A Ransomware Detection Method Using Fuzzy Hashing for Mitigating the Risk of Occlusion of Information Systems

Nitin Naik[1], Paul Jenkins[1] and Nick Savage[2]

[1]Defence School of Communications and Information Systems, Ministry of Defence, United Kingdom
[2]School of Computing, University of Portsmouth, United Kingdom
Email: nitin.naik100@mod.gov.uk, paul.jenkins683@mod.gov.uk, nick.savage@port.ac.uk

***Abstract*—Today, a significant threat to organisational information systems is ransomware that can completely occlude the information system by denying access to its data. To reduce this exposure and damage from ransomware attacks, organisations are obliged to concentrate explicitly on the threat of ransomware, alongside their malware prevention strategy. In attempting to prevent the escalation of ransomware attacks, it is important to account for their polymorphic behaviour and dispersion of inexhaustible versions. However, a number of ransomware samples possess similarity as they are created by similar groups of threat actors. A particular threat actor or group often adopts similar practices or codebase to create unlimited versions of their ransomware. As a result of these common traits and codebase, it is probable that new or unknown ransomware variants can be detected based on a comparison with their originating or existing samples. Therefore, this paper presents a detection method for ransomware by employing a similarity preserving hashing method called fuzzy hashing. This detection method is applied on the collected WannaCry or WannaCryptor ransomware corpus utilising three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B to evaluate the similarity detection success rate by each method. Moreover, their fuzzy similarity scores are utilised to cluster the collected ransomware corpus and its results are compared to determine the relative accuracy of the selected fuzzy hashing methods.**

***Index Terms*—Ransomware; Similarity Preserving Hashing; Fuzzy Hashing; SSDEEP; SDHASH; mvHASH-B; K-Means Clustering; WannaCry; WannaCryptor.**

I. Introduction

Information systems have become one of the key functions within any organisation, with regard to its daily operation. A ransomware attack can completely occlude it by encrypting or locking out its entire data. This subset of malware encrypts files on the compromised system or network, however, there are a number of ransomware versions which erase files or blocks access to the system or the network itself. Ransomware attacks on organisational information systems have become very common and frequent. WannaCry or WannaCryptor ransomware is such an example, having emerged in the last five years, causing a total loss of around $4 billion to both organisations and individuals [1], [2]. In attempting to prevent the escalation of ransomware attacks, it is important to account for their polymorphic behaviour and dispersion of inexhaustible versions. However, a number of ransomware samples possess similarity as they are created by similar groups of threat actors. A particular threat actor or group often adopts some similar practices or codebase to create unlimited versions of their ransomware [3]. Consequently, it is possible to detect new or unknown ransomware by comparing with known samples of ransomware.

Several different methods are available to detect malware, however, they have their own strengths and limitations. A similarity preserving hashing or fuzzy hashing function has been used for malware analysis in the past, which can also be utilised to detect ransomware based on their similarity with other known or existing samples. Additionally, fuzzy hashing similarity scores can be used further for clustering or classification of samples into similar groups [4]. Therefore, this paper presents a detection method for ransomware by employing a fuzzy hashing technique such as SSDEEP, SDHASH or mvHASH-B. In this detection method, the collected WannaCry or WannaCryptor ransomware corpus is examined for similarity amongst the ransomware samples within the corpus utilising these fuzzy hashing methods to evaluate the similarity detection success rate by each method. Moreover, their fuzzy similarity scores are utilised to cluster the collected ransomware corpus and its results are compared to determine the relative accuracy of the selected fuzzy hashing methods.

The paper is divided into the following sections: Section II explains fuzzy hashing in general and its methods SSDEEP, SDHASH and mvHASH-B; Section III discusses the process of collecting WannaCry/WannaCryptor ransomware samples for analysing the above fuzzy hashing methods. Section IV presents the proposed detection method for ransomware using fuzzy hashing and clustering. Section V presents the experimental evaluation of SSDEEP, SDHASH and mvHASH-B fuzzy hashing methods and their similarity detection results. Section VI presents some of the main limitations of fuzzy hashing. Finally, Section VII concludes the paper with possible future enhancements.

II. Fuzzy Hashing and Different Fuzzy Hashing Methods

Cryptographic hashing is one of the most popular forms of hashing which can be used to identify a file exclusively (or duplicate files) to verify its integrity. Nonetheless, this hashing cannot be utilised in computer forensics as malware may be of a similar strain, having only changed a binary digit, rendering the file different from the original cryptographically. Therefore, the quest to discover malware files necessitates the use of a similarity preserving hash function, to determine similarity [5]. Fuzzy hashing is such a function capable of identifying similar files, producing a similarity score expressed as a percentage of their similarity. In this technique, generally, a file is divided into multiple blocks and a hash value is calculated for each block, finally, all hash values of the blocks are concatenated to generate the fuzzy hash value, this process is shown in Fig. 1. The length of the resulting fuzzy hash value depends on several factors such as the block size, the file size, and the output size of the selected hash function [6]. This is contrary to the cryptographic hash function, where the complete file is hashed contemporaneously, and the output has fixed size irrespective of the size of the input file. Fuzzy hashing techniques can be classified into different categories: Context-Triggered Piecewise Hashing (CTPH), Statistically-Improbable Features (SIF), Block-Based Hashing (BBH) and Block-Based Rebuilding (BBR) [7], [8], [9]. The similarity preserving property of fuzzy hashing is useful in forensic investigations to compare unknown files with known malware families based on their similarity, and to triage and cluster malware which use multiple variants from the same malware family that perform the exact same set of operations, but have different cryptographic hashes [3].

The similarity can be defined either as syntactic similarity or semantic similarity [3]. The syntactic similarity between the two files can be determined based on the byte structure of the files (i.e. raw byte sequences of data) and does not consider the interpretation of the data, whereas, the semantic similarity between the two files can be determined based on the interpretation and context of the data and does not consider the byte structure of the files [3]. Commonly, similarity hashing or fuzzy hashing techniques work on a syntactic level, without considering the interpretation and context of the data.

A. SSDEEP

SSDEEP is one of the most popular fuzzy techniques based on a spam detector called spamsum [5]. The algorithm divides an input file into blocks of variable size, randomly, with the division based on the content of that file [3]. The rolling hash is used to determine block boundaries (also known as trigger points) in the file depending on the content of sections of seven bytes at a time according to a predefined criteria, based upon the the Adler32 function [6]. Subsequently, SSDEEP calculates the hash value of each block separately and produces the final SSDEEP hash value by concatenating all the hashes into one hash. This method ensures that two similar files will have similar block boundaries and similar SSDEEP hash values. This method employs an edit distance algorithm based on the Damerau-Levenshtein algorithm used in a number of applications. The algorithm compares two suspected similar blocks calculating the minimum number of operations required to transform one block into the other, using a combination of operations including insertion, deletion, and substitution of a single character, and transpositions of two adjacent characters [3].

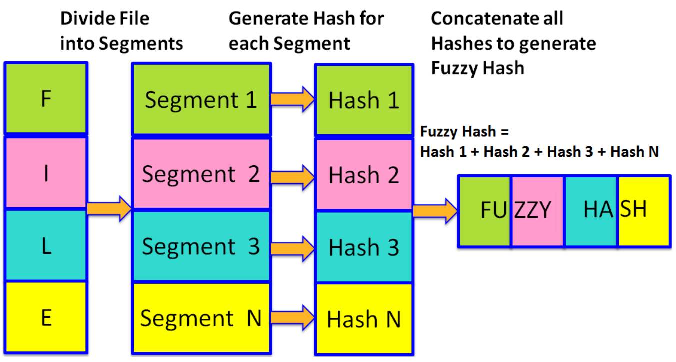
Divide File into Segments
Generate Hash for each Segment
Concatenate all Hashes to generate Fuzzy Hash
F
I
L
E
Segment 1
Segment 2
Segment 3
Segment N
Hash 1
Hash 2
Hash 3
Hash N
Fuzzy Hash = Hash 1 + Hash 2 + Hash 3 + Hash N
FU ZZY HA SH

Fig. 1. Generation of Fuzzy Hash Value in Fuzzy Hashing Method

B. SDHASH

SDHASH, is a relatively different fuzzy technique to that of SSDEEP, presented by Vassil Roussev in 2010, it is based on the concept of statistically-improbable features [10]. The premise is that, any file consists of several statistical features, where some are rare (statistically-improbable) and some are very common [3]. Therefore, similar files will probably have the same rare features, while dissimilar files will probably have different rare features. The more common rare features, probably the more similar files are. Commonly, a SDHASH feature is a 64-byte string. Instead of dividing a file into blocks, SDHASH identifies rare features using an entropy calculation, hashing the selected features with the cryptographic hash function SHA-1, storing them in multiple Bloom filters [11]. A Bloom filter is a space-efficient probabilistic data structure, and a maximum of 128 features can be stored in one Bloom filter and the number of Bloom filters is dependent on the total number of features [3]. The concatenation of the resulting Bloom filters constitutes the final SDHASH fuzzy hash value. While comparing the two files for similarity, SDHASH employs Hamming distance for faster comparison [3].

C. mvHASH-B

There are similarities between mvHASH-B and SDHASH fuzzy hashing techniques, with the latter based upon a different design approach. Its complete hash generation process is divided into three stages: input transformation using the concept of majority votes, encoding of majority votes using the Run-Length Encoding (RLE) approach, and hash generation using a data structure called a Bloom filter [12]. In the first stage, the input is transformed into a byte sequence of 0s and 1s based on the majority of votes, where the majority of bits in the pre-determined neighbourhood is equal to zero, the neighbourhood is represented by a 0, otherwise by a 1 [12]. This ensures that a minor change in the input byte sequence does not affect the majority votes, as the majority of bits remain unchanged. Therefore, the fuzzy hash value does not change and similarity is preserved. In the second stage, the RLE compression approach is used to represent each sequence of 0s and 1s by its length (in bytes). It reduces the length of the input byte sequence by counting the amount of identical consecutive bytes and returning this number. In the third stage, the RLE encoded string is stored in Bloom filters [12]. A Bloom filter is a data structure enabling rapid operations such as comparisons. It stores components in a way which is independent from the original string. Consequently, the comparison algorithm may use Hamming distance for faster operations as the calculation of Hamming distance only requires the low-level operations XOR and bit count.

The generated mvHash-B hash value is stored in one or more Bloom filters. While comparing two mvHash-B hashes, each Bloom filter of the first hash value is compared against each Bloom filter of the second hash value. The similarity value/distance is calculated using the Hamming distance at bit-level between the Bloom filters. Where the Hamming distance signifies the total number of bits that differ between the two compared Bloom filters (i.e. the lower the Hamming distance, the increased probability that the two Bloom filters or Hashes are similar). The mvHASH-B uses its own hash function which is comparable to SHA-1 algorithm.

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 to extort money [13], [14], [15]. It causes loss of money and reputational damage to the business and sometimes potentially permanent loss of data. There are several ransomware categories in existence, however, some of are more relevant in terms of their attacks, ransom amount and geography and hence for investigation. Initial research of different ransomware revealed that the WannaCry or WannaCryptor ransomware is one of the most significant variants of ransomware in recent times and is selected for this study [2], [16], [17], [18]. 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 samples of WannaCry or WannaCryptor ransomware which could be investigated manually. A total of 112 WannaCry or WannaCryptor ransomware samples were collected and verified. All the WannaCry samples were collected from two sources *Hybrid Analysis* [19] and *Malshare* [20] with the majority of the analysis performed based on the information acquired by *VirusTotal* [21]. The main difficulty was to verify the credibility of the collected ransomware samples that they were very likely to be WannaCry ransomware samples. The criteria for the credibility of that sample was determined by a VirusTotal detection engine score of 40 or greater, i.e. at least 40 well-known engines identified the sample as ransomware/malware. To verify that they were WannaCry or WannaCryptor ransomware, their type was manually verified from all the identified detection engines, 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 [22], [23], [24], [25], [26]. 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. Proposed Ransomware Detection Method Using Fuzzy Hashing

This proposed detection method is mainly based on the fuzzy hashing method and the evaluation of different fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B to obtain precise results. In this proposed method, for verifying a new sample whether it is ransomware or not, 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 a classification operation to conclude as a meaningful result. However, the preciseness of the clustering or classification is mainly dependent on the similarity scores of the fuzzy hashing method; therefore, selecting an accurate fuzzy hashing method is a crucial step in this ransomware analysis process. In this detection method, a k-means clustering analysis is performed by utilising the similarity scores generated by the three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B to evaluate their preciseness based on the best clustering outcome. The main benefits of utilising the fuzzy hashing method for detecting ransomware is: compact size of fuzzy hash, small amount of memory, fewer overheads and less processing time [3]. All these benefits suggests that the fuzzy hashing method could be a suitable static analysis method for ransomware if it can produce precise results.

V. Experimental Evaluation of Different Fuzzy Hashing Methods SSDEEP, SDHASH and mvHASH-B

A. Comparative Evaluation of the Similarity Detection Results of SSDEEP, SDHASH and mvHASH-B

Initially, the three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B were applied on the collected WannaCry ransomware corpus to evaluate the similarity detection

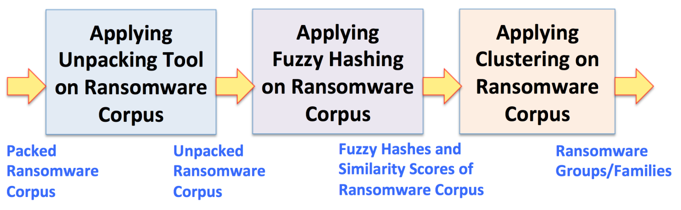
Applying Unpacking Tool on Ransomware Corpus
Applying Fuzzy Hashing on Ransomware Corpus
Applying Clustering on Ransomware Corpus
Packed Ransomware Corpus
Unpacked Ransomware Corpus
Fuzzy Hashes and Similarity Scores of Ransomware Corpus
Ransomware Groups/Families

Fig. 2. A Ransomware Detection Method Utilising Fuzzy Hashing Method

success rate by each 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 the majority of the SSDEEP similarity scores are above the set highest threshold of 30% resulting in its lowest matched samples of 103. In the 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 is very important in utilising relatively consistent fuzzy hashing method for further clustering of these ransomware samples precisely to reveal crucial information about the collected ransomware corpus.

B. Comparative Evaluation of K-Means Clustering Results based on SSDEEP, SDHASH and mvHASH-B Similarity Scores

To assess the effects of fuzzy hashing similarity scores on clustering, k-means clustering is employed using reliable similarity scores of SSDEEP, SDHASH and mvHASH-B above the set highest threshold of 30%. The clustering results are assessed using three evaluation indexes Dunn Index, Silhouette Index and Connectivity Index as shown in three Tables II to IV respectively. The value of each index is calculated for a cluster size 2 to 6 to assess a consistent and improved value of each index. However, to determine the relatively better fuzzy hashing method based on the clustering results, its combined value of three indexes is compared for an optimal cluster size of 2 as shown in Table V. The optimal cluster size is determined manually after analysing all the collected ransomware samples. Based on the combined value of three selected indexes, Table V shows the overall SSDEEP clustering result is relatively better than the other two methods SDHASH and mvHASH-B. This reflects the importance of similarity scores produced by each fuzzy hashing as it determines the success of the clustering method. Therefore, if a suitable fuzzy hashing is selected, the clustering can provide relatively better results, which can be further utilised in developing fuzzy rules for the fuzzy rule-based systems [27], [28], [29], [30], [31], [32].

VI. Limitations of Fuzzy Hashing

The application of fuzzy hashing in detecting ransomware has demonstrated good results for this particular WannaCry/WannaCryptor ransomware corpus. However, it is also important to consider some of the main limitations of fuzzy hashing, which are as follows:

- The similarity score generated by any fuzzy hashing method is always difficult to interpret.
- Any similarity score is intuitively judged by the security expert, which can lead to very different interpretations between security experts.
- Fuzzy hashing methods mostly effective on a syntactic level and check structural similarity but do not consider semantic level i.e., behavioural similarity (context of the data).
- Many fuzzy hashing methods (e.g. SSDEEP) are dependent on the block sizes and the overall size of the file for hashes. This can be easily evaded by appending data to the end of the file, in which header and section data are still identical.
- Bloom filters based fuzzy hashing methods (e.g. SDHASH) do not generate false negatives, however, false positives are possible.
- Most fuzzy hashing methods are severely affected by packers and unable to detect similarity in packed files.

VII. Conclusion

This paper proposed an efficient detection method for ransomware utilising a fuzzy hashing method. This detection method was applied on the collected WannaCry/WannaCryptor ransomware corpus utilising three fuzzy hashing methods SSDEEP, SDHASH and mvHASH-B to evaluate the similarity detection success rate by each method. All three fuzzy hashing methods produced very high similarity detection results (above 91%) for this particular ransomware corpus. Further, their fuzzy similarity scores were utilised to cluster the collected ransomware corpus by employing k-means clustering and its results were compared to determine the relative accuracy of the selected fuzzy hashing methods. The clustering results were evaluated through the use of three internal evaluation indexes; Dunn Index, Silhouette Index and Connectivity Index, to determine the accuracy and consistency of their clustering results, thus, the accuracy of fuzzy hashing results. The SSDEEP fuzzy hashing method produced relatively better results than other two methods SDHASH and mvHASH-B. Subsequently, the best analysis results can be used for both advanced static and dynamic analysis of ransomware. This proposed method is a fast, efficient and easy method to quickly perform the initial

TABLE I
SIMILARITY DETECTION RESULTS FOR THE SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR WANNACRY/WANNACRYPTOR RANSOMWARE CORPUS

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
DUNN INDEX EVALUATION RESULTS FOR THE K-MEANS CLUSTERING METHOD BASED ON THE SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR WANNACRY/WANNACRYPTOR RANSOMWARE CORPUS

Clustering Method	**Cluster Size=2**	**Cluster Size=3**	**Cluster Size=4**	**Cluster Size=5**	**Cluster Size=6**
SSDEEP based k-means Clustering	0.8348	0.7653	0.7074	0.7658	0.6628
SDHASH based k-means Clustering	0.7073	5.7587	0.1978	0.3489	0.0965
mvHASH-B based k-means Clustering	0.7061	0.7072	0.8079	0.7068	0.8243
Note: The Dunn Index has a value between *Zero* and ∞, where the greater value of the Dunn Index represents more accurate clustering results [33].					

TABLE III
SILHOUETTE INDEX EVALUATION RESULTS FOR THE K-MEANS CLUSTERING METHOD BASED ON THE SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR WANNACRY/WANNACRYPTOR RANSOMWARE CORPUS

Clustering Method	**Cluster Size=2**	**Cluster Size=3**	**Cluster Size=4**	**Cluster Size=5**	**Cluster Size=6**
SSDEEP based k-means Clustering	0.8575	0.8474	0.8443	0.8496	0.8611
SDHASH based k-means Clustering	0.9058	0.9112	0.6782	0.6331	0.4385
mvHASH-B based k-means Clustering	0.8654	0.868	0.8735	0.8684	0.8738
Note: The Silhouette Index has a value between the interval $[-1, 1]$, where the greater value of the Silhouette Index represents more accurate clustering results [34].					

TABLE IV
CONNECTIVITY INDEX EVALUATION RESULTS FOR THE K-MEANS CLUSTERING METHOD BASED ON THE SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR WANNACRY/WANNACRYPTOR RANSOMWARE CORPUS

Clustering Method	**Cluster Size=2**	**Cluster Size=3**	**Cluster Size=4**	**Cluster Size=5**	**Cluster Size=6**
SSDEEP based k-means Clustering	2.929	5.8579	11.7159	11.7159	15.5738
SDHASH based k-means Clustering	5.8579	5.8579	10.7627	15.9897	31.529
mvHASH-B based k-means Clustering	8.7869	8.7869	8.7869	14.6448	14.6448
Note: The Connectivity Index has a value between *Zero* and ∞, where the smaller value of the Connectivity Index represents more accurate clustering results [35].					

TABLE V
COMPARISON OF CLUSTERING RESULTS OF SSDEEP, SDHASH AND MVHASH-B FUZZY HASHING METHODS FOR AN OPTIMAL CLUSTERING SIZE=2

Clustering Method	**Dunn Index**	**Silhouette Index**	**Connectivity Index**
SSDEEP based k-means Clustering	0.8348	0.8575	2.929
SDHASH based k-means Clustering	0.7073	0.9058	5.8579
mvHASH-B based k-means Clustering	0.7061	0.8654	8.7869

triaging of ransomware samples. In future, it is essential to evaluate the proposed detection method on a larger sample of WannaCry/WannaCryptor ransomware and on other types of ransomware.

ACKNOWLEDGEMENT

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

REFERENCES

[1] S. Cobb. (2018) RANSOMWARE: An enterprise perspective. [Online]. Available: https://www.eset.com/us/business/resources/white-papers/ransomware-an-enterprise-perspective/

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

[3] 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.

[4] ——, "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.

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

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

[7] 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

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

[9] 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

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

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

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.

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

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

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

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] K. Khattar, "Detecting homologies in encrypted and unencrypted documents using fuzzy hashing," Jun. 12 2018, US Patent 9,996,603.

[28] N. Naik, P. Jenkins, N. Savage, and V. Katos, "Big data security analysis approach using computational intelligence techniques in R for desktop users," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

[29] N. Naik, C. Shang, Q. Shen, and P. Jenkins, "Intelligent dynamic honeypot enabled by dynamic fuzzy rule interpolation," in *The 4th IEEE International Conference on Data Science and Systems (DSS-2018)*. IEEE, 2018, pp. 1522–1529.

[30] ——, "Vigilant dynamic honeypot assisted by dynamic fuzzy rule interpolation," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018.

[31] N. Naik, R. Diao, C. Shang, Q. Shen, and P. Jenkins, "D-FRI-WinFirewall: Dynamic fuzzy rule interpolation for windows firewall," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6.

[32] N. Naik, R. Diao, and Q. Shen, "Dynamic fuzzy rule interpolation and its application to intrusion detection," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, pp. 1878–1892, 2018.

[33] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[34] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[35] G. Brock, V. Pihur, S. Datta, S. Datta *et al.*, "clValid, an R package for cluster validation," *Journal of Statistical Software (Brock et al., March 2008)*, 2011.