Accepted Manuscript

Similarity hash based scoring of portable executable files for efficient malware detection in IoT

Anitta Patience Namanya, Irfan U. Awan, Jules Pagna Disso, Muhammad Younas

PII: S0167-739X(18)32591-3
DOI: https://doi.org/10.1016/j.future.2019.04.044
Reference: FUTURE 4925

To appear in: *Future Generation Computer Systems*

Received date : 24 October 2018
Revised date : 1 April 2019
Accepted date : 19 April 2019

Please cite this article as:, Similarity hash based scoring of portable executable files for efficient malware detection in IoT, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.04.044

Similarity Hash based Scoring of Portable Executable Files for Efficient Malware Detection in IoT

Anitta Patience Namanya,
Irfan U. Awan
School of Electrical Engineering and Computer Science, University of Bradford, UK
apnamany, i.u.awan@bradford.ac.uk

Jules Pagna Disso,
Cyber Risk Intelligence,
BNP Paribas, London, UK
jules.pagnadisso@uk.bnpparibas.com

Muhammad Younas,
School of Engineering, Computing and Mathematics, Oxford Brookes University,
UK.
m.younas@brookes.ac.uk

***Abstract*—The current rise in malicious attacks shows that existing security systems are bypassed by malicious files. Similarity hashing has been adopted for sample triaging in malware analysis and detection. File similarity is used to cluster malware into families such that their common signature can be designed. This paper explores four hash types currently used in malware analysis for portable executable (PE) files. Although each hashing technique produces interesting results, when applied independently, they have high false detection rates. This paper investigates into a central issue of how different hashing techniques can be combined to provide a quantitative malware score and to achieve better detection rates. We design and develop a novel approach for malware scoring based on the hashes results. The proposed approach is evaluated through a number of experiments. Evaluation clearly demonstrates a significant improvement (> 90%) in true detection rates of malware.**

Keywords: Malware, Static Analysis, detection, hashes, Internet of Things,

I. Introduction

Internet of Things (IoT) offer new and exciting opportunities such as smart homes, smart devices, smart cities and smart transportation, to name but a few. IoT is growing at enormous scale and is expected to be used in connecting billion of devices in the near future. But as the market, scope and application areas of IoT increase, it becomes more vulnerable to various kinds of security breaches, such as malware, spoofing, jamming, etc. — these issues been surveyed in related work [33]. This paper focuses on the issue of malware in the IoT. With the growth of IoT, the types of malware are continuously evolving. Having various devices connected to the IoT changes not only the attack target landscape, but also supplies criminals with resources that were previously not available. IoT security challenges have made the IoT devices a vector for powerful DDoS attack in recent years [1]. Malware target IoT devices vulnerabilities so that exploited devices can become part of a botnet. The longer the malware is not detected, the more devices it can exploit. Thwarting analysis implies that the malware samples have become more complex over time, therefore, the evolution of malware is two sided: the growth in numbers collected daily and the complexity of the sample being discovered. For instance, according to AV-Test institute, over 856.62 million malware were collected in 2018. Only 13% (113.78 million) of these were new malware samples. The statistics from AV-test Institute show an exponential growth in the number of malware seen each year. The growth in complexity of malware is shown by the ever-evolving complex methods discovered in collected malware samples that are used to evade and/or disable malware prevention and detection systems.

It is therefore crucial to generate new methods that can isolate files that are variations of malware which have already been known. On one hand, this can shorten the time spent on analysing malware, and on other hand, it can also detect malware in different stages. Detection of malware in stages reduces the impact of sample analysis on system performance as less number of malware is needed to be analysed [2]. The need for secure, trustworthy and high-performance devices [3] in IoT devices and other fast systems automatically limits the use of dynamic analysis-based detection methods. Dynamic analysis requires more resources and more time to execute and observe the behaviour of the file. However, this is not feasible in the IoT environment given the scarcity of resources.

An efficient strategy is to utilise existing static feature-evaluation methods and to design new approaches for better detection rates. Evaluating static features of a sample can be constrained by the structure of the file. In this paper, we therefore focus on the Microsoft portable executable (PE) files. The rationale is that 90% of computer users in the world currently use Windows operating systems [4]. Moreover, with the multiplatform Windows 10, PE files are expected to continue being a possible threat vector as Windows systems are used in or interact with IoT devices.

The first crucial stage of triaging malware and clustering samples based on similarity matching normally uses hashing. Given that malware authors change internal structure/value to defeat basic hashing, a more complex hashing structure is needed. Therefore, we propose a combinational approach which is believed to lead to better results. If file similarities detected by the hashes as used as attribute similarity factors for a sample dataset, multiple attribute decision making, and evidence combination mathematical models are applicable to automate the decision-making process of malware detection. Various uncertainty-based reasoning models have been designed to assist expert systems in decision making based on unreliable data. This paper exploits this theory in order to propose and design a new approach that synthesises different hashing techniques to provide a quantitative malware score and to achieve better detection rates. The main contributions of the proposed method are:

- We combined tried and tested similarity matching hashes that are provided in almost all automated static analysis tools like Peframe and Virustotal. This implies that the deployment cost and manual effort required for dynamic analysis and advanced static analysis are avoided.
- The proposed method is scalable which can be customised to the needs of a malware analyst.
- Considering different hashes as file attributes reduces the storage capacity required by the system. This makes the proposed method light weight and more efficient.

The rest of this paper is structured as follows. Section 2 explores related work. Section 3 provides an overview of the background topics such as hashes, combination methods, and the evaluation approach used in the study. Section 4 describes design and modelling of the proposed method. Evaluation and results are presented in Section 5. Section 6 presents conclusion of the paper.

II. Related work

Although a lot of developments have been made in anti-malware research, most of the them have focused on behavioural analysis and dynamic heuristic analysis [6], [7]. Static analysis-based research has a limited scope. Existing research work around similarity matching hash functions has been limited to malware clustering as discussed herein. Since the proposed approach in this paper investigates into how multiple feature-based decision making has been utilised to improve malware detection rates in various scenarios, we discuss related work that has used multiple feature-based methods to improve malware detection. However, readers interested in general IoT security issues are referred to related work, such as [33] and [34], which provide surveys of challenges and open issues in IoT security.

DigitalNinjas [8] is a technical report that shows an initial work in the use of fuzzy hashing similarity to detect malware. Using only Ssdeep hash to detect different malware families, the work achieves a level of confidence of 67%. French and Casey [9] extended this work by conducting a study using different fuzzy hashing methods. A comparative study of popular similarity hashes used in malware clustering has been carried out in [10]. This study shows that fuzzy hashing outperforms cryptographic hashing. A methodology that clusters malware using Imphash based similarity checking was first introduced by Mandiant. This is now known as FireEye and is analysed in [11]. Although the results in [12] show higher sensitivity matching, the functionality of hashing in malware detection is still restricted to malware clustering. Similarly, in other related work [8], [9], [11]–[15], one hash is used in each study.

Multiple features-based decision making is applied in heuristic engines which use algorithms that do not necessarily provide an optimum solution. Unlike the old signature-based detection methods, heuristics utilise different features in malware and have proven to be better at unknown malware detection. Combination of file features and file relations improve malware detection results. This was introduced in [16] which developed a file verdict system called "Valkyrie". The authors build a semi-parametric classifier model to perform the combination and test the model against a dataset of 39,138 malware samples. This model is reported to have been applied in the Comodo Anti-Malware software.

Kolter & Maloof in [17] examine the results of various classifiers on malware detection through a simple heuristic based technique of text classification, which is known as n-grams. The proposed approach tests techniques which include, Naïve Bayes, decision trees, support vector machines and boosted variants. This approach not only uses multiple methodologies to train and test the algorithm, it also gives good detection rates of 95%-96%. However, this approach used a very limited dataset of 1971 malware which is a rather small dataset and thus it may not be applicable to the enormous malware samples being collected nowadays.

The MaTR [6] approach combines static heuristic file features and decision-tree machine learning algorithms to design a method for improving malware detection. This work initially recreate the experimental environment [17], highlights its weaknesses which are then used to build a a different detection algorithm. Experimentation using a dataset of 31193 malicious and 25195 clean files leads to 99.9 accuracy in the detection rates.

Xinjian et al, [18] propose to combine both static and dynamic features in order to improve malware classification. This method uses classifiers and adopts the prediction when the output is the same. This work tested the proposed method on 282 samples which is a very small sized test dataset and thus has very limits the scope.

The authors in [19] propose combining features using evidence combination methods in the detection of android malware. This work treats each feature statically which is extracted from android applications as information sources. It uses Dempster-Shafer theory of evidence combination to combine the information sources. Using a dataset of 1580 malware samples, the method achieves a detection accuracy of 97% and a false positive rate of 1.9%. The results show that combining different features improve malware detection rates. In our work, we apply this method to PE files and use static based hashes as representatives of heuristic features. These are believed to reduce resources, cost and efforts as compared to existing the method proposed in [19].

Studies towards attaching a malicious score to a file as a method of malware detection have been an evolving topic in security research. Taking the approach of the CVSS (Common Vulnerability Scoring System), MAEC project introduces the concept of a malware threat scoring system. It uses predefined categories to attach a threat score to a file [20]. RSA, the security division of EMC has introduced the RSA Security Analytics Malware Analysis scoring categories [21]. Both the MAEC and RSA categories look at static analysis as a required category. Kumar et al [22] propose to attach a heuristic score to a PE file which is based on the features extracted from PE file itself. Using 10 static features and a dataset of 1360 malware and 1230 clean files, the proposed model achieves an accuracy detection rate of 85%. Although the detection rates are not high, the scoring approach proposes a method of allowing a malware analyst in classifying malware based on urgency. In the quest to build a more resilient cyber space, this work further explores and expands the approach introduced in [22] and is an extension of our previous work presented in [23].

The work in this paper focuses on calculating a malicious file score from combining different hashing techniques (e.g., cryptographic hash, ImHash, SSDEP, PeHash) for malware detection purposes. Mathematical theories rooted in uncertainty reasoning are explored. It also explores the hashes as heuristic feature representatives and investigates into the effect of similarity hashes in relation to malware detection.

III. Overveiw of the Building Blocks of the Proposed Method

Existing malware detection methods rely on the expertise of malware researchers and analysts. However, it is difficult (if not impossible) to provide such expertise that can effectively and timely handle the massive numbers of newly discovered malware. This motivates the need for the design and development of new automated analysis methods that can use uncertain data to make decisions and fight malware. Many expert systems exhibit low errors in decisions making using uncertain data as they employ mathematical theories [24]. Thus as foundational information to our study, this section discusses the identified building blocks; the known and tested hash functions used in malware analysis, and uncertainty based cognitive approaches, and the methods used to evaluate the proposed approach.

A. Hashing Functions

Hashing functions are mathematical computations which take input (messages) and produce output (message digest) according to the contents of a file [12]. Some of the common hashes are illustrated as follows.

1) Cryptographic Hashes: These are the popular cryptographic hashes which include, MD5 sum, SHA1 and SHA256. These are mainly used for file integrity checks. With respect to similarity matching, these are limited in scope and efficiency due to the fact, that a minor change in the file can have a negative influence on the overall computed hash digest. However, these are useful in malware analysis at the initial identification and classification stage [13] as an immediate match means that the file is an exact copy of a known malicious file.

2) Ssdeep Hash: It is used to detect similarity in files and is usually known as Context Triggered Piecewise Hashing (CTPH) [25] or fuzzy hashing. It was initially used for anti-spam research (called Spamsum). It is a non-cryptographic hash based on a combination of the piecewise hashing (Fowler/Noll/Vo –FNV hash) and rolling hashing as shown in Fig. 1, which uses an example of a 5 byte hexadecimal block.

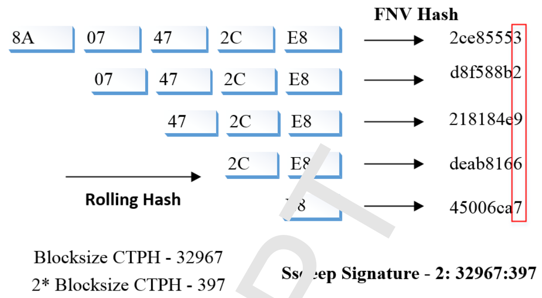

Fig. 1. Calculating the Ssdeep Signature

```
Blocksize: Block_Signature:Double-Block_Signature
6144:tkDtqNp95Ltuj[illegible]K2…a[illegible]q1Dw[illegible]EU/e:utUpDtqKmw/LqJWa
```

Fig. 2. Ssdeep Signature Form

Without considering the 64 signature length requirement of the algorithm, the FNV hashes are computed after setting a rolling window of a byte blocksize. The resulting CTPH signature is a concatenation of one string from the FNV hashes. A comparison algorithm then uses CTPH signature and Levenshtein Distance to calculate the sequence similarity between any 2 hashes. The score is normalised such that 50 score is considered as a reasonable threshold for a good detection. Kornblum [25] adopted Spamsum for forensic science resulting into a function called Ssdeep. It was applied to malware analysis by FireEye [26]. An Ssdeep signature of a file takes the form shown in Fig. 2 − which also includes an extract of an Ssdeep hash of a file. It has a very high confidence of 99% for the return similarity match score for any 2 files. and is therefore considered a critical step in static analysis of files.

3) Imphash: Designed by cybersecurity firm - FireEye [9], Imphash is used to compute the digest of the import section of portable executable files in three stages:

- Extract the structure of the PE file,
- Populate the imports in the order {API, Function (dll or sys or ocx)} for each API being found.
- Return the MD5 digest of the populated strings.

Similarity matching using Imphash allows for clustering of malware based on the contents and order of the executables' import tables. This hash is easily compromised by a change in the imports table order. Since malware can sometimes share some common system interaction behaviours, Imphash still plays a role in malware clustering.

4) PeHash: It represents a binary cryptographic hash value [14] which is related to the structure of a executable's file. In addition to the structure of the file, PeHash algorithm uses bzip2 compression ratio as an approximation for Kolmogorov complexity to get obfuscated data in file's sections. With the possibility that some malware repeat the use of specific encryption techniques, different instances of the malware sample can result in the same Kolmogorov complexity, thus creating a clustering mechanism. The algorithm first creates 2 classes of hash buffers: global properties and section hashes buffer. The PeHash is the SHA1 value of the overall hash buffer of the file and is noted to provide efficient clustering for polymorphic malware.

B. Evidence Combinational Methods

These are mathematical approaches that combine various belief factors which are determined based on different degrees of uncertainty in order to make the best effort decision [27]. Assuming two pieces of evidence defined by different degrees, e.g., A and B are respectively defined with degrees $\boldsymbol{a}$ and b. If these supports the hypothesis (**M**), then the resultant decision mainly relies on the degree of belief gathered from the evidences. We use the strict Archimedean t-conorms (as with logical connectives) to design combinational decision making methods [28]. The degree of belief in Maliciousness hypothesis (M) is defined by the function:

$$a * b \ \text{ in } \boldsymbol{M} \qquad (1)$$

1) Fuzzy logic: It is used in situations when deterministic data is not available. It states that the accurary or truth of end result depends on the accuracy of the support evidence [29]. According to [28], the algebraic sum is given by the following equation:

$$a * b = a + b - a.b \qquad (2)$$

2) The Certainty Factor model: This model is used in rule based systems such as MYCIN expert system that is used to diagnose bacterial infections. In this model, the overall belief in the hypothesis is calculated by taking into account the uncertainty in a rule and a single common factor. Using the T-conorms, given two pieces of supporting evidence, the overall degree of belief (O-DoB) ≥ (DoB-SE); which is the degree of belief in single evidence [30]. This is computed as:

$$a * b = \frac{a+b}{1+a.b} \qquad (3)$$

C. Method Evaluation Approach

Evaluating the malware detection performance of the proposed method requires the use of the binary classification of the confusion matrix, as shown in Table I.

TABLE I. CONFUSION MATRIX.

		Analysis Results	
		Malicious	Clean
Actual Sample State	Malicious	True Positive(TP)	False Negative (FN)
	Clean	False Positive (FP)	True Negative (TN)

The options in the confusion matrix result obtained from the similarity matches lead to being able to calculate various detection rates by using different metrics, shown in Table II.

IV. DESIGN OF THE PROPOSED METHOD

During the design phase we revisited and extended the PE format and our previous work [31].

A. Design Choice of Hashes

Table III shows the reasons as to why the various hashes were chosen. The resource section (rsrc) of a PE file is known to contain the information about any names and types of embedded resources. By combining the various aspects of the file sample using 4 various hashes, the overall achieved score is intended to represent the file's similarity with respect to already known malware samples.

TABLE II THE ALGORITHM NOTATIONS.

Notation	**Meaning**
H_{db}	Database of Hashes
Imp_H	Imphash
Pe_H	PeHash
Sd_H	Ssdeep Hash of the file
RSd_H	Ssdeep Hash of the file's Resource Section
Ni	Set of elements of attribute, i
MD5	MD5 sum
HFlag_set (H)	The flag setting function for hash type, H
$PopH_{db}$	Populating Malicious files Hash Database Function
HbDR	Hash Based Comparison Detection Rates Function
i	Hash of type, I, e.g., Imp_H, Pe_H, Sd_H, RSd_H
CFIi	Common Factor Index of an attribute, i
ESFi	Evidence Support Factor of Attribute, i
TDR	True Detection Rate calculated by: $\frac{TP+TN}{Total\_Sample}$
FPR	False Positive Rate: a measure of the negative samples flagged as positive. This is given by: $\frac{FP}{TN+FP}$
Recall	Based on thefollowing equation it calculates the number of actual positive files being detected: $\frac{TP}{TP+FN}$
PPV	Precision/ Positive Predictive Value (PPV) is measured: $\frac{TP}{TP+FP}$
ACC	This is a measure of Accuracy of true detections, which is calculated as $\frac{TP+TN}{TP+FP+TN+FN}$
F_1	The harmonic mean of precision and recall is calculated: $\frac{2.PPV.Recall}{PPV+Recall}$
CHA	Combined Hashing Approach
TLBSA	Traffic Light Based Scoring Assessor
FLM	Fuzzy Logic Method
CFM	Common Factor Model Method
GTP	Green Threshold Percentage
ATP	Amber Threshold Percentage

TABLE III. ARGUMENT FOR IN SCOPE HASHES.

Hash Type	**Reason**
PeHash	Overcoming Malware Obfuscation
Imphash	Classification by API
File Ssdeep Hash	Overall File similarity
Resource section Ssdeep Hash	PE Resource section file similarity.

ACCEPTED MANUSCRIPT

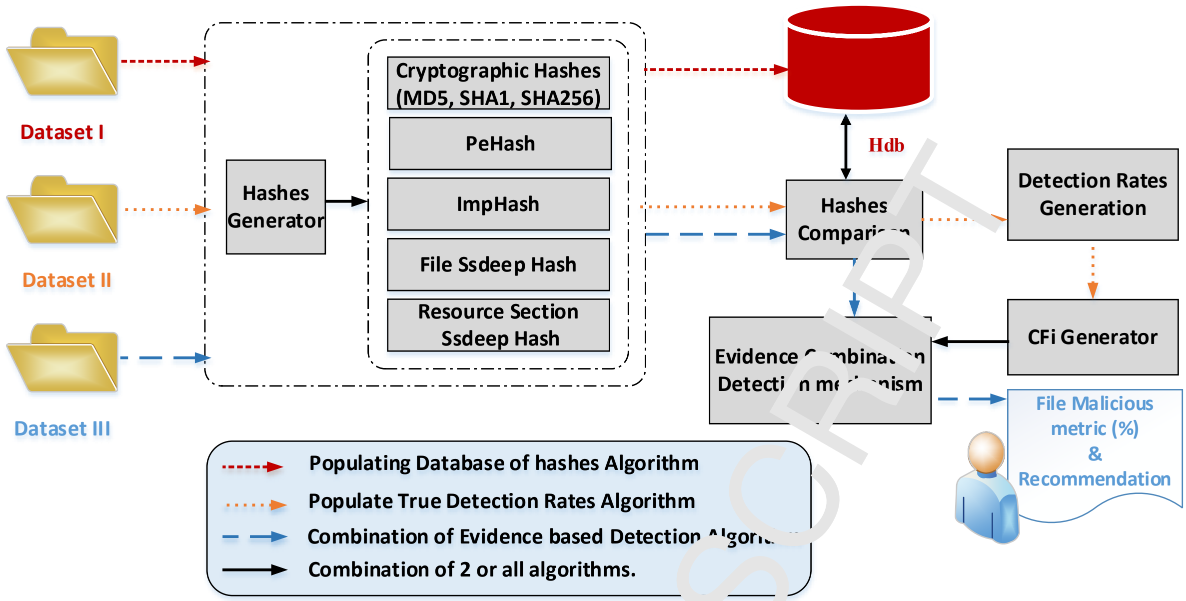

Fig. 3. The architectural representation of the proposed method

TABLE IV. DATASETS FORMATION AND WHERE THEY ARE USED IN THE METHOD.

Dataset	Use in the system
I	To populate the database of hashes
II⟵{IIm, IIc}	Used to calculate the True detection rated of the Hashes and the respective CFI.
III ⟵{IIIm, IIIc}	To validate the proposed approach

ALGORITHM I: ALGORITHM FOR GENERATING THE DATABASE OF HASHES

Input: Malware Dataset *I*
Output: Signature Hashes Database H_{db}

```
procedure: PopHdb
for file f in I do
    Extract the file hashes
        Hashes (f) ⟵ {MD5, Imp_H, Pe_H, Sd_H, RSd_H}
    If Hashes(f) ∉ Hdb then
        add Hashes(f) to Hdb
end for
end procedure
```

B. Architecture of the Proposed Method

Architecture of the proposed method is shown in Fig. 3. It considers the notations and metrics shown in Table II. The proposed method is divided into six different steps which are explained as follows.

Step 1: The Initial Single File Study

This initial study was performed on one randomly chosen clean file (arp.exe) from a Windows-based system. The original file was analysed and the different hashes of interest were computed. The file was then edited using Radare and the file hashes were recomputed. The hashes from the two files were compared.

Step2: Collecting the Datasets

Datasets in Table IV and Table VII are collected using the methods described below:

a) This study gathers dataset of malicious PE files from various sources such as malware from online malware repositories, our own honeypots and the malware repository of Netitude Ltd, UK.

b) Clean files from various types of Windows systems (e.g., Windows XP , Win 7, Win 8 and Win 10) were collected. Each file was saved as its MD5 sum to ensure that there was no file duplication in the dataset. As shown in Table IV, malicious files were split into 3 sub-datasets, I, IIm and IIIm, and Clean files were split into 2 sub-datasets, IIc and IIIc.

Step 3: Populating the Database of Hashes Signatures

The database of hashes (H_{db}) for the malicious files that are used as the initial signatures are calculated from random malware samples. These are collected in dataset I using the process of Algorithm I.

Step 4: Hashes Similarity Based Criteria Factor Index (CFI) Formulation.

Dataset II which has both malicious files and clean files is used at this stage. This step is broken down into 2 sub-steps;

a) Determine the individual performance of the hashes in relation to malware detection.

This involves comparing the hashes calculated for files in dataset II against the H_{db} by formulating the HFlag_set, where each of the 4 hashes has a specific position. For each file in Dataset II, five respective hashes are computed. Four different queries are run against the database. Each query returns a set of tuples;

$$X_{Hi} \leftarrow \{md5, \{Imp\_H, Pe\_H, Sd\_H, RSd\_H\}\} \qquad (4)$$

During the comparison of PeHash and Imphash, only the hashes, which are the same as the calculated hash, are pulled from the database. The HFlag_set position corresponding to the hash of type i is not a set if the set X_{hi} is Ø (null) and is a set otherwise. For resource Ssdeep hash and file Ssdeep hash, all the hashes are pulled from the database. A Ssdeep similarity match is done for the file hashes and the respective database populated hashes. If the maximum similarity percentage calculated is greater than zero, the HFlag_set position corresponding to the hash of type i is set. It is not set otherwise. Each file corresponds to one set of HFlag_set. The total count to achieve the confusion matrix parameters is populated for each hash as shown in Algorithm II.

ALGORITHM II: ALGORITHM FOR CALCULATING DETECTION RATES.

```
Input: Ds ⟵Dataset II , H_db
Output: Det_Rates
Overall Hash Based Detection Rate Phase
procedure: HbDR
for file (f) in Ds do
   HFlag_set f
   for i = 1 → 4                          ⊳Loop through all the hashes flags
        if f ∈ IIm then
           if HFlag_set fi then
                TP_i = +1
           else
                FN_i = +1
           end if
        end if
        if f ∈ IIc then
           if HFlag_set fi then
                FP_i = +1
           else
                TN_i = +1
           end if
        end if
        Update DetectionRates_i ⟵{TP_i, FN_i, FP_i, TN_i}
   end for
return Det_Rates
end procedure
```

b) Calculate the CFI of all the individual hashes

The detection rates obtained in sub-step (a) are used to calculate the CFI of each hash which is used as a belief factor for each hash. To minimise the error in the belief factors, true detection rates are used to calculate the factors. The true detection rates are normalised to the uniform range [0, 1]. Simple Additive weighting [32] is applied to the detection rate so that the degree of belief/ Criteria Factor Index (CFI) for each Hash method is defined as:

$$CFI_a = [\textstyle\sum_{n=1}^{4} TDR_n]^{-1}.TDR_i \qquad (5)$$

These CFI values are used as the belief factors for the respective hashing techniques. This supports the hypothesis that the file is indeed malicious. The values calculated are applied in the next step in order to obtain an overall malicious score for the file under test.

Step 5: Application of Evidence Combination Theory.

The values of Criteria Factor Index (CFI) are used as inputs to the combinational approach application. The MD5 comparison phase is a redundancy step, which is introduced to avoid replication of the malware samples in the experiment. The Hashes comparison phase uses the file calculated hashes and compares them against H_{db}. The query in equation (4) is used in this phase too. The belief factors for the hashes are computed from the results obtained from the respective queries. For PeHash and Imphash, if the resulted set is not null, then the corresponding ESF is equivalent to the CFI of the respective hash. Otherwise the hash's ESF is set to zero. For Resource Section Ssdeep hash and file Ssdeep hash, the corresponding ESF is equivalent to the CFI multiplied with the maximum similarity percentage, which is achieved by comparing the file and the hashes in the database. The Calculated ESF values of the various hashes are combined using the evidence combinational models detailed in Section III. This is to get the algebraic sum for the overall hypothesis which is fed into the TLBSA (Traffic Light Based Scoring Assessor).

Step 6: TLBSA Threshold.

The resultant percentages from the combined hashing technique are compared to add an overall TLBSA that evaluates the score attached to the file. It gives the user a recommendation based on Table V. Since the system does not completely guarantee that the file is safe, the final decision on how the file analysis is handled is left to the system user or analyst.

TABLE V. THE TLBSA COLOUR DEFINITIONS

Colour	Deduced file intent	System Recommendation
Red	Definitely malicious	Do not Install
Amber	Medium Suspicion	Highly encouraged to submit it for further analysis
Green	Low Suspicion	Submit it for further analysis

TABLE VI. TEST BENCH SPECIFICATIONS

Tool	Specifications/ Details
Computer system	Dell T1700, CPU – Intel Xeon@ 3.1GHz, RAM 32GB. Hard Disk – 500GB
Machine OS	Linux Mint 17.1 (#64 – Ubuntu SMP)
Static Analysis tool	Study specific Static Analysis Tool – calculates the Ssdeep, Resource Section Ssdeep hash, PeHash, and Imphash
Data management tools	SQLite Studio version 3.0.6. Python IDLE version 2.7.9

TABLE VII. THE EXPERIMENTAL DATASET

Dataset	I	II	III	Total Files
Malicious files	34224	32844	37460	104528
Clean files		698	940	1638

TABLE VIII. MALWARE TYPE DISTRIBUTION IN THE MALWARE DATASET

Malware Type	Percentage	Malware Type	Percentage
Trojan	66.84%	Dropper	0.65%
Adware	22.30%	Virus	0.29%
Worm	9.03%	Spyware	0.11%
Downloader	0.71%	Exploit	0.08%

V. EVALUATION OF THE PROPOSED METHOD

This section presents the evaluation of the proposed method. It first describes the dataset preparation process and the test environment. It then provides an analysis and discussion of the results.

A. Dataset Preparation and Test Environment

For the experiment, we collected 104528 malicious files. All these were investigated using ClamAv engine version 0.99.2 in order to ensure that they were indeed known malicious files. As shown in Table VII, the total dataset was prepared so as to have different sets for the different steps in the experiment. The malware family distribution of the used dataset is shown in Table VIII. The algorithms were implemented in Python and the database of Hashes was managed using SQLite in a Linux box. The specifications are shown in Table VI. We use the Linux as a safe environment since the malware are all PE files and therefore ensuring that the results are not corrupted by unknown self- infection.

B. Results and Analysis.

Table IX shows the similarity matching based results achieved in the first phase of the study with the single file analysis. Some hashes are heavily affected by a small change in a file while there is possibility of a small or no effect in other hashing functions. This justifies the reason of further exploring hash-based similarity matching for a possibility of efficient malware detection.

In the second phase, Dataset II is used to compute the CFI metric values of the four hashing techniques which are shown Table X. The results obtained are also used to evaluate detection rates of the different hashing techniques, as shown in Fig. 4. Dataset III is used to calculate the overall percentage of file maliciousness in order to validate the proposed framework. The results achieved for the proposed approach are compared against the results achieved for each individual hash in Fig. 6. Fig. 5 represents the file scoring area curves of each adopted method which shows that most of the malicious files' score is higher than the clean files. We compare the two proposed methods and the individual hashes in Fig. 6. Since the aim of this study is to devise an optimum malware detection methodology, we further investigate the true positive and false negative trade-off of the two methods in Fig. 7. Fig. 8 is used to determine TLBSA threshold percentages. We then present the detection rates of each family of malware achieved in Fig. 9.

TABLE IX. COMPARISON OF HASHES FROM THE SINGLE FILE STUDY

Hash Type	Original File Value	Edited File Value	Match (%)
MD5	33f9b0e02d9d93f920605d02fb53f3fd	a[illegible]d6591b8b8dad5f7f1470c90971e75	0
SHA1	4a22e401ad5adb7b3de8f819e86d8461d764d195	[illegible]b98e35c1f92f844b57376ee467ee977cc074bd	0
SHA256	1f4c090dfa389b3c6b16eb42299fb815f24efac7ca541bb60821e3da01 31b8f6	[illegible]d4f056223439e83f2fffbe3c463e178da8465fabeb51243c04 a3d2922de8fa2	0
Ssdeep-File	384:5u3Smmq6aYaBpYFAfjhXrToHWS4mW4sme9V:Avmq6affY FAfjhr8sgE	384:5u3Smmq6aYaBpYFmfjhXrToHWS4mW4sme9V:Av mq6affYFmfjhr8sgE	99
PeHash	5515f8e47661c7e170aee948cca7c8dc6198c08f	5515f8e47661c7e170aee948cca7c8dc6198c08f	100
Imph	880bb6799a6e1a5ff7b4f022ff4003a9	880bb6799a6e1a5ff7b4f022ff4003a9	100
Ssdeep - Resources	96:8EWS1pEmWwOh/VsBgtAb88caS5Ur9I5fa9VWPBMXsmrC9V :NWS4mWNJXCu6Xsme9V	96:8EWS1pEmWwOh/VsBgtAb88caS5Ur9I5fa9VWPBM XsmrC9V:NWS4mWNJXCu6Xsme9V	100

TABLE X. COMPUTED METRICS

Malware detection performance of the individual in-scope Hashes and calculation of the CFI							
	Recall (%)	PPV (%)	ACC (%)	F-score (%)	Detection Rates		CFI (%)
					TRUE (%)	FALSE (%)	
ImpH	85.6	93.3	89.7	89.3	85.7	14.3	27
PeH	82.8	100	91.4	90.6	83.1	16.9	26.2
FuzH	76.2	100	88.1	86.5	76.7	23.3	24.1
ResFH	71.7	99	85.5	83.2	72.3	27.7	22.7

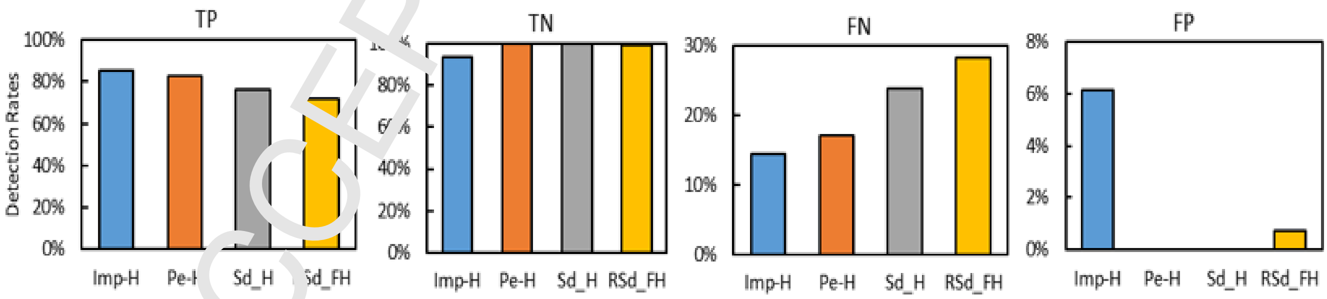

Fig. 4. The Hashes Detection Rates using Dataset II.

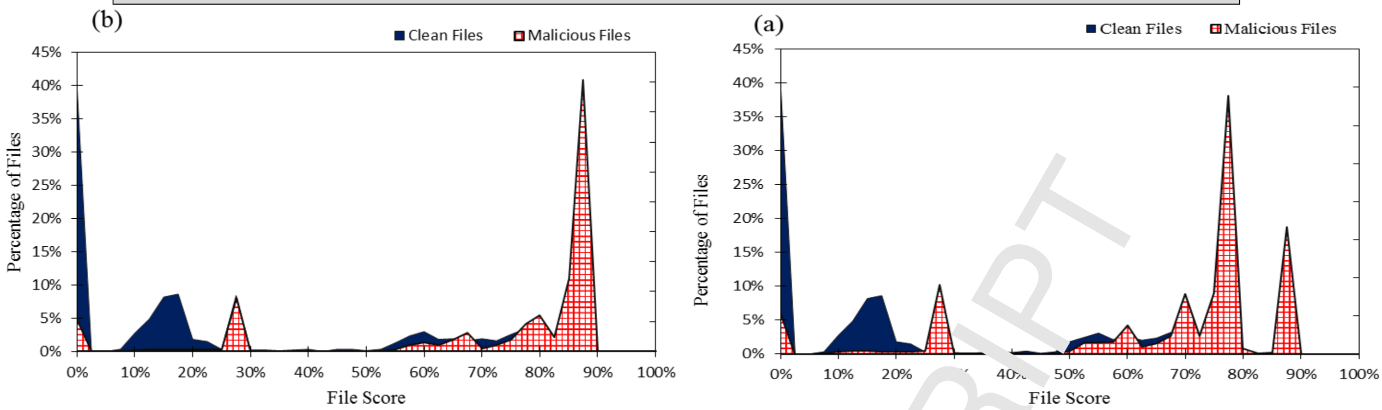

Fig. 5. The Clean and Malware file Score Area curves (a) Common Factor method and (b) Fuzzy Logic Method.

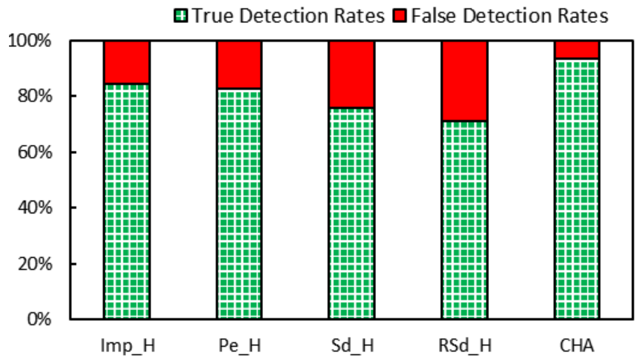

Fig. 6. Comparison of the Hashes and the Evidence Combination methods

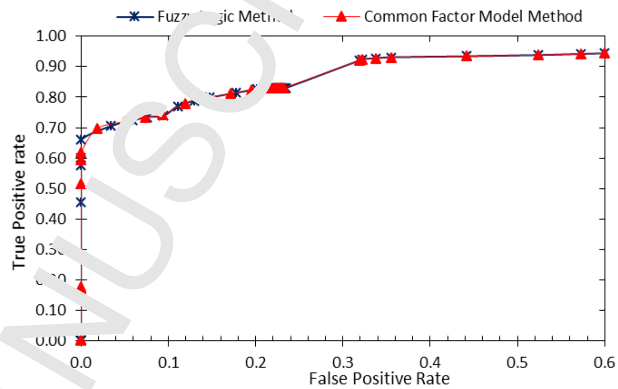

Fig. 7. TP rate vs FP rate curves for the Combination methods.

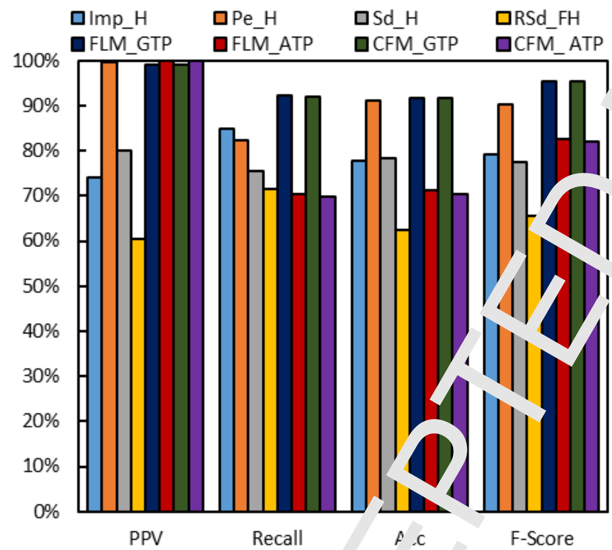

Fig. 8. Recall, Precision, Accuracy and F-Score Comparison for the proposed methodology percentage thresholds.

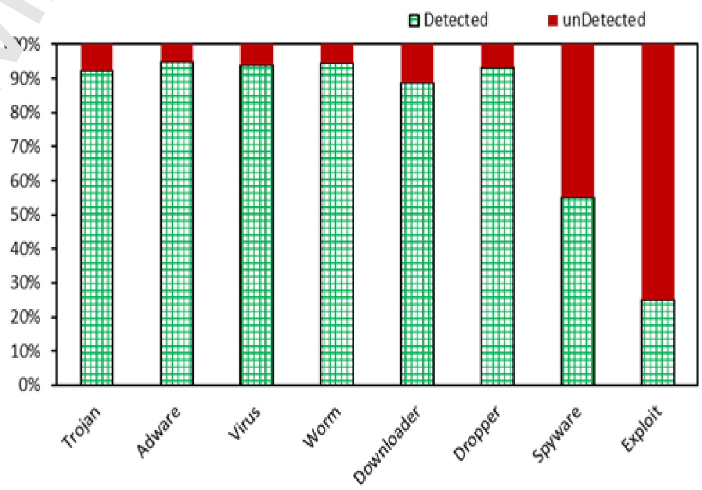

Fig. 9. Malware type detection ratios for the dataset used.

C. *Analysis and Observations*

This study designed and evaluated two methods for combining the individual hashes results for malware detection. Table IX results, achieved at the first stage of the study, show that similarity hashes are effective in matching similar files, which have slight differences in their content. Using dataset B, the introduced resource section hash matching gives the second-best precision value in the 4 hashes in which PeHash is the best performing of the 4 hashes as shown in Fig. 4. Imphash gives the highest false positive detection but also provides the lowest false negative detection. The different levels in the detection rates provide an argument for combining them to achieve a more efficient detection approach. Analysis of the logs to validate the Combined hashing methodology results into achieving an overall false detection rate of 6.8% and a true detection rate of 93.2%. These are the best performance values in comparison to the results achieved by the individual hashing algorithms as shown in Fig. 6. We analysed the dataset clean file scores vs malicious file scores for the two evidence combination methods. Both curves in Fig. 5 show that 83% of the malicious files obtain a malicious score above 50% while 78% of the clean files have a malicious score less than 50%. However, reviewing the true positive to false positive detection trade-off in Fig. 7, the proposed methods shows that this technique is susceptible to very high false positive of 60%, thus requiring an evaluation of the model to achieve a better trade off.

TABLE XI. COMPARING DETECTION RATED FOR THE TLBSA THRESHOLDS

Comparative analysis of the performance of the proposed method after application of the TLBSA.					
		Prec (%)	**Recall (%)**	**Acc (%)**	**F-Score (%)**
Fuzzy Logic Method	**(FLM_GTP (≥25%)**	99.2	92.2	91.6	95.5
	(FLM_ATP (≥75%)	99.9	70.5	71.2	82.7
Common Factor Method	**(CFM_ GTP (≥25%)**	99.2	92.1	91.6	95.5
	(CFM_ATP (≥70%)	100	69.8	70.4	82.1

We therefore introduced the TLBSA assessor at this stage, as described earlier by creating the percentage thresholds for the 3 zones. With the thresholds obtained, we evaluated how well our methods work against the individual hashing algorithms in Fig. 8. ATP outperforms all the individual hash techniques. However, since this percentage creates a very low True Positive rate of 70% for the Fuzzy logic method and 62% for the Common Factor Model method, there is a need to analyse the needed GTP. It creates a much-needed rise in the True Positive rate of 92% for both the proposed techniques. The use of TLBSA increases the detection efficiency of the system as shown in Table XI. The threshold percentages allow optimum trade-offs and enable the system to provide a user with information that helps protect their system with an accuracy of at least 92% that has been achieved in this study. Fig. 9 shows the overall detection ratios for the malware types in the used dataset. Of the 8 types collected, the designed method provides efficient malware detection for 6 types.

VI. CONCLUSION

This study developed a new approach to combine the results from individual similarity hashes to demonstrate an overall best performing recall of 92%, a system accuracy of 91%, a precision of 99%, and an F-score of 95%. These results significantly outweigh the results when one considers the detection rates of the existing individual hashes. Our approach is flexible and it can be customised and extended by malware analysts for the analysis of other file types. Our approach is safe against sandbox and dynamic analysis environment evading malware since it uses static analysis. It simplifies the identification of malicious files by providing a quantitative value that indicates how malicious a file is. It also optimises the storage required for database of hashes. Furthermore, it allows for an easy update of signatures so that performance can be increased with the increase in number of hash signatures. Our system design used light weight tools that makes it significantly efficient. The results achieved in this study show that the proposed method provides a way of building an efficient, integrated malware detection system for IoT devices. .

REFERENCES

[1] "IoT botnets responsible for more powerful DDoS attacks - Bitdefender BOX Blog," *Bitdefender*. [Online]. Available: https://www.bitdefender.com/box/blog/iot-news/iot-botnets-responsible-powerful-ddos-attacks/. [Accessed: 26-Mar-2019].

[2] Y.-D. Lin, Y.-C. Lai, C.-N. Lu, J.-K. Hsu, and C.-Y. Lee, "Three-phase behavior-based detection and classification of known and unknown malware," *Secur. Commun. Netw.*, p. n/a-n/a, Jan. 2015.

[3] A. B. Waluyo, D. Taniar, W. Rahayu, and B. Srinivasan, "Trustworthy data delivery in mobile P2P network," *J. Comput. Syst. Sci.*, vol. 86, no. Supplement C, pp. 33–[illegible], Jun. 2017.

[4] "Operating system market share." [Online]. Available: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0. [Accessed: 27-Dec-2017].

[5] "Triage Analysis," *Malware Unicorn*. [Online]. Available: /RE101/section4/. [Accessed: 08-Jan-2018].

[6] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimaila, and S. Rogers, "Malware target recognition via static heuristics," *Comput. Secur.*, vol. 31, no. 1, pp. [illegible]–147, Feb. 2012.

[7] Z. Cui, F. Xue, X. Cai, Y. Cao, G. g Wang, and J. Chen, "Detection of Malicious Code Variants Based on Deep Learning," *IEEE Trans. Ind. Inform.*, pp. 1–1, 2018.

[8] DigitalNinja., "Using Fuzzy Hashing Techniques to Identify Malicious Code." Apr. 2007.

[9] David French, "Beyond Section Hashing," 2010 CERT Research Report CMU/SEI-2012-TR-004, 2011.

[10] N. Sarantinos, C. Benzaïd, O. Arabiat, and A. Al-Nemrat, "Forensic Malware Analysis: The Value of Fuzzy Hashing Algorithms in Identifying Similarities," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 1782–1787.

[11] S. Arik, T. Huang, W. K. Lai, and Q. Liu, *Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings*. Springer, 2015.

[12] Y. Li *et al.*, "Experimental Study of Fuzzy Hashing in Malware Clustering Analysis," presented at the 8th Workshop on Cyber Security Experimentation and Test (CSET 15), 2015.

[13] C. Oprisa, M. Checiches, and A. Nandrean, "Locality-sensitive hashing optimizations for fast malware clustering," in *2014 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2014, pp. 97–104.

[14] Georg Wicherski, "peHash: a novel approach to fast malware clustering," in *LEET'09 Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, 2009, vol. 1–1.

[15] "Tracking Malware with Import Hashing," *M-unition*. [Online]. Available: https://www.mandiant.com/blog/tracking-malware-import-hashing/. [Accessed: 14-Jul-2015].

[16] Y. Ye *et al.*, "Combining File Content and File Relations for Cloud Based Malware Detection," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2011, pp. 222–230.

[17] J. Z. Kolter and M. A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild," *J Mach Learn Res*, vol. 7, pp. 2721–2744, Dec. 2006.

[18] X. Ma, Q. Biao, W. Yang, and J. Jiang, "Using multi-features to reduce false positive in malware classification," in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, 2016, pp. 361–365.

[19] Y. Du, X. Wang, and J. Wang, "A static Android malicious code detection method based on multi-source fusion," *Secur. Commun. Netw.*, vol. 8, no. 17, pp. 3238–3246, Nov. 2015.

[20] "Malware Threat Scoring System | MAEC Project Documentation." [Online]. Available:

http://maecproject.github.io/documentation/use_cases/cyber_threat_analysis/malware_threat_scoring_system/. [Accessed: 04-Nov-2016].

[21] "Malware Scoring Modules," *RSA Security Analytics Documentation*, 05-Mar-2014. [Online]. Available: https://sadocs.emc.com/0_en-us/090_10.4_User_Guide/40_InvestigAnalysis/00_Investig_Flo/MaScorMod. [Accessed: 04-Nov-2016].

[22] A. Kumar and G. Aghila, "Portable executable scoring: What is your malicious score?," in *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, 2014, pp. 1–5.

[23] A. P. Namanya, Q. K. A. Mirza, H. Al-Mohannadi, I. U. Awan, and J. F. P. Disso, "Detection of Malicious Portable Executables Using Evidence Combinational Theory with Fuzzy Hashing," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016, pp. 91–98.

[24] S. Salicone and M. Prioli, "Mathematical Methods to Handle Measurement Uncertainty," in *Measuring Uncertainty within the Theory of Evidence*, Springer, Cham, 2018, pp. 17–36.

[25] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digit. Investig.*, vol. 3, Supplement, pp. 91–97, Sep. 2006.

[26] Dunham Ken, "A fuzzy future in malware research," *The ISSA J.*, vol. 11, no. 8, pp. 17–18, 2003.

[27] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[28] B. Więckowski, "Review of Proof theory for fuzzy logics. Applied Logic Series, vol. 36," *Bull. Symb. Log.*, vol. 16, no. 3, pp. 415–419, 2010.

[29] S. Salicone and M. Prioli, "Basic Definitions of the Theory of Evidence," in *Measuring Uncertainty within the Theory of Evidence*, Springer, Cham, 2018, pp. 93–105.

[30] R. R. Yager and L. Liu, *Classic Works of the Dempster-Shafer Theory of Belief Functions*. Springer Science & Business Media, 2008.

[31] A.P. Namanya, J. P,Diss and I.Awan "Evaluation of automated static analysis tools for malware detection in Portable Executable files," in *2015 31st UKPEW*, University of Leeds 2015, pp. 81–95.

[32] "Simple Additive Weighting Method," in *Multiple Attribute Decision Making*, 0 vols., Chapman and Hall/CRC, 2011, pp. 55–67.

[33] M. A. Khan and K. Salah "IoT security: Review, blockchain solutions, and open challenges", Future Generation Computer Systems, Vol 82, May 2018, pp. 395-411

[34] K. Sha, W. Wei, T.A. Yang, Z. Wang and W. Shi "On security challenges and open issues in Internet of Things" Future Generation Computer Systems, Vol. 83, June 2018, pp. 326-33

Dr. Anitta Patience Namanya completed her PhD with the Cyber Security Research Group at the University of Bradford. She obtained her Master's degree in Network and Performance Engineering from the University of Bradford and holds a bachelor's in Telecommunication Engineering. Her research interests include malware analysis, intrusion detection, honeypots, cyber security policy, mobile security and network performance. She is a member of IEEE and ACM.

Prof. Irfan Awan is a Professor of Computer Science and the Head of the School of Electical Engineering and Computer Science at the University of Bradford, UK. He holds a PhD in Computer Science from the University of Bradford. His research interests include network security, communication systems and performance modelling. He has chaired and organised various international conferences and workshops and served as a technical programme committee member for several international conferences. He is a steering committee member of UKPEW workshop series, IEEE PMECT workshop series and an associate editor of several internal journals and has edited several special issues of international journals. He is a member of IEEE, BCS and fellow of HEA.

Dr. Jules Pagna Disso is the Group Head of Cyber Risk Intelligence at BNP Paribas. He has extensive knowledge in malware analysis, protocol analysis, threat intelligence gathering and vulnerability discovery. Jules holds a PhD in Intrusion Detection System as well as many cyber security related certifications. His research interests include but not limited to cyber related security for Industrial Control Systems, honeypots, botnet, cloud security, forensics, threat analysis, and vulnerability identification. He is a member of IEEE and ACM.

Muhammad Younas is a Reader/ (Associate Professor) in Computer Science at the School Engineering, Computing and Mathematics, Oxford Brookes University, UK. His research interests include Web technologies, Big Data, IoT and Cloud computing. He received a Ph.D. in Computer Science from the University of Sheffield, UK. He has published more than hundred papers in international journals and conferences. He is on the editorial and advisory boards of international journals and is also involved in the steering, organizing and program committees of refereed international conferences and workshops

Similarity Hash based Scoring of PE Files for Efficient Malware Detection in IoT

Paper – Highlights

a) This paper explores four hash types (PeHash, Ssdeep, ImpHash and Resource Section Ssdeep Hash) currently used in malware analysis for portable executable (PE) files. We use evidence combinational mathematical methods to combine the results from the four hashes; Fuzzy logic and the Certainty Factor Model.

b) Similarity hashing has been adopted for sample triaging in malware analysis and detection. File similarity is used to cluster malware into families such that their common signature can be designed.

c) We design and develop a novel approach for malware scoring based on the hashes results. The proposed approach is evaluated through a number of experiments. Evaluation clearly demonstrates a significant improvement (> 90%) the in true detection rates of malware.

d) The main contributions of this work are to improve detection rate of malware and to provide a quantitative malware scoring mechanism for achieving improved confidence level in decision making.

e) Hash functions are easily calculated during the basic static analysis of a malware sample. This implies that the deployment cost and manual effort required for dynamic analysis and advanced static analysis are avoided.

f) It is scalable and can be customised to the needs of a malware analyst. The algorithms can also be adapted to other file types using file similarity matching hashes.

g) Considering the different hashes as file attributes reduces the storage capacity required by the system. This makes it lightweight and therefore the method does not impact system resources heavily.

h) We combined most popular similarity matching hashes that are provided in almost all automated static analysis tools like Peframe and Virustotal.

i) A dataset of 104528 malicious files which were used against 1638 clean files collected from fresh windows installs. The experiment was run in a Linux box to avoid self-infection. All scripts were written in python and the database created was managed using SQLite.

j) This initial study was performed on one randomly chosen clean file (arp.exe) from a Windows-based system. The original file was analysed and the different hashes of interest computed. The file was then edited using Radare and the file hashes were recomputed. Similarity matching based results showed that some hashes are heavily affected by a small change in a file while there is possibility of a small or no effect in other hashing functions.

k) In the second phase of the study, the introduced resource section hash matching gives the second-best precision value in the 4 algorithms and PeHash is the best performing of the 4 hashes. ImpHash gives the highest false positive detection but also provides the lowest false negative detection.

l) In the third phase, the Combined hashing methodology results into achieving an overall false detection rate of 6.8% and a true detection rate of 93.2 %, which are the best performance values in comparison to the results achieved by the individual hashing algorithms.

m) However, reviewing the true positive to false positive detection trade-off for the proposed method shows that this technique is susceptible to very high false positive of 60%, thus requiring an evaluation of the model to achieve a better trade-off. We therefore introduced the TLBSA (Traffic Light Based Traffic Light Based Scoring Assessor) at this stage. It creates a much-needed rise in the True Positive rate of 92% for both the proposed techniques.

n) This study developed a new approach to combine the results from individual similarity hashes to demonstrate an overall best performing recall of 92%, a system accuracy of 91%, a precision of 99%, and an F-score of 96%. These are better figures than when one considers the detection rates of the existing individual hashes.

o) Our approach can be customised and extended by malware analysts for the analysis of other file types.