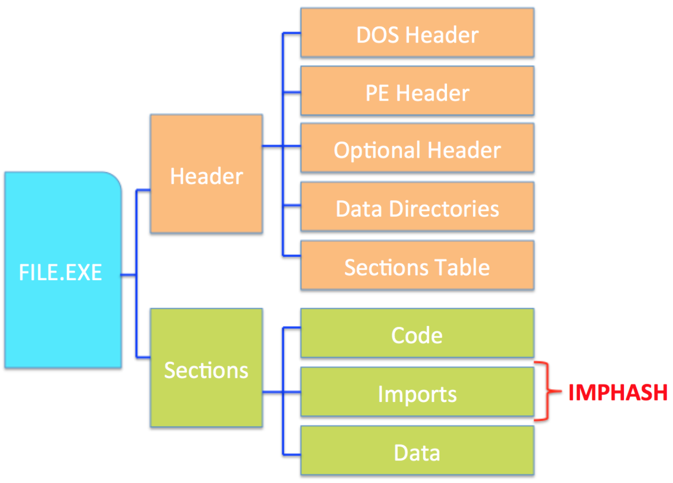


Fig. 3. Generation of IMPHASH Value from the Import Address Table (IAT) of a Portable Executable (PE) File

C. *Fuzzy Hashing*

In security analysis, hashing is used to determine both the integrity and similarity of files under examination, the latter utilising cryptographic techniques and the former utilising fuzzy techniques [7], [8]. In malware investigation, when attempting to determine malware strains it is the similarity of the sample which is of interest as often malware developers utilise similar code, leading to different variants [9]. In this type of analysis, generally, a file is divided into multiple blocks and a hash value is calculated for each block, the final step being the concatenation of all hash values of the blocks to generate the fuzzy hash value as shown in Fig. 4. Several factors are involved in determining the length of fuzzy hash value including the block size, the file size, and the output size of the selected hash function [10]. In contrast the complete file is hashed in cryptographic hashing with the output hash having a fixed size irrespective of input file size. There are different categories of fuzzy hashing techniques, classified as follows: Context-Triggered Piecewise Hashing (CTPH), Statistically-Improbable Features (SIF), Block-Based Hashing (BBH) and Block-Based Rebuilding (BBR) [11], [12], [13]. The comparison of files in forensic analysis, where known malware files are compared with unknown samples for the purpose of triaging and clustering of malware to identify new variants, requires an understanding of the degree of similarity between samples. This suggests the use of the similarity preserving characteristic of fuzzy hashing which is effective in forensic investigation when comparing new samples with existing malware families for their triage and clustering, in samples which have the same functionality, but not the same cryptographic hash values [2].

Generally, the similarity of samples can be measured based upon their syntactic or semantic levels [2]. At a syntactic level, two files are compared to find similarity on the basis of their byte sequence of data but not the context of data. Whereas at semantic level, two files are compared to find similarity on the basis of their context [2]. Fuzzy hashing is only utilised to find similarity between two files at syntactic level.

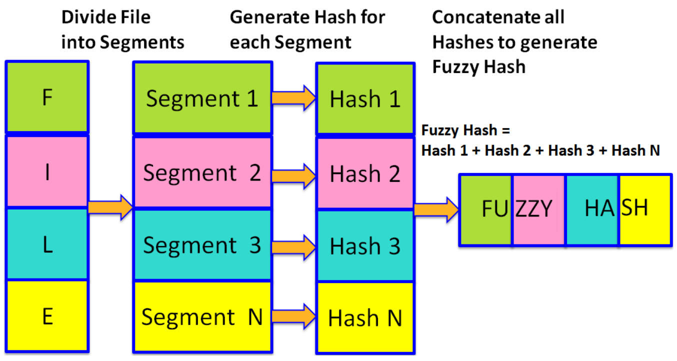

Divide File into Segments
Generate Hash for each Segment
Concatenate all Hashes to generate Fuzzy Hash
F
I
L
E
Segment 1
Segment 2
Segment 3
Segment N
Hash 1
Hash 2
Hash 3
Hash N
Fuzzy Hash = Hash 1 + Hash 2 + Hash 3 + Hash N
FU ZZY HA SH

Fig. 4. Generation of Fuzzy Hash Value in Fuzzy Hashing Method

1) SSDEEP: The SSDEEP fuzzy hashing method was initially developed for finding spam emails [9]. This method divides a file into number of blocks based on the content of that file. The endpoint points of these blocks are determined by a rolling hash method utilising the Adler32 function [10]. Generating the SSDEEP fuzzy hash value for the file, consists of calculating an individual hash value for each block and concatenating these into a single hash value. Similarity between the two files is calculated by utilising Damerau-Levenshtein distance function.

2) SDHASH: The SDHASH fuzzy hashing method finds common and rare features in a file and matches the rare features in another file to determine the degree of similarity between the two files [14]. Generally a feature is a 64-byte string and is found using an entropy calculation. It employs the cryptographic hash function SHA-1 and Bloom filters to calculate the SDHASH fuzzy hash value of a file [15]. A Bloom filter is a space-efficient probabilistic data structure to find whether the element is definitely not present in the set or may be present in the set. Similarity between the two files is calculated by utilising a Hamming distance function.

3) mvHASH-B: The mvHASH-B fuzzy hashing method is slightly different from SDHASH fuzzy hashing method, which focuses on keeping the data unchanged even if there is a small change in it. Thus resulting the same hash value in case of a minor change and preserving the similarity. However, mvHASH-B transforms the input data based on the concept of majority votes, then encodes the majority vote bit sequence with RLE (Run-Length Encoding - a type of lossless data compression approach), and finally generates mvHASH-B fuzzy hash value utilising Bloom filters [16]. Moreover, it uses a self-defined hash function which has a higher run time efficiency and its complexity is equivalent to the cryptographic hash function SHA-1.

III. Collecting WannaCry/WannaCryptor Ransomware Samples

A Ransomware attack is a nefarious attack to extort money from victims which is a more sophisticated tactic than the DDoS attack [17], [18], [19]. It causes loss of money and reputational damage to the business and sometimes potentially permanent loss of data. Ransomware attacks could be a minor or severe depending on the category of ransomware, nonetheless, certain types of ransomware have iniquitous intentions. Such ransomware are the priority for this investigation such as WannaCry or WannaCryptor ransomware is one of the most significant variants of ransomware recently and is selected for this study [20], [21], [22], [23]. The most labour intensive task was the collection of credible samples of the WannaCry ransomware. As a result of this process, it was decided to collect a reasonable number of WannaCry or WannaCryptor ransomware samples which could be easily investigated manually. All the WannaCry samples were gathered from two sources *Hybrid Analysis* [24] and *Malshare* [25] and their analysis were performed through the information acquired by *VirusTotal* [26]. The main difficulty was to verify the credibility of the collected ransomware samples that they were very likely to be WannaCry ransomware samples. The credibility of samples was evaluated through the criteria set on the basis of the result of various detection engines on VirusTotal, which was greater than or equal to 40, meaning a minimum of 40 detection engines on VirusTotal diagnosed the particular sample as ransomware/malware. To verify that they were WannaCry or WannaCryptor ransomware, they were manually checked on every detection engine, where a number of the engines identified a sample as a WannaCry or WannaCryptor ransomware. Nevertheless, this ransomware verification process was complex, and mainly dependent on the discretion of authors [27], [28], [29], [30], [31]. The selection process was lengthy and demanding, consequently, 112 samples of WannaCry or WannaCryptor ransomware were selected after each sample was fully analysed manually.

IV. Ransomware Triaging Using Selected Triaging Methods: YARA Rules, Fuzzy Hashing and Import Hashing

Initially, the three triaging methods fuzzy hashing, import hashing and YARA rules are selected to perform the triaging operation on the collected WannaCry ransomware corpus. The main reason for selecting these three triaging methods is to perform static, fast and efficient analysis of collected samples. Most hashing methods are resource-optimised, cost-effective and fast in execution; in particular, import hashing is one of the best-fit method for all these criteria [2]. Alongside the accuracy of a triaging method, all these criteria are very crucial while considering any triaging method for the analysis of a large volume of malware. However, both fuzzy hashing and import hashing only check structural similarity and are not effective on packed samples.

The fuzzy hashing method is applied on the unpacked sample (the sample requires unpacking prior to fuzzy hashing if it is a packed sample). The fuzzy hashing method generates a fuzzy hash value of the sample and matches it against the database of fuzzy hashes of known ransomware or it can be directly matched against the database of known samples. If it finds one or more matched ransomware samples it generates the results in the form of their degree of similarity with the sample. It can indicate which is the exact or closest matched ransomware sample based on the highest degree of similarity. However, the interpretation of the fuzzy hashing result is dependent on the security expert and how efficiently they utilise it for further advanced analysis. The fuzzy hashing is a triaging process and its result is a preliminary indication which requires a further clustering or classification operation to conclude as a meaningful result [32]. Import hashing generates an IMPHASH value for each sample and works in a similar way to fuzzy hashing with the exception that it only determines binary similarity (i.e. whether the sample is matched or not without giving any degree of similarity). Both hashing methods can only be used for detecting similarity of samples with known malware samples.

YARA rules are different from hashing methods and performs triaging of malware based on the created customised rules. It is a more flexible and effective method, however, its effectiveness is dependent on the quality of the rules. Developing effective rules is a challenging task and requires expertise and experience. YARA rules can be generated automatically, however, these rules require further processing to customise and may still not be effective for the targeted operation. In this ransomware triaging experiment, the utilised YARA rules are created by *yarGen* tool [33]. It is based on Fuzzy Regular Expressions, Naive Bayes Classifier and Gibberish Detector [34] with the default setting of the top 20 strings based on their score and without IMPHASH, because IMPHASH is used as an independent triaging method in this work. To ensure that YARA rules are comparable with the other triaging methods, individual rules are tested alongside super rules. YARA rules based triaging operation may be slower as it is dependent on the set of attributes included in the rules and its searching criteria.

V. Ransomware Triaging Using the Proposed Fused YARA Rules

Standard YARA rules include certain strings/attributes unique to malware or malware families and may not be effective in cases where it could not find those included strings/attributes of the rule(s) due to intelligible modifications made by threat actors in their new malware samples for the purpose of evading common YARA rules. Increasing the number of strings/attributes or rules may not be effective for triaging as it can cause redundancy, slow down the operation and increase computational overheads. Furthermore, writing sophisticated YARA rules requires sufficient knowledge of advanced aspects of YARA rules, which demands expertise, experience and significant time [35], [36], [37]. Therefore, it is important to make simple and effective YARA rules without having all these adverse effects on the triaging performance. This necessitates the requirement to augment YARA rules using some different mechanism other than unique strings/attributes. Fuzzy hashing is a fast and resource-optimised method which can sometimes produce improved triaging results in comparison to YARA rules, as demonstrated in ransomware triaging [2]. This indicates that fuzzy hashing could be effective in case of structural similarity even though the selective strings/attributes of YARA rules may not be found in malware.

If the fuzzy hashing is embedded in YARA rules then it can increase the triaging performance of the fused YARA rules by matching the structural similarity, other than selective strings/attributes without affecting the efficiency and overheads as fuzzy hashing is compact and fast. Additionally, it can provide the degree of similarity for the matched sample with the existing malware due to the fuzzy characteristic of fuzzy hashing which is not possible in standard YARA rules. It can further improve the clustering or classification results of YARA rules by obtaining the combined results of both triaging methods, increasing the confidence level of the triaging operation. In this paper, three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B are evaluated to determine the best-fit fuzzy hashing method to embed with YARA rules offering fast, efficient, and more precise triaging results.

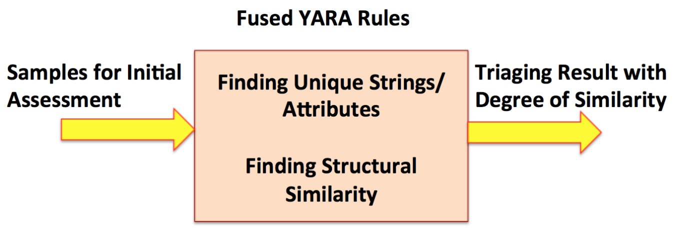

Fused YARA Rules
Samples for Initial Assessment
Finding Unique Strings/ Attributes
Finding Structural Similarity
Triaging Result with Degree of Similarity

Fig. 5. YARA Rules Fused with Fuzzy Hashing

VI. Experimental Evaluation of Different Triaging Methods for WannaCry Ransomware

All the collected ransomware samples belong to only one ransomware category (i.e. WannaCry) and therefore they possess some similarity with each other despite the fact that they are different variants of WannaCry ransomware. These WannaCry samples are also analysed manually to check that each sample possesses some similarity with at least one other sample. After the manual analysis and verification of all the samples, the testing criterion for all the triaging methods is set to find the total number of samples that are successfully identified as a known WannaCry ransomware (i.e. each sample is matched with at least one other sample in the corpus as every sample is a WannaCry ransomware) by each triaging method. In other words, this criterion is to find the miss rate of each triaging method that is how many samples it could not identify as WannaCry ransomware.

A. Comparative Evaluation of the Triaging Results of Different Fuzzy Hashing Methods (SSDEEP, SDHASH and mvHASH-B)

Initially, the three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B were applied to the collected WannaCry ransomware corpus evaluating the similarity detection success rate of each fuzzy hashing method. The similarity detection results were analysed on the basis of different similarity threshold criteria (covering all matched samples (from 1-100%), matched above 10%, matched above 20%, and matched above 30%) as shown in Table I. Here, the first row of Table I shows the total number of matched samples with one or more than one other samples in the ransomware corpus of 112 samples, where SSDEEP found similarity in 104 samples, SDHASH found similarity in 108 samples and mvHASH-B found similarity in 108 samples. Likewise, the other similarity detection results were determined for three similarity thresholds of above 10%, 20% and 30% (see Table I), where only those results were considered which were above the decided similarity threshold. The analysis of these four similarity detection results shows that most of the SSDEEP similarity scores are above the set highest threshold of 30%, resulting in its lowest matched samples of 103. In case of SDHASH and mvHASH-B, several similarity scores are below the set highest threshold of 30% resulting in its lowest matched samples of 104 and 102 respectively. While comparing SDHASH and mvHASH-B, the latter producing greater lower similarity scores than SDHASH methods reflected by its lowest matched samples size of 102. In summary, the similarity detection rate for all the three fuzzy hashing methods was quite high (above 91%). This comparison of similarity scores and success rate of each fuzzy hashing is very important because it will be combined with other performance criteria to determine the best-fit fuzzy hashing method for embedding in YARA rules.

B. Comparative Evaluation of the Triaging Results of Fuzzy Hashing, Import Hashing and YARA Rules

In the previous subsection, the three different fuzzy hashing methods and their triaging results are analysed and finally their triaging results above the threshold of 30% similarity scores are considered as final results of each method for further analysis purposes. These triaging results are compared with triaging results of two other triaging methods: import hashing and YARA rules as shown in the Table II. However, the results of both triaging methods are slightly lower than any of the fuzzy hashing methods for this particular WannaCry ransomware corpus. This indicates that despite the success of YARA rules, sometimes it cannot generate good results and this leads to the requirement of some additional enhancement to include such situations and produce better results.

C. Comparative Evaluation of the Triaging Results of Fused YARA Rules with Different Fuzzy Hashing Methods

As indicated the moderate results of YARA rules and the slightly better results of fuzzy hashing in the previous subsection, when the YARA rules are fused with fuzzy hashing to obtain better triaging results irrespective of any malware corpus. The results of the fused YARA rules with three fuzzy hashing methods are shown in Table III, where all three fuzzy hashing methods have provided the same triaging results of 110 matched samples (i.e. approximately 98.21%) for this particular WannaCry ransomware corpus. This triaging result of fused YARA rules is better than any individual fuzzy hashing method and YARA rules. This shows the successful fusion of these two training methods and the success of the proposed fused YARA rules.

D. Performance Evaluation of Fused YARA Rules with Fuzzy Hashing

Table IV and Fig. 6 show the comparison of the triaging results of fused YARA rules with all the three triaging methods and its success in producing the best triaging results amongst all. However, it is important to evaluate the operational efficiency and overheads of this fused YARA rules method in comparison to the standard YARA rules. This performance evaluation of fused YARA rules is shown in Table V, where YARA rules fused with SSDEEP fuzzy hashing methods offers almost similar performance with slightly increased rule generation time. Nonetheless, the other two fuzzy hashing methods SDHASH and mvHASH-B offer good performance but with relatively higher overheads than the SSDEEP fuzzy hashing method. Therefore, for this particular WannaCry corpus and implementation, YARA rules can be fused with SSDEEP fuzzy hashing method to obtain the better results and performance efficiency.

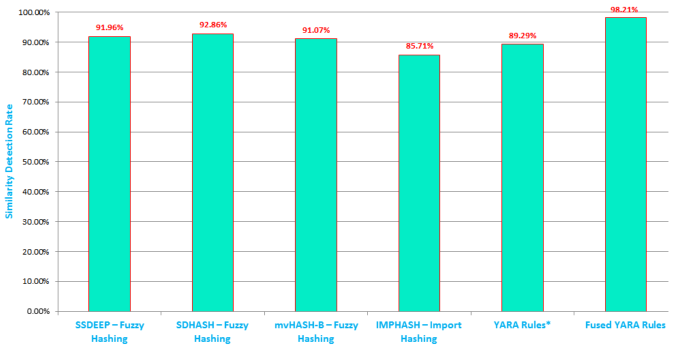


Fig. 6. Comparison of Similarity Detection Rate of Fused YARA Rules with other Triaging Methods

VII. Conclusion

This paper proposed the augmented YARA rules fused with fuzzy hashing method to obtain better triaging results and performance efficiency in ransomware triaging. Initially, it performed ransomware triaging using fuzzy hashing, import hashing and YARA rules. For this ransomware triaging purpose, it employed the three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B, import hashing method IMPHASH and standard YARA rules. The triaging results of all fuzzy hashing methods were slightly better than YARA

TABLE I
COMPARISON OF TRIAGING RESULTS OF FUZZY HASHING METHODS - SSDEEP, SDHASH AND MVHASH-B

Fuzzy Hashing Matching Criteria for Ransomware Samples	**Number of Samples Matched by SSDEEP**	**Number of Samples Matched by SDHASH**	**Number of Samples Matched by mvHASH-B**
Based on all Fuzzy Similarity Scores (from 1-100%)	104	108	108
Based on Fuzzy Similarity Scores above the threshold of 10%	104	108	108
Based on Fuzzy Similarity Scores above the threshold of 20%	104	106	103
Based on Fuzzy Similarity Scores above the threshold of 30%	103	104	102

TABLE II
COMPARISON OF TRIAGING RESULTS OF TRIAGING METHODS - FUZZY HASHING, IMPORT HASHING AND YARA RULES

WannaCry Ransomware Corpus	**Number of Samples Matched by SSDEEP**	**Number of Samples Matched by SDHASH**	**Number of Samples Matched by mvHASH-B**	**Number of Samples Matched by IMPHASH**	**Number of Samples Matched by YARA Rules***
Total samples matched out of 112 samples	103	104	102	96	100
Where * represents that employed YARA rules are generated by **yarGen** tool (based on Fuzzy Regular Expressions, Naive Bayes Classifier and Gibberish Detector) with the default setting of the top 20 strings based on their score and without IMPHASH.					

TABLE III
COMPARISON OF TRIAGING RESULTS OF FUSED YARA RULES WITH DIFFERENT FUZZY HASHING METHODS

WannaCry Ransomware Corpus	**YARA Rules Fused with SSDEEP**	**YARA Rules Fused with SDHASH**	**YARA Rules Fused with mvHASH-B**
Total samples matched out of 112 samples	110	110	110

TABLE IV
COMPARISON OF TRIAGING RESULTS OF FUSED YARA RULES WITH FUZZY HASHING, IMPORT HASHING AND YARA RULES

WannaCry Ransomware Corpus	**Number of Samples Matched by SSDEEP**	**Number of Samples Matched by SDHASH**	**Number of Samples Matched by mvHASH-B**	**Number of Samples Matched by IMPHASH**	**Number of Samples Matched by YARA Rules***	**Number of Samples Matched by Fused YARA Rules**
Total samples matched out of 112 samples	103	104	102	98	100	110

rules and IMPHASH. This means that despite the success of YARA rules, on some occasions, it could not generate good results, resulting in augmented YARA rules fused with fuzzy hashing which were developed and tested, producing optimal results in comparison to any other individual triaging methods. Furthermore, the performance efficiency of the fused rules was compared against the standard YARA rules and it showed that fusion of YARA rules with the SSDEEP fuzzy hashing method can offer almost similar performance efficiency as standard YARA rules, however, the other two fuzzy hashing methods SDHASH and mvHASH-B are good but could cost slightly more in performance overheads. This proposed fused YARA rules performed well for this particular WannaCry corpus and implementation. However, in future, this proposed fused YARA rules would be tested on the two grounds: large samples size and diverse samples of ransomware or malware.

ACKNOWLEDGEMENT

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

REFERENCES

[1] J. Jang, D. Brumley, and S. Venkataraman, “Bitshred: feature hashing malware for scalable triage and semantic analysis,” in *Proceedings of*

TABLE V
COMPARISON OF PERFORMANCE EFFICIENCY OF FUSED YARA RULES WITH STANDARD YARA RULES

Performance Criteria	**Standard YARA Rules***	**YARA Rules Fused with SSDEEP**	**YARA Rules Fused with SDHASH**	**YARA Rules Fused with mvHASH-B**
Similarity Detection Criteria	Textual/Binary Patterns	Textual/Binary Patterns and Byte Structure	Textual/Binary Patterns and Byte Structure	Textual/Binary Patterns and Byte Structure
Rule Size	Size is dependent on the nature of of the rule (mainly number of Strings/Attributes)	It will not affect significantly	Slight increase in size	Slight increase in size
Run-Time Efficiency	Speed is also dependent on the nature of of the rule (mainly number of Strings/Attributes)	Slightly slower than standard YARA Rules	It requires time upto twice than standard YARA Rules	It requires time upto twice than standard YARA Rules
Minimum File Size for Fuzzy Hash Generation	It does not include fuzzy hash value in rules	It may not generate fuzzy hash of a file lower than 4 KB in size	It may not generate fuzzy hash of a file lower than 512 Byte in size	It may not generate fuzzy hash of a file lower than 2 KB in size

the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 309–320.

[2] N. Naik, P. Jenkins, N. Savage, and L. Yang, “Cyberthreat Hunting-Part 1: Triaging Ransomware using Fuzzy Hashing, Import Hashing and YARA Rules,” in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019.

[3] C. Harrell. (2013) Finding Malware: Like Iron Man. [Online]. Available: https://digital-forensics.sans.org/summit-archives/DFIR_Summit/Finding-Malware-Like-Iron-Man-Corey-Harrell.pdf

[4] VirusTotal. (2019) YARA in a nutshell. [Online]. Available: https://virustotal.github.io/yara/

[5] Readthedocs. (2019) Writing YARA rules. [Online]. Available: https://yara.readthedocs.io/en/v3.5.0/writingrules.html

[6] Mandiant. (2014) Tracking malware with import hashing. [Online]. Available: https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html

[7] N. Naik, P. Jenkins, and N. Savage, “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)*, 2019.

[8] N. Naik, P. Jenkins, J. Gillett, H. Mouratidis, K. Naik, and J. Song, “Lockout-Tagout Ransomware: A detection method for ransomware using fuzzy hashing and clustering,” in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.

[9] J. Kornblum, “Identifying almost identical files using context triggered piecewise hashing,” *Digital investigation*, vol. 3, pp. 91–97, 2006.

[10] A. Tridgell, “Efficient algorithms for sorting and synchronization,” Ph.D. dissertation, Australian National University Canberra, 1999.

[11] F. Breitinger and H. Baier, “A fuzzy hashing approach based on random sequences and hamming distance,” in *Annual ADFSL Conference on Digital Forensics, Security and Law. 15*, 2012. [Online]. Available: https://commons.erau.edu/adfsl/2012/wednesday/15

[12] C. Sadowski and G. Levin, “Simhash: Hash-based similarity detection,” 2007. [Online]. Available: www.webrankinfo.com/dossiers/wp-content/uploads/simhash.pdff

[13] V. Gayoso Martínez, F. Hernández Álvarez, and L. Hernández Encinas, “State of the art in similarity preserving hashing functions,” 2014. [Online]. Available: http://digital.csic.es/bitstream/10261/135120/1/Similarity_preserving_Hashing_functions.pdf

[14] V. Roussev, “Data fingerprinting with similarity digests,” in *IFIP International Conference on Digital Forensics*. Springer, 2010, pp. 207–226.

[15] ——, “An evaluation of forensic similarity hashes,” *digital investigation*, vol. 8, pp. S34–S41, 2011.

[16] F. Breitinger, K. P. Astebøl, H. Baier, and C. Busch, “mvhash-b-a new approach for similarity preserving hashing,” in *2013 Seventh International Conference on IT Security Incident Management and IT Forensics*. IEEE, 2013, pp. 33–44.

[17] N. Naik, P. Jenkins, R. Cooke, D. Ball, A. Foster, and Y. Jin, “Augmented windows fuzzy firewall for preventing denial of service attack,” in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[18] N. Naik and P. Jenkins, “Fuzzy reasoning based windows firewall for preventing denial of service attack,” in *IEEE International Conference on Fuzzy Systems*, 2016, pp. 759–766.

[19] ——, “Enhancing windows firewall security using fuzzy reasoning,” in *IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2016, pp. 263–269.

[20] J. M. Ehrenfeld, “Wannacry, cybersecurity and health information technology: A time to act,” *Journal of medical systems*, vol. 41, no. 7, p. 104, 2017.

[21] R. Richardson and M. North, “Ransomware: Evolution, mitigation and prevention,” *International Management Review*, vol. 13, no. 1, pp. 10–21, 2017.

[22] K. Cabaj, P. Gawkowski, K. Grochowski, and D. Osojca, “Network activity analysis of Cryptowall ransomware,” *Przeglad Elektrotechniczny*, vol. 91, no. 11, pp. 201–204, 2015.

[23] Y. Klijnsma. (2019) The history of Cryptowall: a large scale cryptographic ransomware threat. [Online]. Available: https://www.cryptowalltracker.org/

[24] Hybrid-Analysis. (2019) Hybrid Analysis. [Online]. Available: https://www.hybrid-analysis.com/

[25] Malshare. (2019) A free Malware repository providing researchers access to samples, malicious feeds, and YARA results. [Online]. Available: https://malshare.com/index.php

[26] VirusTotal. (2019) Virustotal. [Online]. Available: https://www.virustotal.com/#/home/upload

[27] N. Naik, P. Jenkins, B. Kerby, J. Sloane, and L. Yang, “Fuzzy logic aided intelligent threat detection in cisco adaptive security appliance 5500 series firewalls,” in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018.

[28] N. Naik, P. Jenkins, R. Cooke, and L. Yang, “Honeypots that bite back: A fuzzy technique for identifying and inhibiting fingerprinting attacks on low interaction honeypots,” in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018.

[29] N. Naik, P. Jenkins, and N. Savage, “Threat-aware honeypot for discovering and predicting fingerprinting attacks using principal components

analysis," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018.

[30] N. Naik and P. Jenkins, "A fuzzy approach for detecting and defending against spoofing attacks on low interaction honeypots," in *21st International Conference on Information Fusion (Fusion)*. IEEE, 2018, pp. 904–910.

[31] ——, "Discovering hackers by stealth: Predicting fingerprinting attacks on honeypot systems," in *2018 IEEE International Symposium on Systems Engineering (ISSE)*, 2018.

[32] N. Naik, P. Jenkins, N. Savage, and L. Yang, "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, 2019.

[33] F. Roth. (2018) yarGen is a generator for YARA rules. [Online]. Available: https://github.com/Neo23x0/yarGen

[34] ——. (2017) How to post-process YARA rules generated by yarGen. [Online]. Available: https://medium.com/@cyb3rops/how-to-post-process-yara-rules-generated-by-yargen-121d29322282

[35] V. Alvarez. (2019) YARA Documentation, Release 3.10. 0. [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/yara/latest/yara.pdf

[36] R. Dias. (2014) Intelligence-Driven Incident Response with YARA. [Online]. Available: https://www.sans.org/reading-room/whitepapers/forensics/intelligence-driven-incident-response-yara-35542

[37] C. S. Culling. (2018) Which YARA Rules : Basic or Advanced? [Online]. Available: https://vt-gtm-wp-media.storage.googleapis.com/2.0-Which-YARA-Rules-Rule-Basic-or-Advanced-1.pdf