Fuzzy Hashing Aided Enhanced YARA Rules for Malware Triaging

Nitin Naik[1], Paul Jenkins[2], Nick Savage[2], Longzhi Yang[3], Kshirasagar Naik[4], Jingping Song[5], Tossapon Boongoen[6] and Natthakan Iam-On[6]

[1]School of Informatics and Digital Engineering, Aston University, United Kingdom
[2]School of Computing, University of Portsmouth, 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
[6]School of Information Technology, Mae Fah Luang University, Thailand
Email: n.naik1@aston.ac.uk, {paul.jenkins, nick.savage}@port.ac.uk, longzhi.yang@northumbria.ac.uk, snaik@uwaterloo.ca, songjp@swc.neu.edu.cn, tossapon.boo@mfu.ac.th, natthakan@mfu.ac.th

***Abstract*—Cybercriminals are becoming more sophisticated wearing a mask of anonymity and unleashing more destructive malware on a daily basis. The biggest challenge is coping with the abundance of malware created and filtering targeted samples of destructive malware for further investigation and analysis whilst discarding any inert samples, thus optimising the analysis by saving time, effort and resources. The most common technique is malware triaging to separate likely malware and unlikely malware samples. One such triaging technique is YARA rules, commonly used to detect and classify malware based on string and pattern matching, rules are triggered and alerted when their condition is satisfied. This pattern matching technique used by YARA rules and its detection rate can be improved in several ways, however, it can lead to bulky and complex rules that affect the performance of YARA rules. This paper proposes a fuzzy hashing aided enhanced YARA rules to improve the detection rate of YARA rules without significantly increasing the complexity and overheads inherent in the process. This proposed approach only uses an additional fuzzy hashing alongside basic YARA rules to complement each other, so that when one method cannot detect a match, then the other technique can. This work employs three triaging methods fuzzy hashing, import hashing and YARA rules to perform extensive experiments on the collected malware samples. The detection rate of enhanced YARA rules is compared against the detection rate of the employed triaging methods to demonstrate the improvement in the overall triaging results.**

***Index Terms*—Malware Triaging; YARA Rules; Fuzzy Hashing; Import Hashing; Ransomware; Indicator of Compromise; IoC String.**

I. Introduction

The significant growth in malware and its related cyberattacks is a major concern for every individual, organisation, business and government. The malware writer exploits simple software-aided techniques to advanced AI-aided techniques to perform these cyberattacks in a more sophisticated manner [1]. Therefore, counter measures to prevent these cyberattacks require a corresponding sophisticated response, including more intelligent mechanisms. However, due to the large quantity of malware produced and collected daily, it is equally important to scrutinise these large sample sizes for their validity and concentrate the main effort and time for those samples which are indicated as a malicious in nature. This requires a preliminary investigation process called triaging to determine likely malware and unlikely malware samples [2]. This triaging process can be a static process or dynamic process [3], where dynamic process is more comprehensive, nonetheless, the static process is comparatively safer as all samples are investigated without being executed or run. Some popular static triaging methods are fuzzy hashing, import hashing and YARA rules, where YARA rules have become the most extensively used method.

YARA rules are one of the most effective and popular malware triaging methods used to detect malware based on string and pattern matching. The Indicator of Compromise (IoC) string is one of the most important parameters of YARA rules, including how many IoC strings and how they are selected for a rule, which is crucial for its success. Nonetheless, the generation of YARA rules requires a thorough understanding of security and an in-depth analysis of malware and their families. Though YARA rules can be generated easily and automatically, however, they still require further processing to achieve their optimal performance. When YARA rules are triggered, a malware alert is generated for a sample when the set condition (mostly a string-matching condition) in the rule is satisfied otherwise the sample is not alerted as malware. If only few or none of the selected strings are found in the targeted samples then YARA rules do not flag samples as malware even though they may be malware.

There are a number of mechanisms to remedy this issue, for example, the addition of more strings to extend its search criteria, however, there is a disadvantage as increased and complex rules can adversely affect the performance of YARA rules. Moreover, this is not an easy task for users to write such a complex set of YARA rules or modify automatically generated rules, as it requires significant expertise in computer security [4], [5], [6]. Consequently, it is essential to find an

easy solution to make YARA rules more efficient without incurring all complexities stated earlier. This paper proposes fuzzy hashing aided enhanced YARA rules to improve the detection rate of YARA rules without significantly increasing the complexity and overheads of YARA rules. The proposed method utilises an additional fuzzy hash function alongside the basic YARA rules, thus complementing each other, so that when one method cannot find a match, then the other can and vice versa [7]. This paper employs three triaging methods fuzzy hashing, import hashing and YARA rules to perform extensive experimentation and comparison on the collected ransomware samples, namely the four categories WannaCry, Locky, Cerber and CryptoWall. Subsequently, it evaluates the performance of the three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B to establish the best-fit fuzzy hashing method to integrate with YARA rules.

The paper is divided into the following sections: Section II discusses the chosen triaging methods YARA rules, import hashing and fuzzy hashing. Section III explains the collection and verification process of ransomware samples. Section IV discusses ransomware triaging process employing the selected triaging methods: fuzzy hashing, import hashing and YARA rules. Section V discusses the ransomware triaging process employing the proposed enhanced YARA rules. Lastly, Section VI presents the summary of the research work and suggests some future work.

II. Malware Triaging Methods

A. YARA Rules

YARA rules are developed to detect malware by primarily matching its signatures/strings with the existing malware signatures/strings [8], [9]. These rules contain predetermined signatures/strings related to known malware which is used in attempting to match against the targeted files, folders, or processes [10]. YARA rules consist of three sections: meta, strings and condition as shown in Figs. 1 and 2. Here, strings can be classified into three types of strings: text strings, hexadecimal strings and regular expression strings. Text strings are generally a readable text complemented with some modifiers (e.g., nocase, ASCII, wide, and fullword) to manage the process more effectively [11]. Hexadecimal strings are a sequence of raw bytes complemented with three flexible formats: wild-cards, jumps, and alternatives [11]. 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 [11]. Text strings and regular expressions which express a sequence of raw bytes through the use of escape sequences. The final part of YARA rules is a rule condition that specifies the number of signatures/strings required matching with the target to declare the sample as malware [12]. YARA conditions determine whether to trigger the rule or not, however, these conditions are Boolean expressions similar to those used in all other programming languages [11].

```
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
}
```

Fig. 1. YARA Rules: Syntax

```
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
}
```

Fig. 2. YARA Rules: Example

B. Fuzzy Hashing

Fuzzy hashing is used to determine the similarity between digital files, which makes it a very useful method for malware analysis as several pieces of malware and their variants possess some similarity with each other, which is not detected by a cryptographic hash as it has a binary outcome i.e., either the two files are exactly identical or not [13], [14]. In a fuzzy hashing technique, the file of interest is split 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 Fig. 3). 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 [15]. 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) [16], [17], [18]. 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 especially important when considering the analysis and clustering of suspected malware in order to discover new variants [19]. As a result, the use of the similarity preserving property of fuzzy hashing is useful in malware analysis while comparing unknown files with known malware families, where samples possess similar functionality, yet different cryptographic hash values [2].

1) SSDEEP: The SSDEEP fuzzy hashing technique was specially created to distinguish spam or junk emails [13]. 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 [15]. 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 the similarity distance of concerning files.

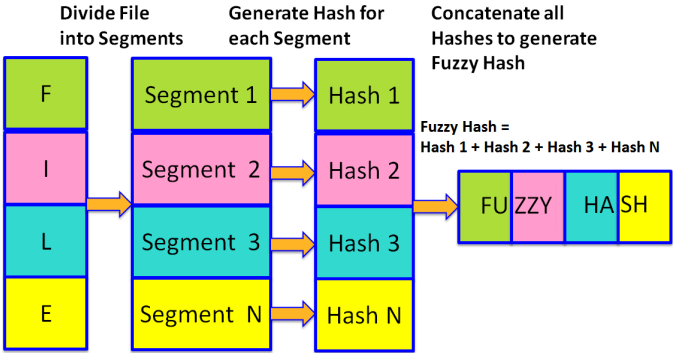

Fig. 3. Generation of Fuzzy Hash Value in Fuzzy Hashing Method [10]

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 these files [20]. 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.

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 [21]. Furthermore, it employs its own outlined hash function which is comparable with the standard SHA-1 function, with better run time efficiency.

C. Import Hashing - IMPHASH

Import hashing is another method that can be utilised to determine the similarity of digital files. It generates a hash value from the particular segment (Imports) of a Portable Executable (PE) file (see Fig. 4), which contains information about all the functions imported by an executable from DLLs. Import hashing utilises function calls included within a program, where the order in which they are called and these functions are utilised to generate a hash value called IMPort HASH (IMPHASH). Precisely, this IMPHASH is generated from the Import Address Table (IAT), which is a list of the program and their functions required by an executable including all the other DLL files which are to be bound and linked with the relocatable code of the original program to build the final application [22]. Thus, two pieces of software that were compiled with similar code except with a different order of functions will generate different IMPHASH values. This method is analogous to fuzzy hashing with regard to its speed, computation, complexity and hash size, however, it is noteworthy that IMPHASH provides a binary similarity result, rather than the degree of similarity of two files.

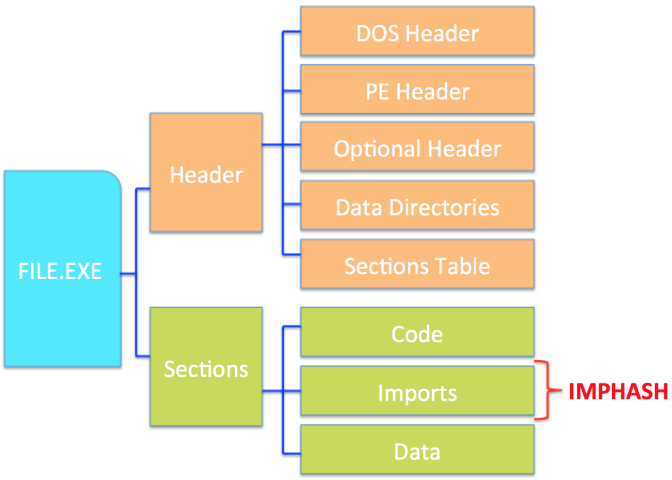

Fig. 4. Generation of IMPHASH Value from the Import Address Table (IAT) of a Portable Executable (PE) File [10]

III. Collection of Malware Samples

In this implementation, one of the most prevalent malware, ransomware was selected to perform all triaging operations and evaluating the performance of enhanced YARA rules. Ransomware was selected for the experiment as it is one of the most relevant and damaging malware that exploits victims for the financial gain. Numerous types of ransomware were created and used in cyberattacks, though, some ransomware categories were worthy of more focus due to their severity of attacks and financial loss. Based on primary research, four ransomware categories were targeted for this work: WannaCry, Locky, Cerber and CryptoWall [23], [24], [25]. Thousands of malware samples were downloaded from the two sources *Hybrid Analysis* [26] and *Malshare* [27]. Later, these samples were verified for their credibility as numerous samples were false samples. It was critical to select only credible samples of a specific category as a reference to test all triaging methods and enhanced YARA rules successfully. These samples were investigated based on the information available on *VirusTotal* [28]. To determine that every sample was indeed genuine malware or ransomware and belonged to a specific ransomware category, the criterion was set that it must be identified as malware by at least 40 or more detection engines on *VirusTotal*. To check 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 sample collection and verification process was very lengthy and time consuming. Finally, 1000 ransomware samples were selected out of several thousand samples, and equally divided 250 samples into four ransomware categories WannaCry, Locky, Cerber and CryptoWall. The four different categories of ransomware were chosen to evaluate how each triaging method works on different types of ransomware.

IV. Malware Triaging Process Using Fuzzy Hashing, Import Hashing and YARA Rules

Originally, the three selected triaging methods fuzzy hashing, import hashing and YARA rules are utilised to carry out the triaging process on the collected and verified ransomware samples, of the four ransomware corpora WannaCry, Locky, Cerber and CryptoWall. These three triaging methods are selected to perform static analysis which should be fast, efficient and resource-optimised. Fuzzy hashing and import hashing are compact, fast and resource-optimised triaging methods [2]. In addition to the accuracy of malware analysis results, these criteria are very decisive in determining the appropriate method for the analysis of a large volume of malware. This section discusses the methodology and experiment of each triaging method for the collected ransomware samples. The experiment is aimed at illustrating the similarity detection success rate of each triaging method for each ransomware category separately and collectively. It is expected and most probably that 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 triaging method.

A. Fuzzy Hashing: Methodology

When fuzzy hashing is applied on an unpacked ransomware sample, it generates a fuzzy hash value for that ransomware sample. This fuzzy hash value can be matched against either existing identified ransomware samples or their fuzzy hash values. If the fuzzy hash of a sample in question matches with any of the pre-identified ransomware samples or its fuzzy hash value then the fuzzy hash result is generated as a degree of similarity between the two. This fuzzy similarity result is presented in the range of 1% (least matched) to 100% (exactly matched), however, it is entirely at the discretion of security experts how they interpret this value depending on their analysis requirement. Generally, a threshold value can be set to accept or ignore the fuzzy similarity score to determine as matched or not matched scenario respectively. The fuzzy hashing should only be used as an initial investigation that may assist in any further analysis but not as a conclusive result [29].

B. Fuzzy Hashing: Experiment

In this experiment, the SSDEEP, SDHASH and mvHASH-B fuzzy hashing methods were used to detect similarity for each ransomware category separately. It was important to assess the performance of these three methods in different threshold conditions for comparison purposes; therefore, their similarity detection results were evaluated in four different conditions: 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 four evaluation results for four ransomware categories are presented in Tables I to IV. One of the most important findings in all four evaluation results is that the results of SDHASH and mvHASH-B fuzzy hashing methods decreased and in some cases quite significantly as the similarity threshold value increased. The detection rate of the SSDEEP fuzzy hashing method is lower, however, consistent in all four experiments [14], [30]. At the final similarity threshold limit of 30%, most SSDEEP results are superior to the other two fuzzy hashing methods. This finding is crucial when utilising these similarity results in further analysis as they can affect the next stage (e.g., clustering or classification) result significantly.

TABLE I
Similarity Detection Results of the SSDEEP, SDHASH and mvHASH-B Fuzzy Hashing Methods for WannaCry Ransomware Corpus

Fuzzy Hashing Matching Criteria for WannaCry Ransomware	**Similarity Detection Rate of SSDEEP**	**Similarity Detection Rate of SDHASH**	**Similarity Detection Rate of mvHASH-B**
Based on all Fuzzy Similarity Scores (1-100%)	91.2%	93.6%	90%
Based on Fuzzy Similarity Scores above the Threshold of 10%	91.2%	93.6%	90%
Based on Fuzzy Similarity Scores above the Threshold of 20%	91.2%	90%	84.4%
Based on Fuzzy Similarity Scores above the Threshold of 30%	90.8%	90%	84.4%

TABLE II
Similarity Detection Results of the SSDEEP, SDHASH and mvHASH-B Fuzzy Hashing Methods for Locky Ransomware Corpus

Fuzzy Hashing Matching Criteria for Locky Ransomware	**Similarity Detection Rate of SSDEEP**	**Similarity Detection Rate of SDHASH**	**Similarity Detection Rate of mvHASH-B**
Based on all Fuzzy Similarity Scores (1-100%)	42%	58.4%	72.4%
Based on Fuzzy Similarity Scores above the Threshold of 10%	42%	38.4%	64%
Based on Fuzzy Similarity Scores above the Threshold of 20%	41.6%	35.6%	36.4%
Based on Fuzzy Similarity Scores above the Threshold of 30%	41.6%	30.4%	33.6%

C. Import Hashing: Methodology

Similarly, when import hashing is applied on an unpacked ransomware sample, it generates an IMPHASH hash value for that ransomware sample. Moreover, this IMPHASH hash value can be matched against either existing identified ransomware samples or their IMPHASH hash values. If the IMPHASH hash matches with any of the pre-identified ransomware samples or its IMPHASH hash value then the result is generated as a matched sample with one or more samples. However, it does not provide a degree of similarity, rather a binary output (i.e.

TABLE III
SIMILARITY DETECTION RESULTS OF THE SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR CERBER RANSOMWARE CORPUS

Fuzzy Hashing Matching Criteria for Cerber Ransomware	Similarity Detection Rate of SSDEEP	Similarity Detection Rate of SDHASH	Similarity Detection Rate of mvHASH-B
Based on all Fuzzy Similarity Scores (1-100%)	33.6%	71.2%	94.8%
Based on Fuzzy Similarity Scores above the Threshold of 10%	33.6%	62.8%	90.4%
Based on Fuzzy Similarity Scores above the Threshold of 20%	33.6%	37.6%	36.8%
Based on Fuzzy Similarity Scores above the Threshold of 30%	33.6%	28.4%	36%

TABLE IV
SIMILARITY DETECTION RESULTS OF THE SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR CRYPTOWALL RANSOMWARE CORPUS

Fuzzy Hashing Matching Criteria for CryptoWall Ransomware	Similarity Detection Rate of SSDEEP	Similarity Detection Rate of SDHASH	Similarity Detection Rate of mvHASH-B
Based on all Fuzzy Similarity Scores (1-100%)	28%	52.4%	83.6%
Based on Fuzzy Similarity Scores above the Threshold of 10%	28%	32.8%	56.8%
Based on Fuzzy Similarity Scores above the Threshold of 20%	28%	24%	20.8%
Based on Fuzzy Similarity Scores above the Threshold of 30%	28%	20.4%	20.4%

either matched or not matched). The import hashing should only be used as an initial investigation that may help in any further analysis but not as a conclusive result [2].

D. Import Hashing: Experiment

In this experiment, the import hashing method was used to detect similarity for each ransomware category separately. The similarity detection results for all the four ransomware categories are shown in Table V. The import hashing result is a mixed result when compared with the fuzzy hashing results. In one case it is somewhat better however, in other cases it is slightly lower. It is worth noting that import hashing can only be used on PE file format, therefore, its effectiveness depends on the type of samples investigated.

E. YARA Rules: Methodology

The first two triaging methods are hashing methods and 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. It requires an in-depth analysis of malware and their family to generate YARA rule(s) for a specific malware or their family. Therefore, generation of effectual YARA rules demands effort

TABLE V
SIMILARITY DETECTION RESULTS OF IMPORT HASHING FOR WANNACRY, LOCKY, CERBER AND CRYPTOWALL RANSOMWARE SAMPLES

Detection Rate for Particular Ransomware Category	Similarity Detection Rate of Import Hashing
WannaCry Ransomware Samples	87.6%
Locky Ransomware Samples	31.6%
Cerber Ransomware Samples	61.6%
CryptoWall Ransomware Samples	27.2%

and expertise, unlike both hashing methods, where untrained personnel can apply the process of hash generation to generate hashes and perform the analysis. YARA rules can be generated manually or automatically, whilst automatic rule generation is easier than the manual process, however, it may require some post-processing operations to optimise them. Here *yarGen* tool [31] is employed to generate the YARA rules for all ransomware samples. This tool generates two types of rules ordinary rules and super rules depending on the malware sample types by utilising some intelligent techniques such as Fuzzy Regular Expressions, Naive Bayes Classifier and Gibberish Detector [32]. All the basic YARA rules generated for ransomware samples contain up to 20 strings based on their highest scores and do not include IMPHASH as it is employed as one of the triaging method in the paper.

F. YARA Rules: Experiment

In this experiment, after generating YARA rules for all four ransomware categories separately, they are used to detect similarity for each ransomware category separately. The similarity detection results for all four categories are shown in Table VI. The result of YARA rules is a mixed result when compared with the both hashing results, as in two cases it is slightly improved, and in others it is not. However, 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 [33]. Furthermore, if the number of strings and attributes are increased or decreased then it may change the analysis results. If the number of strings and attributes are significantly increased then it adversely affects the performance of YARA rules as malware analysis is always performed on a large sample size.

In summary, all three triaging methods performed slightly better and slightly worse, and it is difficult to determine the best triaging method for all given scenarios. Therefore, further analysis is required to improve the performance of the triaging methods. YARA rules are customisable and contain some advanced features, whereas fuzzy hashing can be a compact and add more value to YARA rules. Accordingly, this integration of YARA rules and fuzzy hashing would be examined in the next section of proposed enhanced YARA rules.

TABLE VI
SIMILARITY DETECTION RESULTS OF YARA RULES FOR WANNACRY, LOCKY, CERBER AND CRYPTOWALL RANSOMWARE SAMPLES

Detection Rate for Particular Ransomware Category	**Similarity Detection Rate of YARA Rules***
WannaCry Ransomware Samples	89.6%
Locky Ransomware Samples	54.4%
Cerber Ransomware Samples	77.2%
CryptoWall Ransomware Samples	27.6%

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.

V. MALWARE TRIAGING PROCESS USING THE PROPOSED ENHANCED YARA RULES

A. Enhanced YARA Rules: Methodology

The Indicator of Compromise (IoC) string is one of the most important parameters of YARA rules and the quantity of strings and how they are selected for a rule is crucial for its success. However, threat actors are equally intelligent and understand such mechanisms and attempt evasion by using perceptive modifications in their malware. If only few or none of the selected IoC strings are found in the targeted samples then YARA rules do not flag samples as malware even though they may be malware. Accumulating a large number of IoC strings in rules may increase the computational complexity and affect the performance of YARA rules significantly. Additionally, in order to write such complex YARA rules or modify automatically generated rules, a high degree of expertise is required in cyber security [4], [5], [6]. Consequently, it is essential to find an easy solution to make YARA rules more efficient without incurring all complexities stated earlier. This requires exploring alternative mechanisms other than IoC strings to enhance YARA rules. Fuzzy hashing is a compact, fast and resource-optimised mechanism employed for triaging which may not be effective on its own, nonetheless it can complement YARA rules enhancing its triaging performance without affecting complexity significantly [2]. Fuzzy hashing attempts to find structural similarity between the two files in their entirety, in circumstances where the selected IoC strings cannot be found in the sample. Additionally, fuzzy hashing can provide the degree of similarity of each matched sample alongside the outcome of YARA rules which is not achievable in YARA rules alone. Sometimes they can complement each other in finding a missed opportunity by one of the mechanisms. Thus, the combined search result can increase the accuracy and confidence level of the overall triaging process. The logical approach for the implementation of the proposed enhanced YARA rules is shown using the pseudocode in Algorithm 1 and Fig. 5.

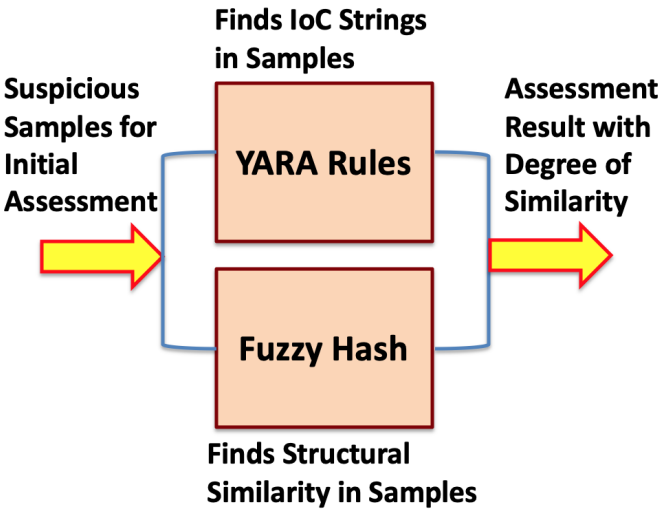

Fig. 5. Fuzzy Hashing Aided Enhanced YARA Rules

Algorithm 1: Pseudocode for the Proposed Fuzzy Hashing Aided Enhanced YARA Rules

$\mathbb{S}$, Set of Samples for Investigation
$\mathbb{R}$, Set of YARA Rules
\$, Set of Strings in a YARA Rule
$\mathbb{F}$, Set of Fuzzy Hashes of Known Malware
F, Fuzzy Hash Value
β_T, YARA String Count Threshold
δ_T, Fuzzy Hash Similarity Threshold
Δ, Degree of Similarity
C, Counter for Matched Strings
for $(i = 1; i < |\mathbb{S}|; i++)$ **do**
  for $(j = 1; j < |\mathbb{R}|; j++)$ **do**
    for $(k = 1; k < |\$|; k++)$ **do**
      if $\$_k \in \mathbb{S}_i$ **then**
        $C_{i,j}++$
    if $\Sigma_{k=1}^{|\$|} C_{i,j} \geq \beta_T \;\; OR \;\; \Delta(F_{\mathbb{S}_i}, F_l) \geq \delta_T \; [F_l \in \mathbb{F}]$ **then**
      return $YARA\ Rule$

B. Enhanced YARA Rules: Experiment

The enhanced YARA rules with fuzzy hashing were evaluated utilising three tried and tested fuzzy hashing approaches SSDEEP, SDHASH and mvHASH-B. This evaluation was to determine whether this integration was successful or not, and if successful, then which fuzzy hashing method produced greater accuracy in results. The similarity detection results of enhanced YARA rules utilising three different fuzzy hashing methods for all the four ransomware categories are shown in Table VII. Here, the fuzzy similarity scores greater than 30% were utilised for all the three fuzzy hashing methods. Noticeably, enhanced YARA rules with all the three fuzzy hashing methods showed a minor improvement, but SSDEEP fuzzy hashing contributed to the highest improvement in the overall result of triaging. Interestingly, for one ransomware

category Cerber, no fuzzy hashing could improve the result of YARA rules, however, in this case, YARA rules already produced better results than other methods, which can compensate the fuzzy hashing "miss outs". Alternatively, in all those categories where YARA could not produce respectable results, fuzzy hashing assisted improvement in the results, which is crucial for the success of this integration. On the basis of these experiments, the SSDEEP based result of YARA rules is recommended as the final results of enhanced YARA rules. Furthermore, SSDEEP is more compact, faster and a resource-optimised fuzzy hashing method in comparison to the SDHASH and mvHASH-B methods [14], [30].

TABLE VII
SIMILARITY DETECTION RESULTS OF ENHANCED YARA RULES UTILISING FUZZY HASHING SSDEEP, SDHASH AND MVHASH-B FOR WANNACRY, LOCKY, CERBER AND CRYPTOWALL RANSOMWARE SAMPLES

Detection Rate for Particular Ransomware Category	**Similarity Detection Rate of Enhanced YARA Rules based on SSDEEP (Similarity Score >30%)**	**Similarity Detection Rate of Enhanced YARA Rules based on SDHASH (Similarity Score >30%)**	**Similarity Detection Rate of Enhanced YARA Rules based on mvHASH-B- (Similarity Score >30%)**
WannaCry Ransomware Samples	93.2%	92.8%	92%
Locky Ransomware Samples	59.6%	58%	58.4%
Cerber Ransomware Samples	77.2%	77.2%	77.2%
CryptoWall Ransomware Samples	38.4%	34.8%	34.4%

C. Comparative Evaluation of the Triaging Results of Enhanced YARA Rules with Different Triaging Methods

Finally, the triaging results of enhanced YARA rules (SSDEEP fuzzy hashing based) are compared against the triaging results of all other triaging methods as shown in the Table VIII and Fig. 6. Evidently, basic YARA rules performed somewhat better than the other two triaging methods fuzzy hashing and import hashing. Nonetheless, enhanced YARA rules produced slightly better results (67.1%) than basic YARA rules results (62.2%). This improvement is not large but still indicates the moderate success of this integration of YARA rules and fuzzy hashing methods.

VI. CONCLUSION

This paper proposed fuzzy hashing aided enhanced YARA rules to improve the detection rate of YARA rules without significantly increasing the complexity and overheads of YARA rules. This proposed approach utilised an additional fuzzy hash value alongside basic YARA rules to complement each other when one method is unable to find a match and vice versa. This work employed three triaging methods fuzzy hashing, import hashing and YARA rules to perform extensive experiments on the collected ransomware samples. Finally, the detection rate of enhanced YARA rules was compared against the detection rate of the selected triaging methods, where enhanced YARA rules produced slightly better results for collected ransomware samples in comparison to all other methods. However, the success of this integration is dependent on the nature of malware samples as fuzzy hashing may work well in some cases but may not in others. The selected SSDEEP fuzzy hash is compact and does not increase the complexity of enhanced YARA rules significantly; however, other fuzzy hashing methods may increase the complexity of enhanced YARA rules, all these factors require further investigation in the future.

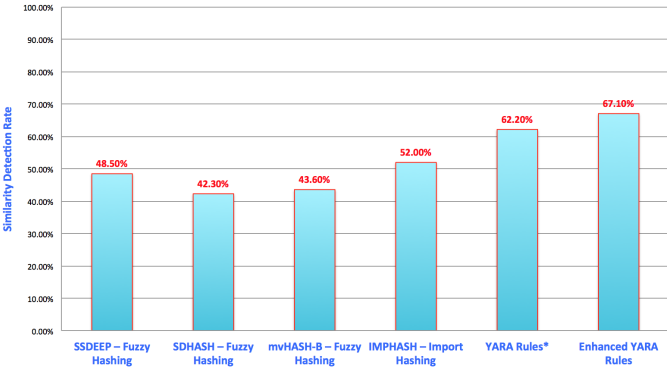

Fig. 6. Overall Similarity Detection Rate of SSDEEP, SDHASH, mvHASH-B, IMPHASH, YARA Rules and Enhanced YARA Rules for the collected Ransomware Corpus

ACKNOWLEDGEMENT

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

REFERENCES

[1] N. Naik, C. Shang, P. Jenkins, and Q. Shen, "D-FRI-Honeypot: A secure sting operation for hacking the hackers using dynamic fuzzy rule interpolation," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

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

[5] 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

[6] 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

[7] N. Naik, P. Jenkins, N. Savage, L. Yang, T. Boongoen, and N. Iam-On, "Fuzzy-Import Hashing: A malware analysis approach," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2020.

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

TABLE VIII
COMPARISON OF SIMILARITY DETECTION RESULTS OF ENHANCED YARA RULES WITH SSDEEP, SDHASH, MVHASH-B, IMPHASH AND YARA RULES FOR THE COLLECTED RANSOMWARE CORPUS

Detection Rate for Particular Ransomware Category	**SSDEEP-Fuzzy Hashing Similarity Detection Rate**	**SDHASH-Fuzzy Hashing Similarity Detection Rate**	**mvHASH-B-Fuzzy Hashing Similarity Detection Rate**	**IMPHASH-Import Hashing Similarity Detection Rate**	**Standard YARA Rules Similarity Detection Rate**	**Enhanced YARA Rules Similarity Detection Rate**
WannaCry Ransomware Samples	90.8%	90%	84.4%	87.6%	89.6%	93.2%
Locky Ransomware Samples	41.6%	30.4%	33.6%	31.6%	54.4%	59.6%
Cerber Ransomware Samples	33.6%	28.4%	36%	61.6%	77.2%	77.2%
CryptoWall Ransomware Samples	28%	20.4%	20.4%	27.2%	27.6%	38.4%

[9] N. Naik, P. Jenkins, N. Savage, L. Yang, K. Naik, and J. Song, "Embedding fuzzy rules with YARA rules for performance optimisation of malware analysis," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2020.

[10] ——, "Augmented YARA rules fused with fuzzy hashing in ransomware triaging," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.

[11] V. Alvarez. (2019) Writing YARA rules. [Online]. Available: https://yara.readthedocs.io/en/v3.4.0/writingrules.html

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

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

[14] 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.

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

[16] 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

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

[18] 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

[19] N. Naik, P. Jenkins, N. Savage, and L. Yang, "A computational intelligence enabled honeypot for chasing ghosts in the wires," *Complex & Intelligent Systems*, 2020.

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

[21] 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.

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

[23] K. Savage, P. Coogan, and H. Lau, "The evolution of ransomware - Symantec," pp. 1–57, 2015.

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

[25] Malwarebytes. (2019) Ransomware. [Online]. Available: https://www.malwarebytes.com/ransomware/

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

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

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

[29] 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.

[30] 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.

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

[32] ——. (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

[33] N. Naik, P. Jenkins, R. Cooke, J. Gillett, and Y. Jin, "Evaluating automatically generated YARA rules and enhancing their effectiveness," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020.