Malicious Portable Executable Static Scoring method using Evidence Combinational Theory with Fuzzy Hashing

Anitta Patience Namanya[1]
School of Electrical Engineering and Computer Science
University of Bradford
Bradford, UK
A.Namanya @student.bradford.ac.uk

ABSTRACT
Malware detection and prevention systems are bypassed by malicious files in computer networks as malware become more complex and vast in number. This work introduces and investigates how different hashing results can be combined to achieve better detection rates. Two evidence combination theory based methods are applied in this work in order propose a novel way of combining the results achieved from different hashing algorithms. Our results show that the detection rates are improved when evidence combination techniques are applied.

CCS CONCEPTS
• Security and privacy→Intrusion/anomaly detection and malware mitigation

Keywords
Malware Static Analysis, Malware detection, Evidence combinational theory, Fuzzy hashing, PE files.

1. INTRODUCTION
The evolution of malware takes a two sided dynamic; the growth in number and the complexity of the samples discovered daily. AV- Test Institute reports to be registering over 390,000 new malware samples daily while McAfee reported to have collected nearly 500 million malware samples by the end of 2015 [1, 2]. Of the nearly 550 million malware samples collected this year by AV-Test Institute, just about 12 % are new samples. Malware samples have also become more complex and therefore, tedious to detect and analyse because of new detection evasion methods also called obfuscation techniques that are discovered daily. Some old malware are repackaged as new malware using sophisticated packers which means that old signatures cannot be used to detect them. Some malware employ anti- debug and anti-Virtual machine techniques which means that they might give false results during analysis if they detect that they are being run in a debugger or a virtual environment. The unknown 300 test [3] is evidence that simple evasion techniques are effective on most anti-virus systems. The scarcity of the malware analysis expertise means that methods that have the ability to optimise the detection of variants of already known malware samples are needed. Of the known malware analysis processes; static and dynamic, static analysis is less time consuming and has less chances of wrong analysis results as there is no execution of the malware samples. Hashing is a static based method that is used to detect similarity in malware samples and therefore has been widely adopted by the malware analysis community for clustering and classification of malware families. This speeds up malware analysis processes.

Evaluating file static features can be constrained by the structure of the file so this work is limited to Microsoft binary portable executable (PE) files. Moreover, over 90% of all computer users still use windows operating systems [4]. Evidence combinational theory is rooted in reasoning models which use unreliable data from different sources to make decisions based on uncertainty [5]. The novel ideas to be introduced by this work are:

- Using evidence combinational theory to combine hashing detection results.
- Investigation into the effect of the application of evidence combinational theory to malware detection.
- A quantifiable metric to represent the malicious status of a file and a recommendation to guide the end user.

2. RESEARCH PROBLEM
Malware analysts use information from different sources to make more conclusive decisions about malicious samples, build detection signatures for anti-virus systems. Different Hashes are computed from the digests of a malware sample and included in almost all analysis reports. While hashing methods have been adopted to classify and cluster malware families during sample triaging in malware analysis process, it is still heavily affected by high false negatives. There is a possibility that a combination of different results from the hashing methods could lead to be better detection rates. With no defined meaning to the sample similarity percentages for the hashing methodologies, there is need to provide a more quantitative measure of how malicious a file is for an everyday user based on the analysis results.

3. BACKGROUND
In order to handle the huge numbers of malware samples daily, automation of many of the malware analysis techniques is necessary. The better strategy is to optimise existing static analysis based methods to make the processes faster and more efficient therefore increasing their malware detection rates. File hashes are some of the information retrieved from the file during static analysis. Designing tools that replicate the decision making process of a malware analyst using the results from the hashes is one way of achieving this. Most expert systems show low errors in decisions that are based on uncertainty because of the different mathematical theories developed and implemented [6].

3.1 Hashing Algorithms
Hashing algorithms compute digests of a file that are unique to the sequence of the file binary contents and structure. The hashes identified for this study are:

[1] Supervised by Prof Irfan Awan and Dr Jules Pagna Disso

3.1.1 *Ssdeep Hash*

Ssdeep was first introduced in anti-spam research to detect similarity in files and later adopted by Kornblum [7] into Ssdeep for the purposes of forensic science. The percentage of similarity attached to any two files by this hash can sometimes justify why the files are the same or in the same family of malware. In this work, the file Ssdeep hash and the resource section Ssdeep hash are considered.

3.1.2 *Imphash*

First proposed by Mandiant cybersecurity firm [8], Imphash is a hashing method that is calculated from the digest of the import section of the executable file. It has been incorporated in many static analysis tools like VirusTotal.com, Peframe and Pefile among others.

3.1.3 *PeHash*

It is a function that generates the binary cryptographic hash value of the structural data found in the file header and executable's section data[9]. It provides interesting clustering matches for instances of similar polymorphic malware samples.

3.2 Evidence Combinational Methods

If we have two different pieces of evidence with two different degree of belief (**X** with Degree *x* and **Y** with degree *y*) that support the hypothesis (***M***) that the file is either malicious or not, the result heavily depends on the degree of belief placed on the different pieces of evidence. Using the fuzzy set union operators T-conorms introduces logical connectives to design the reasoning system [6] based on the degrees of belief. We use strict Archimedean t-conorm because they can approximate every continuous t-conorm that take the value in the range (0-1). The two identified methods are:

3.2.1 *Fuzzy logic*

The fuzzy logic approach follows that the end result is only true if and only if either of the support evidence is true. Considering the initial hypothesis of Maliciousness (M), our degree in belief of this hypothesis using fuzzy logic approach defines the function: x * y in **M**. Using the important class of "strictly Archimedean" t-conorms of fuzzy logic [6] the algebraic sum is given by $x * y = x + y - x \bullet y$

3.2.2 *The certainty Factor model (CF model)*

A reasoning method that manages uncertainty in rule based systems which was developed in 1975 for MYCIN expert system [10] that was designed to diagnose bacterial infections. To compute the overall belief in the hypothesis, an expert represents the uncertainty in the rule by using a single Common Factor (CF) for every rule. The CFs work as the degree of belief attached to each rule. Using the notation defined previously, the overall belief is given by $x * y = \frac{x + y}{1 + x \cdot y}$

4. METHOD

The designing of this methodology followed 5 steps. The last 3 steps form the core of the proposed method design in Figure 1. The steps are:

1. Single File Analysis: One clean file was edited using a binary editor Radare and 2 characters were added thus changing the contents of the file. The 7 different file hashes; MD5, SHA1, SHA256, file Ssdeep hash (FuzH), PeHash (PeH), Imphash (ImpH) and resource section Ssdeep hash (ResFH) of the two files were tested for similarity. While PeH, ImpH and ResFH return 100% match, FuzH return a 99% match and the rest returned a 0% match. Therefore, the study focuses on the 4 hashes that return a similarity match metric.
2. Collecting and formulation of the datasets

Table 1. The Experiment Dataset

File Status	Total Files	Split into groups
Malicious files	104528	A, Bm and Cm
Clean files	1638	Bc and Cc

Table 2. Datasets Formulation and their uses

Dataset	Number of files	Use in the system
A	34224	Training phase.
B ⟵{Bm, Bc}	33542	Baseline-Creation phase and CF generation.
C ⟵{Cm, Cc}	38261	Detection-method evaluation phase.

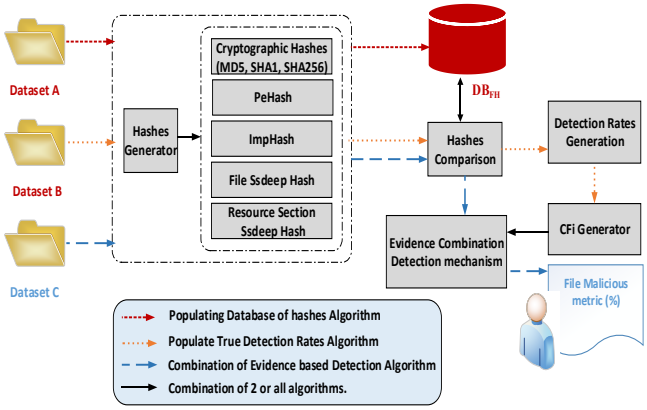

Figure 1. The pictorial representation of how steps 3, 4 and 5 interact with the elements of the system design

3. Training the Database of Hashes (Signatures): In building the malicious hashes that are used as the initial signatures, the randomly selected malware samples in dataset A are used to populate the database of hashes (DBFH)
4. Individual Hashes Malware Detection Rates study: The 4 respective hashes of each executable file in Dataset B are computed and then the hashes are used as query variables to query DBFH. The Hash's corresponding flag is set if and only if it returns a positive response. Since the files were tagged clean or malicious depending upon the status, false and true detection could easily be identified based on the confusion matrix approach.
5. Evidence Combinational Theory Approach: In order to make the detection rates compatible with the combinational theories, the positive detection rates are normalised to probabilities that add up to 1. The normalised detection rates take the form of the degree of belief in the uniform range [0, 1]. degree of belief/ Common factor for each Hash method (CFi) is defined as: $CF_i = \left[\sum_{n=1}^{4} TDR_n\right]^{-1} * TDR_i$

Where $TDR_a = \frac{True\_Detections_a}{Number\_in\_DatasetB}$

The overall Malicious score of each file which is the quantitative metric is calculated from the CF values. The test environment of the experiment is built using the specifications in Table 3.

Table 3. The Design and Test Environment Specifications

Tool	Specifications/ Details
Computer	Dell T1700, CPU – Intel Xeon@ 3.1GHz, RAM 32GB. Hard Disk – 500GB
Machine OS	Linux Mint 17.1 (#64 – Ubuntu SMP)
Static Analysis Tools	Study Specific Python written scripts that have extracts from Pefile and Peframe

MD5 of the file being analysed	HashFlag Set	Common Factor Method Score	Fuzzy Logic Method Score
364723c338d24d6bea0e28486f96ac9c29169e02	IPFR	85.0147863536	75.6670215578
1c992754f9ef58604aa32191ded21b7648d3e56a		Unknown	
1273683a0a1699064bc5e63a7db4eda73f750213	IPF	77.5449097109	68.8401223241
959ed3a406b97fe94dda96cc3ee8683028484aa6	IPR	77.1864333497	68.4928415872

Figure 2. Results Log.

Table 4. Hash CF calculation from Step 4

		ImpH	PeH	FuzH	ResFH
Detection Rates	TRUE (%)	85.7	83.1	76.7	72.3
	FALSE (%)	14.3	16.9	23.3	27.7
CF		0.256	0.270	0.262	0.241

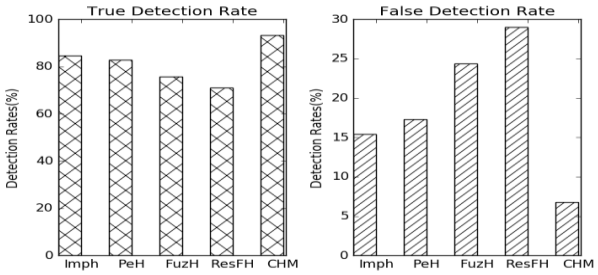

Figure 3: Detection Rates comparison for Individual Hashing Algorithm vs Combined Hashing Method (CHM).

5. RESULTS AND ANALYSIS

The initial detection rates of the individual hash values are calculated and used to get the CF values for each hash as shown in Table 4. Step 5 results into a log shown in Figure 2 and each file has 2 score metrics for each method used. The Unknown results are normalised to 0% for the analysis stage. Comparing the individual hashes against the evidence combinational approach in Table 5 and Figure 3 shows that the detection rates are improved. Since the study introduces a quantifiable metric to measure file maliciousness, investigation into the how the malicious and clean files score are seen in Figure 4 area curves. Common Factor Method has 86% of the clean files scoring below 55% and 80% of the malware scoring above while the Fuzzy Logic method, over 85% of the malware score above 60% and 80% of the clean files score below 55%.

6. CONCLUSION AND FUTURE WORK

This study introduces a fast and efficient way of detecting malicious PE type files using static methods and a quantitative measure for the malicious status of a file. The next step is to further optimize the methods and include a heuristic detection method in the overall malware detection method.

7. ACKNOWLEDGEMENT

Nettitude Ltd are greatly appreciated for all the invaluable support and a thank you to malshare.com for sharing their malware dataset.

Table 5. Detection Rates comparison results for Step 5

		ImpH	PeH	FuzH	ResFH	CHM
Detection Rates	TRUE (%)	84.6	82.7	75.6	71.0	93.2
	FALSE (%)	15.4	17.3	24.4	28.9	6.7

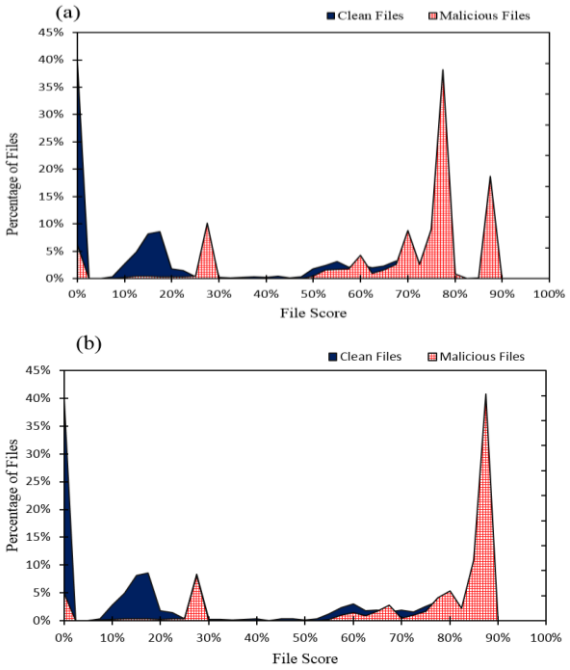

Figure 4. The Combined Hash Score Clean and Malware file Area curves (a) Common Factor method and (b) Fuzzy Logic Method.

8. REFERENCES

[1] A.-T. GmbH, "AV-TEST – The Independent IT-Security Institute." [Online]. Available: https://www.av-test.org/en/statistics/malware/. [Accessed: 19-Jun-2016].

[2] "McAfee Labs Threats Report," Nov-2015. [Online]. Available: http://www.mcafee.com/uk/resources/reports/rp-quarterly-threats-nov-2015.pdf. [Accessed: 01-Feb-2016].

[3] "The Unknown 300 Test Report." [Online].Available: http://www.checkpoint.com/resources/300/300TestReport.pdf. [Accessed 20 April 2016]

[4] "Operating system market share." [Online]. Available: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0. [Accessed: 25-Jul-2015].

[5] J. Pearl, "Decision making under uncertainty," *ACM Comput. Surv.*, vol. 28, no. 1, pp. 89–92, Mar. 1996.

[6] M. M. Gupta and J. Qi, "Fuzzy Logic and Uncertainty ModellingTheory of T-norms and fuzzy inference methods," *Fuzzy Sets Syst.*, vol. 40, no. 3, pp. 431–450, Apr. 1991.

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

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

[9] 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.

[10] B. G. Buchanan and E. H. Shortliffe, Eds., *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Reading, Mass: Addison-Wesley, 1984.