Cyberthreat Hunting - Part 1: Triaging Ransomware using Fuzzy Hashing, Import Hashing and YARA Rules

Nitin Naik[1], Paul Jenkins[1], Nick Savage[2] and Longzhi Yang[3]

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

***Abstract*—Ransomware is currently one of the most significant cyberthreats to both national infrastructure and the individual, often requiring severe treatment as an antidote. Triaging ransomware based on its similarity with well-known ransomware samples is an imperative preliminary step in preventing a ransomware pandemic. Selecting the most appropriate triaging method can improve the precision of further static and dynamic analysis in addition to saving significant time and effort. Currently, the most popular and proven triaging methods are fuzzy hashing, import hashing and YARA rules, which can ascertain whether, or to what degree, two ransomware samples are similar to each other. However, the mechanisms of these three methods are quite different and their comparative assessment is difficult. Therefore, this paper presents an evaluation of these three methods for triaging the four most pertinent ransomware categories WannaCry, Locky, Cerber and CryptoWall. It evaluates their triaging performance and run-time system performance, highlighting the limitations of each method.**

***Index Terms*—Ransomware; Triaging; Similarity Preserving; Fuzzy Hashing; SSDEEP; SDHASH; Import Hashing; IMPHASH; YARA Rules; WannaCry; WannaCryptor; Locky; Cerber; CryptoWall; Context-Triggered Piecewise Hashing.**

I. Introduction

Ransomware is currently one of the most dominant forms of malware affecting major institutions and the public, with its threat and ransom amount rapidly escalating over time. Additionally, ransomware threat actors are developing limitless versions or variants of their ransomware by reusing their codebase to perform the same set of operations [1]. The majority of ransomware encrypts files on the compromised system or network, however, a few ransomware erases files or block access to the system or network [2]. Several categories of ransomware are affecting users such as WannaCry/WannaCryptor, Locky, Cerber, CryptoWall, Petya, Notpetya, GandCrab, Bad Rabbit and CryptoLocker [2], [3], [4]. Triaging ransomware by finding the similarity with well-known ransomware samples is a critical initial step in saving significant time and effort, to enable further advanced analysis to be undertaken promptly.

Cryptographic hashing methods are useful to identify a unique file, based on its unique hash/digest. Nonetheless, it cannot be used to determine similarities between two almost similar files as their cryptographic hashes are quite different [5]. In digital forensics, there are several similarity metrics available to identify whether, or to what degree, two malware binaries are similar to each other [6]. Some of the most popular and proven triaging methods are fuzzy hashing, import hashing and YARA rules. Using these methods, security experts can determine whether a suspected malicious file bears any resemblance to already verified malicious files and to triage them accordingly, which is one of the most effective ways to assess numerous malicious samples swiftly.

The main issue in the triaging process is, how accurately a method can match the sample, and what is the system performance or complexity of that method. Generally, a method is applied to a large number of samples for their assessment; therefore, selecting a method is a balance between the accuracy and the system performance of that method. This paper investigates this issue by applying three of the most popular triaging methods used in digital forensics, namely: import hashing, fuzzy hashing and YARA rules, as applied in ransomware. An evaluation of the accuracy and performance is presented of their corresponding methods (import hashing- IMPHASH, fuzzy hashing- SSDEEP and SDHASH and YARA rules) by applying them to the four most pertinent categories of ransomware WannaCry, Locky, Cerber and CryptoWall. All four methods are evaluated based on their triaging performance, ascertaining the similarity within the same ransomware category and additionally across the other three categories. Furthermore, the run-time performance of each method is compared during the experiments, building a comparative assessment of three completely different triaging methods.

The paper is divided into the following sections: Section II explains fuzzy hashing - SSDEEP and SDHASH methods, import hashing- IMPHASH method and YARA rules. Section III discusses the process of collecting ransomware samples for this experimentation. Section IV presents the experimental analysis of IMPHASH, SSDEEP, SDHASH and YARA rules. Section V discusses the main limitations of import hashing,

fuzzy hashing and YARA rules. Finally, Section VI concludes the paper with the possible future enhancement and brief discussion about the second part of the paper [7].

II. Background Information

A. Fuzzy Hashing

The comparison of files to ensure its uniqueness can be performed utilising cryptographic hashing and hence can be used to locate duplicate files, although it is mostly used to verify the integrity of data. However, this technique cannot be applied in computer forensics as malware may be of a similar strain, having only changed a binary digit, nonetheless this renders 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 [8]. Fuzzy hashing is such a function capable of identifying similar files, producing a similarity score expressed as a percentage of their similarity.

Fuzzy hashing techniques divide a file into multiple blocks, calculating a hash value for each block [9]. The final fuzzy hash value is produced by concatenating all these hashes (see 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. 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) [10], [11], [12]. 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 performing the exact same set of operations, but have different cryptographic hashes.

The similarity can be defined either as syntactic similarity or semantic similarity [13]. 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. Commonly, similarity hashing or fuzzy hashing techniques work on a syntactic level, without considering the interpretation and context of the data.

1) SSDEEP: SSDEEP is one of the most popular fuzzy techniques based on a spam detector called spamsum [8]. The algorithm divides an input file into blocks of variable size (random blocks), based on the content of that file. A 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 Adler32 function [9]. Subsequently, SSDEEP calculates the hash value of each block separately

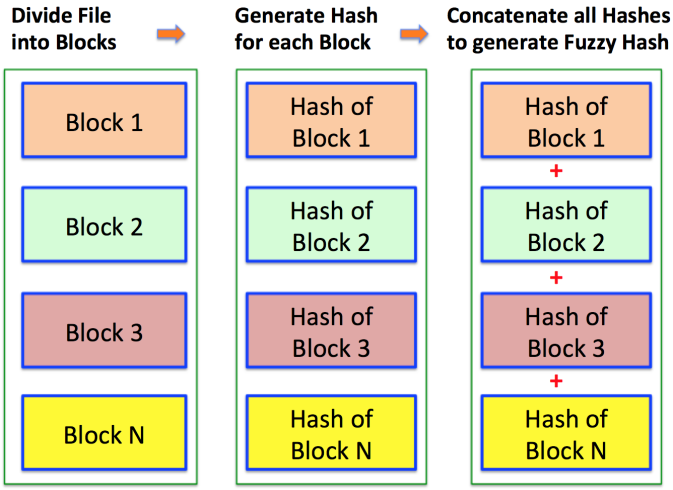

Fig. 1. Fuzzy Hash Generation

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.

2) SDHASH: SDHASH is a relatively different fuzzy technique from SSDEEP, presented by Vassil Roussev in 2010, based on the concept of statistically-improbable features [14]. The premise is that, any file consists of several statistical features, where some are rare (statistically-improbable) and some are very common. 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 [15]. A Bloom filter is a space-efficient probabilistic data structure, and has a maximum of 128 features that can be stored in one Bloom filter and the number of Bloom filters is dependent on the total number of features. 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.

B. Import Hashing - IMPHASH

Import hashing is a hashing technique used to detect the similarity between two files/malware based on the hash generated from the Import Address Table (IAT) of a Portable Executable (PE) file [16]. Imports are the functions that a piece

of software (e.g. malware) calls from other files (typically various DLL, EXE, SYS files that provide functionality to the Windows operating system). Import hashing generates a hash called IMPHASH, which is created based on the library/API names and their specific order within the executable file. Both the order of functions in the original source code and the order of source files at compile time affect the generation of the resulting IAT and thus, the IMPHASH value. Therefore, two files were compiled from the same source code will result the same IAT and thus, same IMPHASH value. Similarly, if the source code of the two files is not organized in the same way it will result in different IATs and thus, different IMPHASH values. Import hashing is comparable with fuzzy hashing with respect to speed, computation, complexity, memory and hash size. However, unlike fuzzy hashing which offers the degree of similarity, import hashing can only determine the binary similarity, i.e. whether the two files are similar or not.

C. YARA Rules

YARA tool is a pattern matching *swiss knife* developed to discover and classify malware, mainly for the research community [17]. It provides an easy and efficient method of writing custom rules (YARA rules), containing descriptions of targeted malware based on strings or byte sequences found in malware, utilising these rules to discover malicious files or processes. YARA syntaxes and semantics closely resemble the C programming language [17]. It can be used through its command-line interface or Python scripts with the *yara-python* extension. YARA rules are a flexible and powerful way to dispense with the rapidly growing problem of malware, furthermore, it supports all major operating systems Windows, Linux and Mac OS X [18].

III. Collection of Ransomware Samples

Ransomware attacks are growing more common and they are more expensive to remediate than their counterpart Denial of Service (DoS) attacks, which can be used as a smoke-screen for ransomware infiltration [19], [20], [21]. There are several ransomware categories in existence, however, some are most pertinent in terms of their attacks, ransom amount and geography, and hence for investigation. Initial research of different ransomware revealed that the four most significant ransomware categories are WannaCry/WannaCryptor, Locky, Cerber and CryptoWall and are selected for this study [1], [2], [3], [4]. The hardest task, with the highest work intensity was the collection of credible samples of these four types of ransomware due to the large number of fake or vague ransomware samples. As a result of this process, it was decided for the initial and manual investigation to collect 50 samples of each ransomware category, a total of 200 ransomware samples for all four categories. All the samples were collected from two sources *Hybrid Analysis* [22] and *Malshare* [23] with the majority of the analysis performed based on the information obtained from *VirusTotal* [24]. A significant issue in determining the credibility of samples and verification is that they are members of the correct category of ransomware. 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 a ransomware/malware. To verify their classification into the correct category, their type was manually verified from all the identified detection engines, where some of the detection engines identified the particular sample as a specific type of ransomware such as WannaCry/WannaCryptor, Locky, Cerber and CryptoWall. Nonetheless, this process was not straightforward, and mostly based on the discretion of the authors [25], [26], [27], [28]. This whole process of collecting and manually analysing samples was time consuming, finally, the 50 samples of each category were collected.

IV. Experimental Analysis of IMPHASH, SSDEEP, SDHASH and YARA Rules

All four methods IMPHASH, SSDEEP, SDHASH and YARA rules were evaluated for their ability to discover similarity in the samples, therefore, they were assessed against the 200 samples of four categories of ransomware WannaCry, Locky, Cerber and CryptoWall (with 50 samples of each category). There is an assumed behavioural similarity among all 200 samples of the four ransomware categories as they are all ransomware, however, their structure may be completely different from each other. Furthermore, there may be structural similarity within the 50 samples of the same category of ransomware. Therefore, for evaluating the triaging performance of each method, it is applied to find the similarity within the same ransomware category and across all the other three categories. As previously discussed, all the 200 samples were meticulously verified as a strong ransomware sample, therefore, the main focus of this experiment was to check how many samples are either matched or not by each method.

The first three methods IMPHASH, SSDEEP, SDHASH are hashing methods (generating hashes for files), consequently, they are directly comparable. However, YARA rules are somewhat different, having a different way of finding similarity amongst files, with the caveat that they can be generated in different ways either manual or automatic, with different sets of attributes and therefore, their performance may vary. In this experiment, the applied YARA rules are generated by *yarGen* tool [29]. It is based on Fuzzy Regular Expressions, Naive Bayes Classifier and Gibberish Detector [30] with the default setting of the top 20 strings based on their score and without IMPHASH, as IMPHASH is considered as a separate method in this paper. To ensure that YARA rules are comparable with the other three methods, individual rules are tested alongside super rules.

Table I shows the similarity detection and comparison results of all four methods. Here, IMPHASH matched 112 samples with at least one other sample in the same category. Similarly, SSDEEP matched 110 samples with at least one other sample in the same category. Likewise, SDHASH matched 138 samples with at least one other sample in the same category. Lastly, YARA rules matched 129 samples with at least one other sample in the same category. Their overall

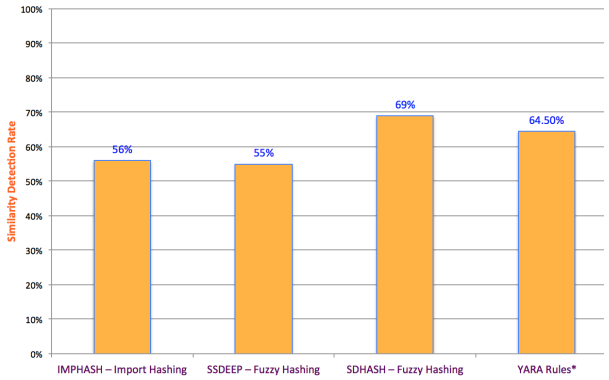

Fig. 2. Overall Similarity Detection Rate of IMPHASH, SSDEEP, SDHASH and YARA Rules for WannaCry, Locky, Cerber and CryptoWall Ransomware

similarity detection rate is shown in Fig. 2, where SDHASH is relatively better than other three methods for the selected ransomware samples. SDHASH generated better results for the three categories WannaCry, Locky and CryptoWall, however, YARA rules generated slightly better results for the Cerber ransomware category.

The second evaluation was to determine their triaging performance across different ransomware categories, which was somewhat complex due to the behavioural similarity amongst different ransomware categories. The two methods IMPHASH and SSDEEP were unable to find any correlation across the ransomware categories. Whereas, SDHASH and YARA rules were able to find only a few correlations as shown in Table I. SDHASH was again relatively better than YARA rules and found 37 matching samples in three categories Locky, Cerber and CryptoWall, whereas YARA rules found 25 matching samples in two categories Locky and Cerber. However, this cross-category comparison was to further check the effectiveness of these methods.

Based on all the experiments, the system performance of all four methods IMPHASH, SSDEEP, SDHASH and YARA rules are also evaluated as shown in Table II. IMPHASH has outperformed all other three methods with respect to runtime efficiency, memory and hash/rule size; therefore, it is the fastest and an efficient way of similarity comparison of files. However, it should be noted that is has a major dependency on the Import Address Table (IAT) and specific order of library/API functions. The system performance of SSDEEP fuzzy hashing method was very close to IMPHASH, however, its effectiveness is limited to simple structural modifications in a file. The system performance of another fuzzy hashing method SDHASH was relatively slower than SSDEEP despite its best similarity performance. Finally, the system performance of YARA rules was relatively the slowest amongst all compared methods despite its impressive performance. It should be noted that the system performance of YARA rules is dependent on several factors such the type of rules, number of rules and number attributes per rule.

TABLE I
SIMILARITY DETECTION AND COMPARISON RESULTS OF IMPHASH, SSDEEP, SDHASH AND YARA RULES FOR WANNACRY, LOCKY, CERBER AND CRYPTOWALL RANSOMWARE SAMPLES

Similarity Detection and Comparison of Ransomware Samples	**IMPHASH-Import Hashing**	**SSDEEP-Fuzzy Hashing**	**SDHASH-Fuzzy Hashing**	**YARA Rules***
Total WannaCry samples matched with at least one other WannaCry sample	46	46	47	46
Total Locky samples matched with at least one other Locky sample	17	21	29	28
Total Cerber samples matched with at least one other Cerber sample	32	25	36	39
Total Locky samples matched with at least one other CryptoWall sample	17	18	26	16
WannaCry matched with Locky, Cerber and CryptoWall	None	None	None	None
Locky matched with WannaCry, Cerber and Cryptowall	None	None	20 Cerber and 8 CryptoWall	22 Cerber
Cerber matched with WannaCry, Locky and Cryptowall	None	None	2 Locky	3 Locky
CryptoWall matched with WannaCry, Locky and Cerber	None	None	7 Locky	None

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.

V. LIMITATIONS OF IMPORT HASHING, FUZZY HASHING AND YARA RULES

A. Limitations of Import Hashing

- Simply changing the order of library/API functions within the file will result a different Import Address Table (IAT) and thus, different IMPHASH values.
- Import hashing may not work for those files which have a very few Import APIs.
- Import hashing is only effective on Portable Executable (PE) file format.
- Import hashing can only determine the binary similarity, i.e., whether the two files are similar or not.
- Import hashing methods mostly effective on a syntactic level and check structural similarity but does not consider semantic level i.e., behavioural similarity (context of the data).
- Import hashing is severely affected by packers and may not be able to detect similarity of packed files correctly.

B. Limitations of Fuzzy Hashing

- The similarity score generated by any fuzzy hashing method is always difficult to interpret.

TABLE II
PERFORMANCE COMPARISON OF IMPHASH, SSDEEP, SDHASH AND YARA RULES FOR WANNACRY, LOCKY, CERBER AND CRYPTOWALL RANSOMWARE SAMPLES

Performance Criteria	**IMPHASH-Import Hashing**	**SSDEEP-Fuzzy Hashing**	**SDHASH-Fuzzy Hashing**	**YARA Rules***
Matching Criteria	Library/API names and their specific order	Byte structure of the files	Byte structure of the files	Textual or binary patterns
Run-Time Efficiency	4 times faster than SSDEEP	3 times faster than SDHASH	2 times faster than YARA Rules	2 times slower than SDHASH
Hash/Rule Size	Smallest	Smaller than SDHASH	Greater than SSDEEP	Dependent on the type of rule and number of strings
Hash/Rule Generation	Programmed	Programmed	Programmed	Programmed/ Manual
Minimum File Size	None	4 KB	512 Bytes	None
Limitation/ Dependency	Dependent on IAT (Import Address Table)	Effective only when modifications are simple and mechanical	Effective only when modifications are simple and mechanical	Dependent on the selected attributes
Applications	VirusTotal, Totalshare, Peframe and Pefile Module	VirusTotal, Shadowserver, It is the de facto fuzzy hashing standard	Autopsy Forensic Browser of the Sleuth Kit	VirusTotal Intelligence, SpamStopsHere, Symantec, Tanium, The DigiTrust Group, ThreatConnect, Thug, Trend Micro, VMRay, etc.

- 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.

C. Limitations of YARA Rules

- Extracting UNICODE and ASCII strings from malware samples is a tedious task and requires experience and expertise.
- YARA rules can be generated automatically, however, generated YARA rules may not be as good as manually created ones.
- Most automatically generated YARA rules still require manual post-processing for optimisation.
- Ineffective YARA rules may generate many false positives.
- Public repositories of YARA rules may not be fit for the purpose or targeted operation.
- Basic rules looking for a combination of predetermined/unique strings may become ineffective and extra overheads if those unique strings are changed in the new/similar malware.

VI. CONCLUSION

This paper has presented an evaluation of the three most popular and proven triaging methods: fuzzy hashing, import hashing and YARA rules for triaging ransomware. These experiments ascertained whether, or to what degree, two ransomware samples are similar to each other. The paper has evaluated four methods IMPHASH -import hashing, SSDEEP and SDHASH -fuzzy hashing and YARA rules against the 200 samples of the four most pertinent categories of ransomware WannaCry/WannaCryptor, Locky, Cerber and CryptoWall with 50 samples of each category. It evaluated their triaging performance and run-time system performance. The evaluation for similarity detection performance of each method was based on the number of samples a method can match within the same ransomware category and across all the other three categories. The fuzzy hashing method SDHASH has performed relatively better than the other three methods based on the total number of samples matched in each category and across the three categories. However, the triaging performance of YARA rules is very close to the SDHASH method. This evaluation offers a direct and valuable comparison among the three completely different triaging methods; however, it requires further testing on large samples of ransomware and in-depth analysis of similarity scores of fuzzy hashing methods. In future, this evaluation can be extended to check the degree of similarity between the ransomware samples and the effects of using opcodes in YARA rules.

In the second part of the paper, in order to discover ransomware threat actors/groups or new ransomware families, an efficient fuzzy analysis approach is proposed to cluster ransomware samples based on the combination of two fuzzy techniques fuzzy hashing (e.g. SSDEEP/SDHASH) and Fuzzy C-Means (FCM) clustering [7]. In the future, this proposed fuzzy analysis approach could be automated by generating sparse fuzzy rules based on the pre-eminent results of FCM [31] and employing an adaptive fuzzy rule interpolation technique [32], [33], [34], [35]. Moreover, this sparse fuzzy rule base can be updated dynamically by employing dynamic fuzzy rule interpolation (D-FRI) method [36], [37], [38], [39], [40], [41], [42]. After further enhancing and automating this fuzzy analysis approach, it can be employed in different security frameworks [43].

ACKNOWLEDGEMENT

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

REFERENCES

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

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

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

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

[5] 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.

[6] 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 *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018.

[7] 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.

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

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

[10] 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

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

[12] 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

[13] N. Naik and P. Jenkins, "Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect," in *11th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2017, pp. 163–174.

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

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

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

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

[20] N. Naik and P. Jenkins, "Fuzzy reasoning based windows firewall for preventing denial of service attack," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 759–766.

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

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

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

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

[25] 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 *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018.

[26] 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*. IEEE, 2018, pp. 904–910.

[27] 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.

[28] N. Naik and P. Jenkins, "Discovering hackers by stealth: Predicting fingerprinting attacks on honeypot systems," in *IEEE International Symposium on Systems Engineering (ISSE)*, 2018.

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

[30] ——. (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

[31] Y. Tan, H. P. H. Shum, F. Chao, V. Vijayakumar, and L. Yang, "Curvature-based sparse rule base generation for fuzzy rule interpolation," *Journal of Intelligent & Fuzzy Systems*, Feb. 2019. [Online]. Available: https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs169978

[32] L. Yang, F. Chao, and Q. Shen, "Generalized adaptive fuzzy rule interpolation," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 839–853, Aug 2017.

[33] L. Yang and Q. Shen, "Adaptive fuzzy interpolation," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 6, pp. 1107–1126, Dec 2011.

[34] J. Li, L. Yang, Y. Qu, and G. Sexton, "An extended takagi–sugeno–kang inference system (tsk+) with fuzzy interpolation and its rule base generation," *Soft Computing*, vol. 22, no. 10, pp. 3155–3170, May 2018. [Online]. Available: https://doi.org/10.1007/s00500-017-2925-8

[35] L. Yang and Q. Shen, "Closed form fuzzy interpolation," *Fuzzy Sets and Systems*, vol. 225, pp. 1 – 22, 2013, theme: Fuzzy Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165011413001486

[36] N. Naik, C. Shang, Q. Shen, and P. Jenkins, "D-FRI-CiscoFirewall: Dynamic fuzzy rule interpolation for Cisco ASA Firewall," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019.

[37] 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.

[38] ——, "Genetic algorithm-aided dynamic fuzzy rule interpolation," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2014, pp. 2198–2205.

[39] N. Naik, R. Diao, C. Quek, and Q. Shen, "Towards dynamic fuzzy rule interpolation," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2013, pp. 1–7.

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

[41] 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. 1520–1527.

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

[43] N. Elisa, L. Yang, F. Chao, and Y. Cao, "A framework of blockchain-based secure and privacy-preserving e-government system," *Wireless Networks*, Dec 2018. [Online]. Available: https://doi.org/10.1007/s11276-018-1883-0