PE File-Based Malware Detection Using Machine Learning

Chapter · January 2021

DOI: 10.1007/978-981-15-4992-2_12

2 authors, including:

Prachi Chaudhary
The Northcap University
44 PUBLICATIONS **187** CITATIONS

SEE PROFILE

PE File-Based Malware Detection Using Machine Learning

Namita and Prachi

Abstract In current times, malware writers write more progressive sophisticatedly designed malware in order to target the user. Therefore, one of the most cumbersome tasks for the cyber industry is to deal with this ever-increasing number of progressive malware. Traditional security solutions such as anti-viruses and anti-malware fail to detect these advanced types of malware because the majority of this malware are refined versions of their predecessor. Moreover, these solutions consume lots of computational resources on the host to accomplish their operations. Further, malware evades these security solutions by using intelligent approaches such as code encryption, obfuscation and polymorphism. Therefore, to provide alternatives to these solutions, this paper discusses the existing malware analysis and detection techniques in a comprehensive/holistic manner.

Keywords Malware · Static analysis · Dynamic analysis · PE files · Machine learning

1 Introduction

Internet technology is fully integrated with every domain worldwide. Due to its widespread usage, lots of sensitive data about users is available online. Attackers use various malware like viruses, worm, rootkit, Trojan, bots, spyware, ransomware and so on in order to perform lots of malicious activities. Malware [1] is a generic term that is used for any kind of software designed to disrupt the system operations, access to remote systems, flood the networks, delete/modify data, corrupt the hardware/software and collect personal information without authorization. The scope of

Namita · Prachi (✉)
Computer Science and Engineering Department, The NorthCap University, Gurugram, India
e-mail: prachiah1985@gmail.com

Namita
e-mail: namita.dabas@gmail.com

P. Bansal et al. (eds.), *Proceedings of International Conference on Artificial Intelligence and Applications*, Advances in Intelligent Systems and Computing 1164,
https://doi.org/10.1007/978-981-15-4992-2_12

malware is not limited to stand-alone systems; it can also propagate through multiple channels of communications and can be downloaded. In addition, most of the malware can change their native code to prevent their detection. Modern-day malware comes with obfuscation techniques such as polymorphism, metamorphism and encryption. Further, if a malware code is available in public, then anyone with malicious intent can manipulate the code and add some extra features to generate new malware even without much programming language. This allows attackers to recreate more sophisticated versions of pre-existing malware.

As per Malwarebytes Labs 2019 reports [2], in the year 2018, malware authors shifted their focus from individual consumers to business organizations in order to gain high profits. Overall, malware in business rose significantly, i.e., 79% in 2018-primarily due to the increased activities of malware such as backdoors, miners, spyware and information stealers. The statistics released by Kaspersky [3] identified 243,604 users, which were attacked by malware just to steal the money. Further, the data from AV-Test [4] claims that 914.79 million malware samples were reported in the year 2019. Therefore, malware detection and prevention have become the primary objective of the security companies as well as researchers in the recent past.

As the numbers of malware are increasing exponentially, it became impossible to manually analyze and detect them. Consequently, a number of security companies are providing various anti-virus and anti-malware solutions to protect legitimate users from such attacks. Some of the popular security solutions provided are as listed: Microsoft security essentials, Avast antivirus, AVG technologies Comodo, Kaspersky, Norton, windows defender, McAfee, Malwarebytes, ESET Nod 32 and so on [5]. However, the majority of anti-viruses and anti-malware solutions are based on signature-based methods.

Signature is a short unique sequence of bytes, often distinctive to a malware [6]. The signature-based detection approaches use a database of predefined signatures. This type of scheme works well for previously known malware but fails to detect new malware or variants of existing malware. Nowadays, malware developers use automated malware generation tools based on encryption, obfuscation, polymorphism, metamorphism, packing and emulation techniques. These tools provide an edge to the black-hat community over the white-hat community and facilitate malware writers to write and change the code of already-written malware. This newly written malware possesses new signatures. Every new development to detect malware is followed by evasion methods. So, there is an urgent requirement of developing alternative analysis and detection methods to address the shortcomings of signature-based solutions. Before diving deep into the discussion, it is important to highlight that the malware analysis is majorly classified into the two main categories.

1.1 Static Analysis

In static analysis, the malware samples are analyzed without the code execution [6]. Static analysis is preferred when a quick decision needs to be made in a resource-efficient manner. The effectiveness of static analysis relies on the comprehensiveness of analysis and parameters chosen for analysis. Static analysis includes analysis of source code, hashes, header information, strings, metadata, etc. Simple static analysis remains largely ineffectual against sophisticated malware because it may miss important functionality of the malware. Advanced static techniques involve reverse engineering of malware through disassembler tools such as IDA Pro [7], OllyDbg [8] and Olly Dump [9] to understand the code of malware. Some other tools like memory dumper LordPE [10] help to obtain data about changes from the system's memory. These patterns are further analyzed based on features, such as binary code, opcodes, strings, byte n-grams and control flow graphs. The static analysis is used in signature-based detection methods. The static analysis-based approaches are easy and fast but cannot detect ever-evolving obfuscated malware correctly as they leave some of the functionalities unidentified. The limitations of static analysis have been explored [11]. These limitations provide an edge to a dynamic analysis-based approach.

1.2 Dynamic Analysis

In dynamic analysis, the malware samples are executed and analyzed in a real or a controlled environment. Dynamic analysis can be done using a variety of debuggers. Various other tools such as Process Monitor, Regshot, Filemon and Process Explorer can also be used to retrieve behavioral features such as API, system calls, instruction traces [12], registry changes, file writes, memory writes and network changes. Dynamic analysis is majorly used for understanding the functionalities of malware samples under consideration. However, environment aware malware does not exhibit their true behavior whenever they identify that they are being executed in a controlled environment. Therefore, the dynamic analysis methods must possess some real environment characteristics in order to make it challenging for malware to distinguish between real and controlled environment. There are numerous online sandbox environments available for dynamic analysis. Some of the widely used sandboxes are CW Sandbox, Cuckoo Sandbox, Anubis, Norman Sandbox, etc. Dynamic approach exposes the natural behavior of the sample under examination. However, the dynamic analysis approach is time- and resource-consuming (memory overheads) as each sample under investigation needs to be executed separately and comprehensively.

Both the analysis techniques are beneficial under different settings but when the objective is to perform the analysis in a timely manner while consuming minimum resources then the static analysis is preferred over dynamic analysis.

Therefore by taking the above fact into consideration, some of the researchers combined the features of static and dynamic analysis and proposed a new method, termed as hybrid analysis. The authors in [13] used dynamic analysis for extracting the features for training and static analysis to test the efficiency of the detection system.

Machine learning algorithms facilitate us to design an automated malware analysis and detection system that can precisely distinguish malware as benign or malicious. These techniques significantly increase the detection efficiency and at the same time reduce the time and resource consumption. New malware can also be detected easily with these intelligent systems. Therefore, a large number of authors have used machine learning algorithms to train and test their models designed using the extensive set of features extracted from large no of benign and malicious samples using various static and dynamic analysis techniques.

The aim of this paper is to discuss and review the malware analysis of PE files. PE files are chosen in this paper because they work on the Windows operating systems and to date Windows is the most commonly used OS (77.93%) by the users all across the world [14]. PE is a 32/64 bit file format for Windows OS executables, object codes, DLLs and others. Malware analysis of PE files can be done with a variety of features as byte sequences, strings, information flow tracking, opcodes, control flow graphs and API calls and so on.

This paper is organized into four sections: Sect. 1 presents a general view about the malware industry, recent trends of malware attacks and type of malware analysis and detection approaches. Section 2 provides some insights into malware detection techniques based on machine learning methods present in the literature. Section 3 discusses the different dimensions of the reviewed work, and Sect. 4 discusses the conclusion and the future directions.

2 Related Work

Numerous malware detection techniques have been proposed in the literature based on machine learning algorithms. Some of the machine learning-based research work related to PE file malware analysis is discussed here.

In 2001, Schultz et al. [15] proposed a machine learning framework for detecting malicious PE files using static features namely PE header, strings and byte sequences. The dataset was divided into two subsets: (1) training dataset for training the classification model, (2) test data set to assess the classifier for unknown binaries. Three machine learning algorithms (Ripper, Multi-Naïve Bayes and Naïve Bayes) were employed for the classification process. The authors used a dataset that contained 4266 samples (3265 malware + 1001 benign). The detection accuracy (97.1%) of the proposed system was higher in comparison with already existing signature-based detection systems of that time.

A malware detection method for PE files was proposed in 2011 [16] based on the graph analysis technique. The static features used for analysis included raw binaries

and opcodes, using *n*-gram approach, whereas the dynamic features used for the analysis included instruction traces, control flow graphs, and system call traces. The authors trained the system with a dataset of 776 benign and 780 malicious samples. A support vector machine algorithm was used as a classification algorithm, and the multiple kernel learning approach was applied to find the similarity index between graph edges.

In 2013, Eskandari et al. [17] proposed a malware detection system using dynamic features of PE files, i.e., API and system calls. The authors used the Bayesian classification model in order to classify benign and malicious PE files.

Khodmoradi et al. [18] presented heuristic based detection model for metamorphic malware in year 2015. The authors used static features (opcodes) for the analysis. They disassembled the file using IDA pro and extracted the features using opcode statistics extractor. Six classification algorithms (j48, j48graft, LADTree, NBTree, Random Forest, REPTree) were used to classify 1200 samples into benign and malicious. The authors highlighted that the classification accuracy of their detection system is dependent on the classification methods applied and the disassembler chosen.

In the year (2015), Lakhotia et al. [19] surveyed on various existing malware detection and prevention techniques. The authors discussed the importance of applying machine learning techniques for malware detection.

Liang et al. [20] suggested a behavior-based malware detection technique in 2016. The authors extracted dynamic features of PE files from API calls, registry and network files. They applied supervised machine learning and trained the system using Jaccard similarity distance to identify the different variants of malware.

In 2017, Baldoni et al. [21] also utilized static analysis techniques to extract sequences, strings and headers from PE files. The authors trained the system with a dataset of 4783 samples using RF classification algorithm. Their proposed system provides a faster and more accurate (96%) analysis as compared to other detection systems.

The authors in [22] came up with a detection system for different variants of malware by predicting their signatures (2017). Their solution primarily focuses on the static analysis of PE files. The feature sets used in the analysis included strings, *n*-grams, API calls and hashes. Unsupervised learning-based machine learning algorithms were used in the training phase.

In 2017 [23], the authors trained the hidden Markov model using static as well as dynamic analysis to compare their detection results on malware variants of different families. [6] provided an extensive study of malware detection approaches based on data mining techniques in the year 2017. They examined the different dimensions of analysis including the feature sets and classification/clustering algorithms.

In the year 2018, [24–28] presented their solutions for malware detection on the basis of analysis (static, dynamic and hybrid) methods and classification algorithms with the applications of data mining algorithms.

The authors applied supervised learning-based machine learning algorithms to their proposed detection system [29]. They developed a static analysis-based automated tool to extract features of PE files. Later, they trained the system with classification algorithms such as SVM, decision trees and applied boosted decision tree algorithms to increase the detection efficiency of the proposed system.

A survey of malware detection techniques was carried out based on the windows platform [21]. Their study was a meta-analytical account of various researches pertaining to machine learning. The authors arranged studied literature based on their primary objectives, features and machine learning algorithms. The primary objectives were further classified into three sub-categories, i.e., malware detection, find in similarities and malware category detection. The researchers also highlighted various limitations and challenges faced by detection methods.

In 2016, [30] developed a detection solution based on machine learning technique. The authors applied both static and dynamic analysis methods for feature extraction. The developed a classifier model using seven classification algorithms (Random Forest, Naïve Bayes, J48, DT, Bagging, IB1, and NLP) and trained the classifier with 3130 PE files dataset. The detection accuracy of their proposed system was best (99.97%) when the RF classification algorithm was applied.

A machine learning-based framework (Virtual Machine Introspection) for detecting malware in virtual machines was proposed [31]. The authors extracted opcodes using static analysis and trained the system with selected features for providing better accuracy. Further, they applied Term Frequency-Inverse Document Frequency (TF-IDF) and Information Gain (IG) as classification algorithms.

In the year 2019, similarity hashing algorithm-based malware detection technique in the IoT environment was proposed [28]. In this technique, malware file scores were calculated to find the similarity between malware samples. The authors used the PE dataset and explored four different hashing techniques (PEHash, Imphash, Ssdeep, resource section Ssdeep). Finally, they combined the results of these hashes using evidence combinational methods such as fuzzy logic and certainty factor model.

This paper also presents a few of the various malware detection approaches for PE files as given in Table 1. After performing the in-depth study of these existing methodologies, it is inferred that opcodes provide low-level details of the executable and hence provides a better opportunity for detection of obfuscated malware at run time. As a matter of fact, it may be concluded that opcode-based analysis provides a better solution as compared to other malware detection solutions.

3 Discussion

It is evident from the above-stated literature that most of the malware detection techniques for PE files are based on machine learning algorithms. This segment briefs a statistical analysis of machine learning algorithms for the reviewed detection methods. Figure 1 shows that 56% of the above-studied methods used supervised learning-based algorithm, 26% have applied unsupervised learning algorithms and

Table 1 List of the reviewed works for malware detection using machine learning

Authors	Datasets	Features used	Extraction method	Approaches	Advantages	Limitations
Anderson et al. [16]	Dataset—15,56 1 samples	Binary, CFGs, opcodes, instruction traces, system calls	Static dynamic	Multiple kernel learning, machine learning	Average Detection Accuracy-97%	Instruction categorization not optimal
Raff et al. [29]	10,868 files (Kaggle + (Derbin)	Byte sequences	Static	Machine learning	Better for large byte sequences, 95% accuracy	Obfuscation reduces detection accuracy
Liang et al. [20]	200 samples	API calls, registry, PE files, network	Dynamic	Multilayer dependency chain, Jaccard similarity	Faster approach, better detection accuracy 70%	Sample size is small
Vadrevu et al. [30]	M-1,251,865 B-400,041	Network traces, PE files	Static	Clustering DBSCAN algorithm	50% time saving with 0.3% information loss	May not work with stalling code
Polino et al. [31]	2136 (1272 M + 864 B)	API/system calls	Hybrid	Unsupervised: Clustering with Jaccard similarity	Automatically finds API call groups (88.46% accuracy)	Obfuscated/packed malware reduces the accuracy, small data set
Miramirkhani et al. [32]	270 real machine users	Disk, registry, network, browsers	Static	Regression techniques	Useful for environment-aware malware	Ineffective with obfuscated malware
Blazytko et al. [33]	1200 randomly generated expression	Byte code, instruction tracing	Dynamic	Monte Carlo tree search, random sampling	80% detection accuracy	Limited by trace window boundaries

(continued)

Table 1 (continued)

Authors	Datasets	Features used	Extraction method	Approaches	Advantages	Limitations
Jordaney et al. [34]	123,435 B + 5560 M (Derbin) + 9592 B + 9179 M (Marvin)	APIs strings, IP address, permissions, opcodes	Static	SMV, classification algorithm	Discovers zero-day attacks, CE statistical metric, accuracy (96%)	
Huang et al. [35]	2029 malware samples	Byte sequences	Static	K-means algorithm	Better detection (74%) accuracy	Small data set, instruction sequence categorization not optimized
Hu et al. [36]	137,055 samples	Opcode sequences	Static	Unsupervised–prototype-based clustering	detects obfuscated malware (80% accuracy)	Obfuscation reduces the efficiency
O'Kane et al. [37]	260 benign (win xp) + 350 malicious samples	Opcodes	Dynamic	SVM classification	Suitable for encrypted malware, reduces irrelevant features	

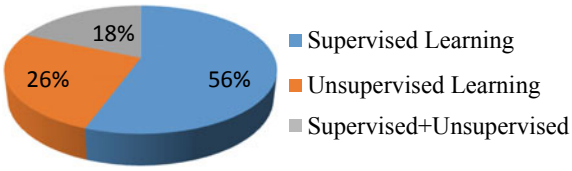

Fig. 1 Machine learning techniques in detection methods

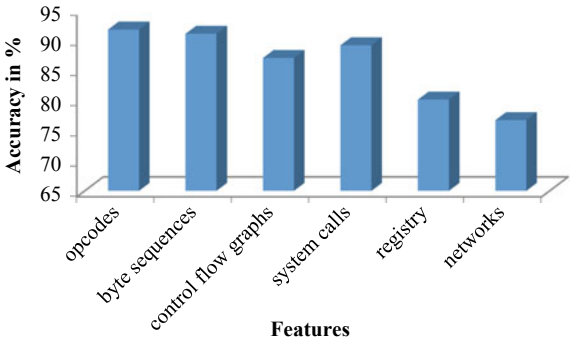

Fig. 2 PE features used for malware detection

18% of the studies have used both supervised and unsupervised learning methods. The statistics presented in Fig. 2 provide insights on the features used for the analysis by various detection methods and their accuracy. Opcodes are the most commonly employed features with the detection accuracy of 91.7%. Byte sequences and API/system call-based detection techniques are also the choices of researchers. Since opcode-based detection methods have better accuracy, our future work will be based on opcode-based malware detection.

This section further discusses that the PE features directly model the behavior of PE samples. To reduce the analysis complexity of the detection systems, only a subset of these features can be considered. Opcodes have been the obvious choice of analysts as they provide low-level details and hence helps in detecting obfuscated malware more efficiently and effectively.

Further, it is evident from the survey of existing studies that most of the solutions are suffering from the issues of datasets availability. The datasets used are not updated since long; some of the repositories are non-existing. The data sizes are small and the sources are not specified. In order to ensure the better availability of datasets, a benchmark dataset needs to be designed. Moreover, malware developers have the advantages over analysts as they can use online public platforms such as Virus total, Metascan and Malwr to test their sample's detection efficiencies by the common anti-viruses. So, the new trend in malware detection is investigating and predicting the future variants using machine learning techniques.

Machine learning methods require complex computations to keep pace with the growing speed of malware developments. The large feature sets increase the time complexity and reduced feature sets decrease the detection accuracy. So a trade-off has been identified between the accuracy and time/space complexity.

4 Conclusion

In light of the increasing complexity of malware, its detection and prevention have become the primary objective of malware researchers. Both the static and dynamic malware analysis techniques are used extensively by the researches to accurately detect the malicious executable. The objective of this paper was to identify the most suitable feature for the detection of malicious executable and devise the challenges in proposing an automated and efficient malware detection system. In the future work, the authors plan to devise an opcode-based malware detection technique on a dataset that can serve as a benchmark for other authors as well.

References

1. Wikipedia 2019 Retrieved on 5 July website: https://en.wikipedia.org/wiki/Malware
2. Website, https://blog.malwarebytes.com/malwarebytes-news/ctntreport/2019/01/2019-state-malware-report-trojans-cryptominers-dominate-threat-landscape/
3. IT Threat Evolution Q1 2019 Statistics, website: https://securelist.com/it-threat-evolution-q1-2019-statistics/90916/. Accessed on 02 July 2019
4. AV-Test IT Security Institute website: https://www.av-test.org/en/statistics/malware/
5. Website, https://securelist.com/mobile-malware-evolution-2018/89689/
6. Y. Ye, T. Li, D. Adjeroh, S.S. Iyengar, A survey on malware detection using data mining techniques. ACM Comput. Surv. (CSUR) **50**(3), 41 (2017)
7. IDA Pro website, https://www.hex-rays.com/products/ida/index.shtml
8. OllyDbg website, http://www.ollydbg.de/
9. OllyDump website, http://www.openrce.org/downloads/details/108/ollydump
10. LordPE website, https://www.aldeid.com/wiki/LordPE
11. A. Moser, C. Kruegel, E. Kirda, Limits of static analysis for malware detection, in *Twenty-Third Annual Computer Security Applications Conference (ACSAC* 2007). IEEE (2007), pp. 421–430
12. M. Egele, T. Scholte, E. Kirda, C. Kruegel, A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. (CSUR) **44**(2), 6 (2012)
13. M. Eskandari, Z. Khorshidpour, S. Hashemi, HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection. J. Comput. Virol. Hack. Techn. **9**(2), 77–93 (2013)
14. Stat counter Website, https://gs.statcounter.com/os-marketshare/desktop/worldwide
15. M.G. Schultz, E. Eskin, F. Zadok, S.J. Stolfo, Data mining methods for detection of new malicious executables, in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P(2001)*. IEEE (2001), pp. 38–49
16. B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane, Graph-based malware detection using dynamic analysis. J. Comput. Virol. **7**(4), 247–258 (2011)
17. M. Eskandari, Z. Khorshidpour, S. Hashemi, Hdm-analyser: a hybrid analysis approach based on data mining techniques for malware detection. J. Comput. Virol. Hack. Tech. **9**(2), 77–93 (2013)
18. P. Khodamoradi, M. Fazlali, F. Mardukhi, M. Nosrati, Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms, in *18th CSI International Symposium on Computer Architecture and Digital Systems (CADS)*. IEEE (2015), pp. 1–6
19. C. LeDoux, A. Lakhotia, Malware and machine learning, in *Intelligent Methods for Cyber Warfare* (Springer, Cham, 2015), pp. 1–42

20. G. Liang, J. Pang, C. Dai, A behavior-based malware variant classification technique. Int. J. Inf. Educ. Technol. **6**(4) (2016)
21. D. Ucci, L. Aniello, R. Baldoni, Survey of machine learning techniques for malware analysis. Comput. Secur. (2018)
22. E. Gandotra, D. Bansal, S. Sofat, Zero-day malware detection, in *Sixth International Symposium on Embedded Computing and System Design* (IEEE, 2016), pp. 171–175
23. A. Damodaran, F. Di Troia, C.A. Visaggio, T.H. Austin, M.A. Stamp, Comparison of static, dynamic, and hybrid analysis for malware detection. J. Comput. Virol. Hack. Tech. **13**(1), 1–12 (2017)
24. Q.K.A. Mirza, I. Awan, M. Younas, CloudIntell: an intelligent malware detection system. Fut. Gen. Comput. Syst. **86**, 1042–1053 (2018)
25. A. Souri, R.A. Hosseini, State-of-the-art survey of malware detection approaches using data mining techniques. HCIS **8**(1), 3 (2018)
26. K. Sethi, S.K. Chaudhary, B.k. Tripathy, P. Bera, A novel malware analysis framework for malware detection and classification using machine learning approach, in *Proceedings of the 19th International Conference on Distributed Computing and Networking* (ACM, 2018), p. 49
27. D. Carlin, P. O'Kane, S. Sezer, Dynamic analysis of malware using run-time opcodes, in *Data analytics and decision support for cybersecurity* (Springer, Cham, 2017), pp. 99–125
28. A.P. Namanya, I.U. Awan, J.P. Disso, M. Younas, Similarity hash-based scoring of portable executable files for efficient malware detection in IoT. Fut. Gen. Comput. Syst. (2019)
29. E. Raff, C. Nicholas, An alternative to NCD for large sequences, Lempel-Ziv Jaccard distance, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2017), pp. 1007–1015
30. P. Vadrevu, R. Perdisci, MAXS: scaling malware execution with sequential multi-hypothesis testing, in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (ACM, 2016), pp. 771–782
31. M. Polino, A. Scorti, F. Maggi, S. Zanero, Jackdaw: towards automatic reverse engineering of large datasets of binaries, in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (Springer, Cham, 2015), pp. 121–143
32. N. Miramirkhani, M.P. Appini, N. Nikiforakis, M. Polychronakis, Spotless sandboxes: evading malware analysis systems using wear-and-tear artifacts, in *IEEE Symposium on Security and Privacy (SP)* (IEEE, 2017), pp. 1009–1024
33. T. Blazytko, M. Contag, C. Aschermann, T. Holz, Syntia: synthesizing the semantics of obfuscated code, in *26th {USENIX} Security Symposium* (2017), pp. 643–659
34. R. Jordaney, K. Sharad, S.K. Dash, Z. Wang, D. Papini, I. Nouretdinov, L. Cavallaro, Transcend: detecting concept drift in malware classification models, in *26th Security Symposium ({USENIX} Security 2017)* (2017), pp. 625–642
35. K. Huang, Y. Ye, Q. Jiang, Ismcs: an intelligent instruction sequence based malware categorization system, in: *Anti-counterfeiting, Security, and Identification in Communication* (IEEE, 2009), pp. 509–512
36. X. Hu, K. G. Shin, S. Bhatkar, K. Griffin, Mutantx-s: scalable malware clustering based on static features, in *USENIX Annual Technical Conference* (2013), pp. 187–198
37. P. O'Kane, S. Sezer, K. McLaughlin, E.G. Im, SVM training phase reduction using dataset feature filtering for malware detection. IEEE Trans. Inf. Forens. Secur. **8**(3), 500–509 (2013)