Towards a fast off-line static malware analysis framework

Macdonald Chikapa
School of Electrical Engineering and Computer Science
University of Bradford
Bradford, UK
macwilliam125@gmail.com

Anitta Patience Namanya
Faculty of Science and Technology,
Ankole Western University
anamanya@awu.ac.ug

Abstract— **The profitability in cybercrime activity has resulted into an exponential growth of malware numbers and complexity. This has led to both industry and academic research building malware research labs to allow for deeper malware analysis so that for more efficient detection techniques can be proposed. Extended malware study could lead to development of more advanced malware signatures, potentially resulting into designing of secure systems thus a resilient cyberspace. Malware classification and clustering based on malware families and traits is an important step in malware analysis. This paper presents a comparative study of file format hashes that are used in the industry is conducted in an effort towards suggesting an approach for faster and easier offline malware classification framework.**

Keywords- Malware; Hash ; Clustering; malware detection

I. INTRODUCTION

Access to networking and the internet is now a basic requirement in many individual's lives. Availability of information online that is deemed profitable to many cybercrime masters implies that there is expected growth in the exploitation of these networks and information centres. Malware is one of the major vectors used to compromise and exploit vulnerabilities in most networks. Anti-malware systems are now a stable for any network or end point systems, however, they rely on signatures which means that malicious files needs to initially be discovered, analysed and then have its detection signature written and uploaded to the anti-malware database. With 390,000 malware samples being discovered daily [1] the concern about how the cybersecurity environment will keep up with growing number of malware is validated. For such huge number, there is a need for fast offline clustering methods so that emerging trends in the malware evolution can be studied and counter measures designed to mitigate the negative effects. This study will utilize existing static analysis tools to automate analysis of the samples to provide some static signatures. It further explores the classification hashes used in the development of this framework to provide a comparative analysis of their clustering efficiency and detection rates.

II. MALWARE

Malware short for Malicious Software refers to software that causes the system to behave differently from its intended behaviour. Such a generalized definition is necessary to most malware categories as malware activity keeps evolving with the rapid changes in the technology industry. Comprehensive understanding of malware is necessary since malware compromises the 3 tenets of information security namely confidentiality, integrity and availability[2].

III. EVOLUTION OF MALWARE

Malware evolution has roughly followed advancement in technology, from computers, to computer networks, to mobile devices, to the cloud, industrial systems and now on to Internet of things. Previously, malware was isolated to desktop computers and computer networks, primary because those were the only devices with processing chips. As a result, one of the earliest malware was a computer virus named Elk cloner[3]. This virus was relatively harmless, since it only displayed a poem on a computer screen. However, subsequent viruses were not as benign. An example of such virus was the Jerusalem virus which could delete executable files on infected host[4]. These viruses were limited to individual works stations.

In 1988, there was the Morris worm which took advantage of the interconnection between networks to spread itself[5]. The morris worm was different from prior malware in that it in addition to self-replication, it could propagate across computer networks autonomously. This resulted in performance degradation of the infected computers and overwhelmed network bandwidth so much that the author of this malware was prosecuted[6]. This is recognised to be the first malware in many of both academic and industrial works as it changed the known landscape of nuisance software to software with specially designed malicious intent.

Emergence of mobile devices provided malware writers with another target. The earliest mobile device malware, named cabir, was a worm which targeted the Symbian operating system, a widely used mobile OS at the time[7]. This malware was benign since it only displayed a message after it got executed on a mobile device. However, subsequent mobile malware, such as skulls went on to delete information from the mobile devices.

Cloud computing defines a new computer use model where computing resources are pooled together and consumed on demand as a utility similar to electricity and water [8]. Cloud computing environments could be private or public [9] neither one of them have been spared the scourge of malware since cloud environments are basically containers holding multiple computers vulnerable to the same malware targeting individual computers. With bitcoin

978-1-5386-7810-7/18/$31.00 ©2018 IEEE
DOI 10.1109/W-FiCloud.2018.00035

mining so lucrative, these pooled resources offer the perfect targets. Internet of things addresses the interconnection of day personal devices, such as wrist watches, cars and refrigerator, to the internet[10]. This connection presents another target for malware writers and hackers. The 2015 jeep hack displayed the vulnerabilities of internet of things. The potential loss of life in this hack resulted in the manufacturer recalling thousands of cars in order to apply a software fix[11].

Industrial control systems have not been safe either [12]. Some industrial control systems control the electrical grid and their disruption could lead to loss of lives in hospitals due to lack of electricity.

As cybercrime because more profit oriented, many enterprises have been the target of various malicious attacks that lead to heavy paydays as seen in the most recent swift attack on the Bangladesh bank where the attacker walked away with $81 million. Advanced Persistent targeted attacks that are specially designed for a system are now the bane of many company information security personnel.

Still, with the Internet of Things evolution, it is only expected that many more systems will be compromised using malware as now, there will be many more viable entry points to any network. Figure 1 shows the main evolution milestones of the malware world.

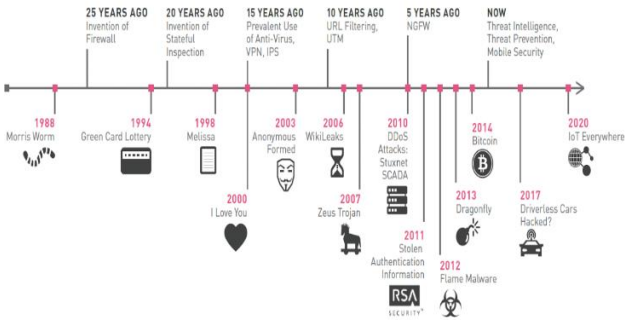

Figure 1. Evolution of malware [13]

IV. MALWARE ANALYSIS TECHNIQUES

Malware analysis involves dissecting a file sample in order to understand its behaviour and therefore its intent when introduced into a system. The expected result at the end of the analysis is to be able to devise ways in which to stop it from infecting systems and methods of removing it from systems it has already infected. [14]. Mitigation efforts could be range from end user training in order to avoid infection to development of detection signatures. There exist 2 main categories of malware analysis, static and dynamic.

A. Static malware analysis

Static malware analysis refers to analysis of malware without executing the source code. This analysis depends on features inherent on the malware code itself. As mentioned earlier, detection rates for static detection are relatively lower than dynamic due to proliferation of malware obfuscation[15]. Another disadvantage is that static malware requires access to a signature database for malware evaluation. Since there exist millions of malware variants, updating the database and keeping it small is a challenge. An example of is the fact that in 2011, ClamAV anti-virus sent out about 120TB of malware signatures every day due to the sheer number of malware and their variants[16].

Another limitation of static malware analysis is that, at times it is not possible for the analyst to reverse engineer the malware, hence, no access to the source code for analysis.

With static malware detection efforts, malware analysts, evaluates various data sections of malware to extract malware signs. Some of these sections could be file headers and class names.

Whilst static malware analysis may not be fruitful in complex malware cases, it still remains an important initial malware analysis step as information extracted during this stage can assist a malware analyst deduce the next step or even conclude with certainty that a file is malicious based on initial knowledge.

B. Dynamic malware analysis

Due to obfuscation techniques mentioned earlier, dynamic malware analysis is often necessary to determine malware effects and mitigation. With dynamic malware analysis, malware is executed and its activities monitored [14]. Dynamic malware analysis is time consuming and often requires specialised tools to accomplish. Increasing malware complexity, hence requiring dynamic analysis, is a tactic meant to waste malware analyst resources, thereby increasing chances for the malware to achieve its goals.

Some of the tools used in Dynamic analysis include REMnux and OllyDbg. Dynamic malware detection involves executing a suspected malware sample and observing its behaviour as it is running. Dynamic malware evaluation usually occurs after static malware analysis, in order to analyse malware samples which have been obfuscated making it almost difficult for static analysis. Dynamic analysis requires specialised tools and is fraught with many risks ranging from accidently attacking one's own network to accidently attacking another workstation on the internet by accident [17]. Ideally, a separate network with a separate internet connection should be configured to isolate network traffic from the main network.

Recently, there has been a growth in the development of automated dynamic analysis tools that provide online sandboxing analysis and/or already configured systems that can be used for sample analysis. An example of such a system is cuckoo which has the malwr online analysis tool and the cuckoo sandbox product available to malware analysts. Although this provides a way of making dynamic analysis faster and more efficient, complex malware samples are known to thwart these systems by detecting the dynamic analysis environment and then executing in a dubious way to mislead the analyst. The execution of samples in these environments also requires time.

V. MALWARE DETECTION TECHNIQUES

Malware detection techniques are methods employed in systems as ways of finding malware in the system or network. The intention is to uncover the malware in order to

mitigate their effects or to enable the start of system recovery processes. There are divided into 3 categories; Signature based detection, Specification based detection and Anomaly based detection. All the three can be built from using results achieved from either static, dynamic or hybrid (combination of both static and dynamic) analysis.

A. Anomaly-based Detection

Anomaly-based is a malware mitigation tactic which aims to detect presence of malware inside a system before humans[18]. With anomaly-based detection, baseline metrics of the systems are measured over a given period and then saved in a database. After that the system is then constantly monitored for any significant deviation from the baseline. Examples of deviations could be new destination ip addresses in the system, which could indicate a malware inside the system communication with a malware command and control centre [19].

B. Specification based detection

With specification based detection, expected system behaviour and properties is catalogued, any deviation from the specifications results in an alert about possibility of system compromise[20]. Approaches based on specification based detection build specified system behaviour based on expert knowledge of the system using the system least privileged principle. This approach in theory can be used to detect unknown attacks since any deviation from the known behaviour triggers an alert. However, building a system that includes all real system behaviours is such a difficult and challenging task that achieving a perfect specification based detection system is an impossible task.

C. Signature based detection

Malware signatures are utilised by malware detection engines to detect presence of malware on a system, through comparison of the suspected malware and the malware signature database. Despite limitations in their inability to detect new patterns of attacks, and challenges when dealing with obfuscated malware, signatures reduce work to be done by malware analyst by eliminating already know samples of malware. Further time reduction can be achieved by automating malware signature generation. [21].

There exist a couple of malware signature types mostly based on futures utilised in the signature generation.

1) Hash Signatures

With hash signatures, hash values are generated for selected features on malware samples. Once the hashes are generated they are uploaded into a database for malware detection systems to check against future infections[22]. A weakness of hash signature is that each slight variation in the file structure or content can lead to a completely different hash which results in bloated signature database[23]. Hash signatures are however used in triaging and clustering malware as discussed later in this paper.

2) Byte Signatures

Another option for signature generation involves analysis of sequence of bytes of the malware. This analysis operates on the assumption that malware from the same family, will possess similar byte sequences enabling malware detection programs to detect them[22]. Evaluation of byte sequences can potentially enable scanning systems to detect new classes of malware and zero-days[23]. This is possible since malware from the same author can, potentially, possess traits common to both making it easier to detect. Such features can range from same encryption code for polymorphic malware, same destination IP address and utilization of same software resources.

3) Binary Diffing

Binary diffing involves analysis of code to identify sections similar to known malware code[24]. If there is greater similarity the code is marked as malware and then used as a digital signature. It would appear that the possibility for false positives and false negatives since a malware sample not matching with any known sample would be flagged as negative. Such a scenario would require addition manual analysis and generation of a new signature to address such a shortfall.

D. New approaches to malware detection

Despite relentless efforts to improve malware detection, the number of malware keeps rising exponentially as mentioned earlier. This proliferation of malware has led to research for newer approaches to malware detection. One such approach involves use of machine learning methods for malware detection [25]. Machine learning involves dividing data into sets, with one set being the training set and having the detection system learning from this training set in order to detect malware.

Other approached include using data mining to detect patterns of malware in pieces of code[26]. Datamining utilizes statistical analysis for malware detection. This is could be useful given the large sample sizes presented by thousands of malware variants.

VI. Malware Clustering Based on Hashes

Malware clustering is a known first step to enable triaging malware that allows for a faster and more efficient malware analysis processes. Clustering refers to mechanism for measuring similarities between samples and grouping them based on those similarities[27]. In malware analysis, clustering enables researchers group malware into 2 classes, one class being malware they have analysed before or variants of the known samples, and another class comprising of unknown malware. This enables the triaging of the malware samples discovered in the wild daily. With the huge numbers that are discovered daily in the wild; 390,000 according to AV-test institute, this process allows malware analysts to concentrate their efforts on the unknown malware samples in order to keep up with the numbers considering the lack of skilled malware analysts.

Before checking for similarity, features to be checked have to be selected first. Examples of such features could be malware file size or import strings in malware source code[38].In other words, clustering is meant to uncover which features are common to all members of a malware class. Other features used for malware classification are length of malware function[29] and malware strings. Indeed,

some researchers have transformed malware into images and generated clusters after the transformation[30]. Combination of malware features can result in high quality clusters, resulting in fewer false positives or false negatives. One of the popular methods is the use of similarity matching hashes which have long been used in forensic computing to match images and for spam filtering. This works as a filtering process where the malware samples discovered from the wild are sieved so that malware can be detected and classified in an effort to skim off the different malware families that can be used to formulate malware family specific signatures. One of the methods this is achieved is by using similarity matching hashes with the assumption that files that generate hashes with high similarity match percentages belong the same family.

A. *Related work*

While conducting this study, various works that have conducted comparative studies on some of these hashes were reviewed. Hashing algorithms that have been used in computer forensics have been adopted in malware analysis as a way of malware classification and clustering[37] . Investigations on hashes like the ssdeep and sdhash have been studied to determine how well they perform when used in malware classification. Results from the studies [35], [36], [38] show that these hashes are very useful in triaging malware which can assist in generating malware family based detection signatures. This can also be a useful tool in filtering out variants of already known malware. Although these hashes have very important use in malware classification, they are plagued by high false positives [39] thus making them unsuitable for malware detection. Some of the rest works have shown researcher building new hashing algorithms for malware detection that use the same principle of similarity matching in order to improve the achievable detection rates[36] .

File format based hashes that are targeting at improving malware similarity matching in obfuscated malware have also been development in various studies. PeHash [47]and impHash [33] are one of the hashes that have been developed to allow for similarity matching for obfuscated portable executable (.exe) files. The analysis of PeHash in the development paper [47] provides an initial analysis of the algorithm while [33] introduces the algorithm and [34] provides the initial analysis study into how well impHash performs in malware family clustering. These studies provide an argument for a need for further comparative analysis of the various malware clustering hashes. Part of this work investigates the performance of these two specific hashes when used for malware clustering and comparing the two.

VII. COMPARATIVE ANALYSIS OF MALWARE CLUSTERING BASED ON HASHES.

In this section, a comparative analysis study is presented and discussed on two identified hashes feature based hashes; PeHash [31] and Imphash [32]. The file Sspecific hashes analysis is designed in order to analyse and compare the effectiveness of the clustering achieved based on the hashes

A. *Experimental Approach and setup*

In this study, PeHash and impHash are compared to see which one out performs the other in malware family clustering. These hashes are used mainly for portable executable files and so our experimentation is limited to files of PE file format (.exe file). The 2 hashes of each file in the experiment dataset are computed and then the true family clustering matches are compared for each family. This comparative analysis experiment processes are represented by figure 2.

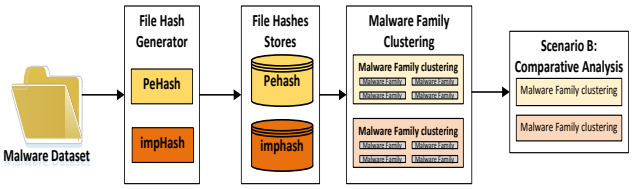

Figure 2. Representation of the Comparative study.

B. *Datasets Collection and formulation*

Since this work is in collaboration with the cyber security research group at the University of Bradford, the malware samples used in this study were ones already collected and identified by the research group. Due to time restrictions, a total number of 1476 samples are used in this study and Table 1 shows the malware family formulation of the dataset used.

TABLE 1. DATASET MALWARE FAMILIES' FORMULATION

Malware Type	Percentage
Win.Adware.Somoto-1	2.7%
Win.Adware.Somoto-4	1.8%
Win.Worm.Kido-307	1.0%
Win.Trojan.Outbrowse-1453	65.8%
Win.Adware.Kraddare-26	0.5%
Win.Malware.Bedep	11.1%
Win.Worm.Kido-200	4.7%
Win.Worm.Agent-35431	3.6%
Win.Worm.Kido-113	3.4%
Win.Worm.Kido-197	3.3%
Win.Trojan.Agent-13574	2.1%

C. *Test Environment*

When designing the test environment, since the work focuses on executable files, a Linux box was used with the specification shown in Table 2. The tools and programs used on the test environment are also defined. The scripts used in this study are python scripts that are written to provide the various hashes of the file that are implemented within the database.

TABLE 2. TEST ENVIRONMENT SPECIFICATIONS

Tool	Specifications/ Details
Machine OS	Ubuntu 16.04
Static Analysis tools	Scripted Automated tools – calculates the PeHash, and ImpHash of the file using the PeHash and Pefile modules
Scripting Language	Python IDLE version 2.7.9

A python script is used for the hash calculation. The output is saved to text files for easier initial data handling. The clustering studies are done within an excel environment for better data analysis and graph plotting.

TABLE 3. FILE-FORMAT SPECIFIC MALWARE CLUSTERING.

Malware Type	PeHash	ImpHash	PeHash	ImpHash
Win.Adware.Somoto-1	40.00	40.00	100%	100%
Win.Adware.Somoto-4	26	26	100%	100%
Win.Worm.Kido-307	4	15	27%	100%
Win.Trojan.Outbrowse-1453	971	971	100%	100%
Win.Adware.Kraddare-26	7	7	100%	100%
Win.Malware.Bedep	164	164	100%	100%
Win.Worm.Kido-200	47	70	67%	100%
Win.Worm.Agent-35431	22	53	42%	100%
Win.Worm.Kido-113	40	50	80%	100%
Win.Worm.Kido-197	26	49	53%	100%
Win.Trojan.Agent-13574	29	30	94%	97%
Total				

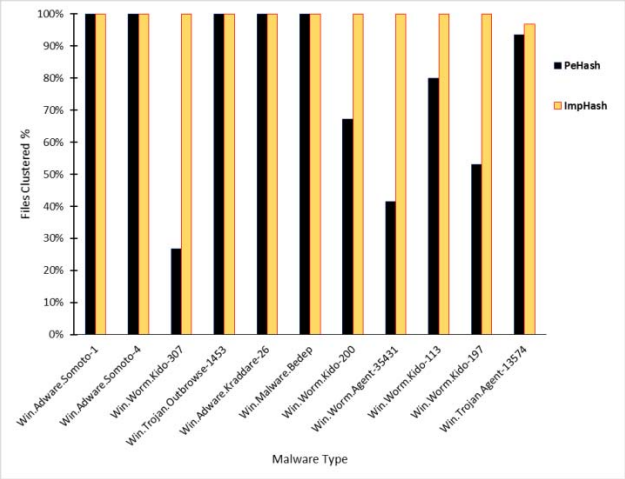

Figure 3. Hash-based Malware clustering comparison

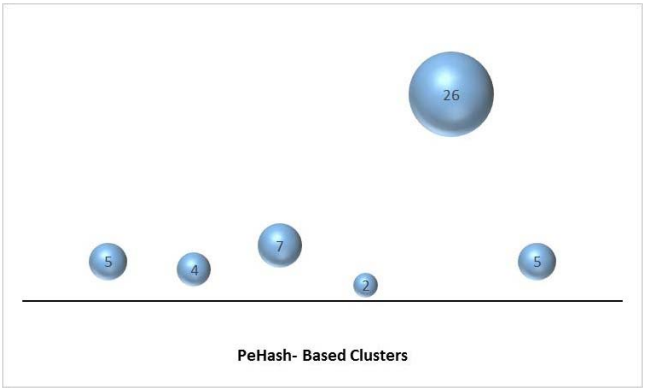

Figure 4. PeHash Clusters for Win.Worm.Kido-197.

D. *Classification and Clustering with Hashes Experimentation Results and analysis.*

The classification study looked at the effectiveness of the clusters achieved by each hash type. Using the malware dataset in Table 1, the biggest cluster achieved by each hash is considered. From Figure 3 and Table 3, it is clear that impHash, out performs PeHash. However, further analysis shows that PeHash has the ability to create smaller clusters out of malware families. For example, considering malware type Win.Worm.Kido-197, while impHash only groups all the malware into one cluster, PeHash creates various clusters as seen in Figure 4.

This is seen in the various malware types where there is no 100% malware clustering. This shows that there is a possibility that PeHash could provide clusters that can provide cluster for further detailed studies especially for cases of malware detection evasion techniques. While this is not considered in this study, this is part of an analysis that can be done in future work.

VIII. CONCLUSION AND FUTURE WORK

A. *Conclusion*

In this study, impHash and PeHash are compared for the classification of known malicious portable executable files with known families. A dataset of 1476 malicious samples which belong to 11 different families based on ClamAV engine is used to perform the study. While it is clear that impHash out performs PeHash in terms of malware type clustering from this study, PeHash is seen to be able to create various clusters for further evasion techniques analysis for Portable executable malware.

B. *Future Work*

During the study, only the file format specific hashes were studied. While impHash outperforms PeHash, PeHash shows to create various clusters for one malware family. In the future work, it would be worth exploring the reasons for different clusters for one malware family. Furthermore, another area that is worth exploring and experimented upon is the comparative study of the various similarity hashes as applied to malware clustering in order to verify the results achieved by the reviewed related works.

[1] P. Paganini, "AV-TEST estimates 12 million new malware variants per month," *Security Affairs*, 17-Jan-2015. [Online]. Available: http://securityaffairs.co/wordpress/32352/malware/av-test-statistics-2014.html. [Accessed: 13-Sep-2016].

[2] R. J. Boyle and R. R. Panko, *Corporate Computer Security, Global Edition*. Pearson Education Limited, 2015.

[3] E. H. Spafford, K. A. Heaphy, and D. J. Ferbrache, "A computer virus primer," 1989.

[4] H. J. Highland, "A history of computer viruses—The famous 'trio,'" *Comput. Secur.*, vol. 16, no. 5, pp. 416–429, 1997.

[5] T. Eisenberg, Id. Gries, J. Hartmanis, D. Holcomb, and M. S. Lynn, "The Cornell commission: on Morris and the worm," *une*, vol. 32, no. 6, 1989.

[6] P. G. Neumann, "Inside risks: insecurity about security?," *Commun. ACM*, vol. 33, no. 8, pp. 170–170, 1990.

[7] M. Piercy, “Embedded devices next on the virus target list,” *Electron. Syst. Softw.*, vol. 2, no. 6, pp. 42–43, 2004.

[8] M. Armbrust *et al.*, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[9] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[11] A. Greenberg, “After Jeep Hack, Chrysler Recalls 1.4M Vehicles for Bug Fix,” *WIRED*, 24-Jul-2015. [Online]. Available: https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/. [Accessed: 27-Jul-2016].

[12] “New malware dubbed Irongate uncovered, targets industrial control systems,” *International Business Times UK*, 06-Jun-2016. [Online]. Available: http://www.ibtimes.co.uk/irongate-malware-that-targets-industrial-control-systems-uncovered-1563895. [Accessed: 27-Jul-2016].

[13] “2015 Check Point’s Annual Security Report.” .

[14] M. Sikorski and A. Honig, *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press, 2012.

[15] P. OKane, S. Sezer, and K. McLaughlin, “Obfuscation: the hidden malware,” *IEEE Secur. Priv.*, vol. 9, no. 5, pp. 41–47, 2011.

[16] S. K. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, and D. G. Andersen, “SplitScreen: Enabling efficient, distributed malware detection,” *J. Commun. Netw.*, vol. 13, no. 2, pp. 187–200, 2011.

[17] K. Kendall and C. McMillan, “Practical malware analysis,” in *Black Hat Conference, USA*, 2007, p. 10.

[18] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Comput. Secur.*, vol. 28, no. 1, pp. 18–28, 2009.

[19] L.-Y. Yeh and Y.-L. Tsai, “An Automated Framework for Command and Control Server Connection and Malicious Mail Detection,” *ICNS 2015*, p. 15, 2015.

[20] C. Ko, M. Ruschitzka, and K. Levitt, “Execution monitoring of security-critical programs in distributed systems: A specification-based approach,” in *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, 1997, pp. 175–187.

[21] S. Chaumette, O. Ly, and R. Tabary, “Automated extraction of polymorphic virus signatures using abstract interpretation,” in *Network and System Security (NSS), 2011 5th International Conference on*, 2011, pp. 41–48.

[22] Hanel, Alexander, “Hooked on Mnemonics Worked for Me: An Intro to Creating Anti-Virus Signatures:,” *An Intro to Creating Anti-Virus Signatures:* .

[23] K. G. S. S. X. Hu and T. Chiueh, “Automatic Generation of String Signatures for Malware Detection,” *Symantec Res. Lab.*, pp. 1–29, 2008.

[24] J. Oh, “Fight against 1-day exploits: Diffing binaries vs anti-diffing binaries,” *Black Hat Black Hat*, 2009.

[25] J. Sahs and L. Khan, “A machine learning approach to android malware detection,” in *Intelligence and Security Informatics Conference (EISIC), 2012 European*, 2012, pp. 141–147.

[26] Y. Fan, Y. Ye, and L. Chen, “Malicious sequential pattern mining for automatic malware detection,” *Expert Syst. Appl.*, vol. 52, pp. 16–25, 2016.

[27] L. Rokach and O. Maimon, “Clustering methods,” in *Data mining and knowledge discovery handbook*, Springer, 2005, pp. 321–352.

[28] C. S.-E. T. Johnson, “Improved Malware Clustering Using VirusTotal Metadata,” 2013.

[29] R. Tian, L. M. Batten, and S. C. Versteeg, “Function length as a tool for malware classification,” in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, 2008, pp. 69–76.

[30] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, “A comparative assessment of malware classification using binary texture analysis and dynamic analysis,” in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 21–30.

[31] 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.

[32] “Tracking Malware with Import Hashing « Threat Research,” *FireEye*. [Online]. Available: https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html. [Accessed: 18- July-2016].

[33] “Tracking Malware with Import Hashing,” *M-unition*. [Online]. Available: https://www.mandiant.com/blog/tracking-malware-import-hashing/. [Accessed: 14-Jul-2016].

[34] 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.

[35] David French and William Casey, “Fuzzy Hashing Techniques in Applied Malware Analysis,” Results of SEI Line-Funded Exploratory New Starts Projects CMU/SEI-2012-TR-004, Aug. 2012.

[36] 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.

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

[38] DigitalNinja., “Using Fuzzy Hashing Techniques to Identify Malicious Code,” Apr. 2007.

[39] V. Roussev, “An evaluation of forensic similarity hashes,” *Digit. Investig.*, vol. 8, Supplement, pp. S34–S41, Aug. 2011.