The Taxonomy of Malware Analysis and Detection Approaches

Subjects: Computer Science, Cybernetics

Contributor: Faitouri A. Aboaoja, Anazida Zainal, Fuad A. Ghaleb, Bander Ali Saleh Al-rimy, Taiseer Abdalla Elfadil Eisa, Asma Abbas Hassan Elnour

The evolution of recent malicious software with the rising use of digital services has increased the probability of corrupting data, stealing information, or other cybercrimes by malware attacks. Therefore, malicious software must be detected before it impacts a large number of computers. While malware analysis is taxonomy and linked to the data types that are used with each analysis approach, malware detection is introduced with a deep taxonomy where each known detection approach is presented in subcategories and the relationship between each introduced detection subcategory and the data types that are utilized is determined.

Keywords: malware detection and classification models ; malware analysis approaches ; malware detection approaches

1. Malware Analysis Approaches and Data Types

Malware analysis and the type of data have a considerable and growing impact on the detection process that determines the classification of the investigated file and therefore influences the overall accuracy of the detection models. Several types of data have been extracted using static, dynamic, and hybrid analysis, such as Byte code, Opcode, API calls, file data, registry data, and so on, to understand and acknowledge the major purpose and function of the files examined and therefore classify them as malware or benign files. **Table 1** contains the characteristics of each reviewed paper in terms of the analysis approach and the extracted data.

Table 1. The relation between analysis approaches and data types.

Ref.	Date	String	PE-Header	Opcode	API Calls	DLL	Machine Activities	Process Data	File Data	Registry Data	Network Data	Derived Data		Accuracy
												Entropy	Compression	
Static Analysis														
[1]	2015			✓										98.60%
[2]	2015		✓											99.97%
[3]	2016	✓												99.00%
[4]	2018	✓										✓		NA
[5]	2017			✓										96.09%
[6]	2017			✓										98.90%
[7]	2017	✓												81.07%
[8]	2018			✓										NA
[9]	2019		✓											99.80%
[10]	2019		✓											87.50%
[11]	2019			✓										97.87%
[12]	2020			✓										NA
[13]	2019			✓										91.43%
[14]	2016	✓											✓	89.44%

Ref.	Date	String	PE-Header	Opcode	API Calls	DLL	Machine Activities	Process Data	File Data	Registry Data	Network Data	Derived Data		Accuracy
												Entropy	Compression	
[15]	2019											✓		NA
[16]	2019		✓									✓		NA
[17]	2020			✓	✓							✓		NA
[18]	2016		✓		✓									98.17%
						Dynamic analysis								
[19]	2015							✓						97.8%
[20]	2016				✓									97.19%
[21]	2016			✓										98.92%
[22]	2016				✓	✓				✓				96.00%
[23]	2016						✓		✓					NA
[24]	2017						✓							98.54%
[25]	2018				✓		✓							NA
[26]	2019						✓							92.00%
[27]	2019				✓						✓			97.22%
[28]	2019				✓									94.89%
[29]	2020				✓									97.28%
[30]	2019				✓				✓					75.01%
[31]	2020				✓									98.43%
[32]	2020	✓			✓				✓	✓	✓			99.54%
[33]	2017				✓									NA
						Hybrid analysis								
[34]	2014	✓			✓									98.71%
[35]	2016			✓					✓					99.99%
[36]	2019		✓		✓									99.70%
[37]	2020	✓			✓									94.05%
[38]	2020		✓		✓									93.92%
[39]	2020	✓	✓	✓	✓									96.30%

1.1. Static Analysis

The static analysis approach has been widely utilized by exploring the source code without running the executable files to extract a unique signature which is used to represent the file under investigation. Several types of static data can be collected via static analysis, including PE-header data [2][9][10][40], derived data such as string-based entropy and compression ratio [4][14][15][16]. Additionally, static analysis tools, such as IDA pro disassembler and Python developed modules, are also used to collect static opcode and API calls [1][5][8][11][17]. Even though static analysis can track all the possible execution paths, it is influenced by packing and encryption techniques.

1.2. Dynamic Analysis

Several researchers performed a dynamic analysis approach to collect various data types to differentiate between malware and benign files by running the executable files in isolated environments, virtual machines (VM) or emulators to monitor the executable file behavior during the run-time and then collect the desired dynamic data [19]. Various kinds of data have been collected utilizing a dynamic analysis approach. Malicious activities can be dynamically represented using both executable file behavior and by retaining memory images during run-time. The executable files' behaviors are identified through collecting the invoked API calls [20][22][25][28][29], machine activities [25][26][41], file-related data [23][30][32], and registry and network data [22][27]. Opcode-based memory image can be taken to represent the malicious activities dynamically [21]. Even though obfuscated malware cannot hide how it behaves when dynamically analyzed, dynamic analysis is unable to satisfy all malicious conditions in order to explore all execution paths.

1.3. Hybrid Analysis

Some previous studies combined data extracted through static and dynamic analysis together to reduce the drawbacks of both analysis approaches and achieve a higher detection rate. Different tools, including Cuckoo sandbox, IDA pro disassembler, and OlleyDbg, are employed to collect dynamic and static data, and then hybrid feature sets are created based on several types of data, such as string, opcode, API calls, and others [35][37][42][43][44]. Even though the hybrid analysis approach benefits from the advantages of both static and dynamic analysis, it also suffers from their disadvantages.

1.4. Malware Analysis and Data Types Discussion

The most used data types with each analysis approach have been shown in **Figure 1**. On the *x*-axis, the data types are depicted as string (St), PE-header (P-h), Opcode (Op), API calls (API), Dynamic link library (DLL), machine activities (MA), process data (PD), file data (FD), registry data (RD), network data (ND), and derived data (DD). Similarly, static, dynamic, and hybrid are mapped on the y-axis as analysis approaches. The horizontal and vertical dotted lines illustrate the relationship between each data type and each analysis approach by showing how frequently each single data type has been used with each particular analysis approach in the literature review.

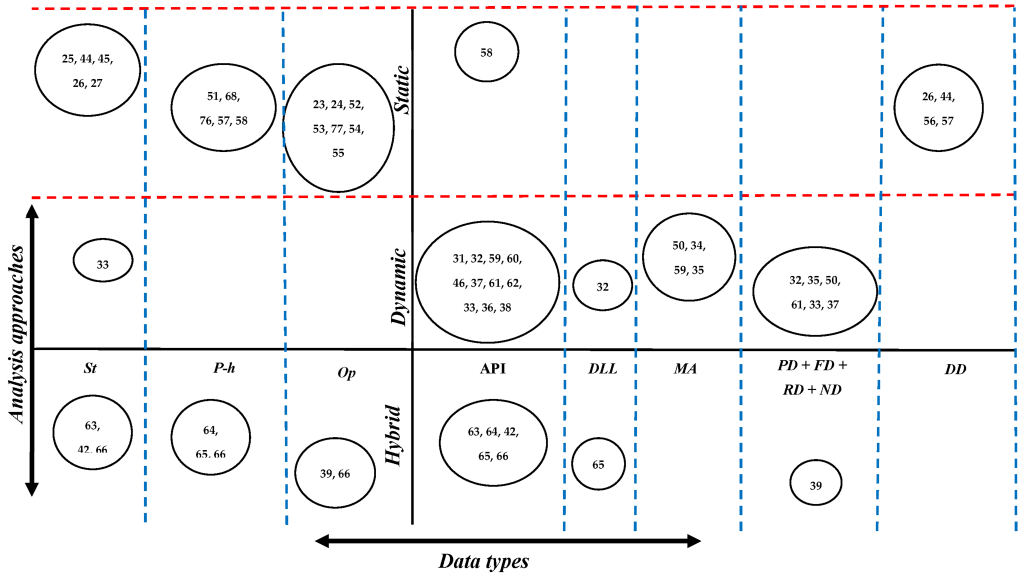

Figure 1. Analysis approaches vs. data types.

By focusing on static analysis, opcode and PE-header represent the first and second most repetitively utilized static data types, respectively. This survey found that using the static data type that is employed in the minimum number of studies, the API calls that are collected statically were not preferable in the literature compared to the usage of byte-code data and the derived data. In contrast to static analysis, the API calls data demonstrates the most significant data type that has been extended using dynamic analysis. While machine activities data is the second trend data type, using data related to registry value, file, and network delivers the average ratio among studies that extracted their required data dynamically. Furthermore, bytecode, PE-header, and Opcode data are rarely extracted using dynamic analysis. In the studies that have utilized both static and dynamic analysis as hybrid analysis, it is clear to note that the same ranges are repeated for the data that are associated with static and dynamic analysis, such as Opcode, bytecode, and PE-header as static data and the API calls as dynamic data.

On the other hand, the usage ratio of some of the dynamic data, such as data related to files, registry, and machine activities, is decreased in the studies that choose hybrid analysis to extract their features.

Even though the static analysis is safe due to there being no need to run the files, malicious software frequently employs encryptors like UPX and ASP Pack Shell to prevent analysis. As a result, unpacking and decompression processes are required before analysis, which is accomplished with disassemblers such as IDA Pro and OlleyDby. Contrary to the static analysis approach, the dynamic analysis approach is more effective because there is no need to unpack the investigated file before the analysis process. Dynamic analysis can support the detection models to be able to detect novel malware behaviors in addition to known malware. Further, the general behavior of the malware is not affected by obfuscation techniques, so it is difficult for obfuscated malware to evade the dynamic analysis approach because the obfuscation techniques change the malware structure but could not change the malware behavior. However, dynamic malware analysis is more sensitive to evasive malware and can detect whether it is being executed in a real or controlled environment.

2. Malware Detection Approaches

The malware detection process is the mechanization that must be implemented to discover and identify the malicious activities of the files under investigation. As a result, several approaches to detecting malware have been improved year after year, with no single approach providing 100% success with all malware types and families in every situation. Therefore, malicious software has been detected based on two main characteristics, which are signatures and behaviors

using three malware detection approaches that are signature-based, behavioral-based, and heuristic-based. Therefore, the following sections discuss the malware detection approaches.

2.1. Signature-Based

Several studies have been undertaken to improve malware detection and classification models by relying on a unique signature that has been previously extracted statically or dynamically and stored to compare it with the collected investigated file signature. Those signatures include but are not limited to a set of API calls, opcodes or byte-code series, and entropy quantity. Static string-based signatures have been generated by [4][45] to detect VBasic malicious software by representing the obtained strings using frequency vectors, while [7] generated static signatures based on n-grams and binary vectors. Additionally, [1] formed malicious static signatures using the statistical values of opcodes. On the other hand, behavioral signatures have been constructed based on the dynamically collected data. The authors of [28][46][47] created behavioral signatures using API calls invoked by malware during their running time. Specific sets of API calls are identified as reflecting malicious activities, and thus the behavioral malicious signatures are constructed utilizing those API calls. Static and behavioral signature-based malware detection models suffer from low detection rates when classifying unknown signatures that may be linked to unknown malware or different variants of known malware.

2.2. Behavioral-Based

After monitoring the executable files in an isolated environment and collecting the exhibited behaviors, features extraction techniques have been developed to extract the sensitive features by which the developed model can classify the known malicious behaviors as well as any behavior that seems to be similar to them with respect to false positive behaviors. The ability to identify novel malware behaviors in addition to the known ones based on collecting behaviors during run-time has made this approach more valuable than the signature-based approach. As a result, the majority of the studies in the literature review focused on using behavioral-based approaches in the form of continuous, sequential, and common behaviors to increase malware detection ratios.

Some studies have been conducted based on the extracted continuous behaviors which, are represented by machine activities. The authors of [8] was interested in the Windows platform and used the Cuckoo sandbox to extract machine activity data (CPU, memory, received, and sent packets). After that, the observations were transformed into vectors, which were used to train and assess classification algorithms. Most of the previous studies were concerned with extracting API calls, system calls, opcodes, and others to form them sequentially (sequential behaviors) or into ordered patterns to understand the malicious functionalities. The sequential or ordered patterns can be API calls, registry data, and network data [17][22][48] or opcode sequences [49]. Moreover, the common behaviors that are performed by malware and benign samples can be used as an indicator to classify the investigated file between malware or benign classes in the binary classification models. In addition, those common behaviors can be observed in each malware family in the case of multi-classification models. The time of the matching process has been reduced from where the developed models classify the test files based on only the common behaviors. Common behaviors graph-based malware detection and classification models have been proposed by [50][51] in their work through observing the most frequent behavior graphs in each malware family. Additionally, ref. [52] presented binary and multi-classification models using (LSTM) long-short term models based on the common API call sequences offered by each malware family.

2.3. Heuristic-Based

A heuristic-based approach has been used in various research by generating generic rules that investigate the extracted data, which are given through dynamic or static analysis to support the proposed model of detecting malicious intent. The generated rules can be developed automatically using machine learning techniques, the YARA tool, and other tools or manually based on the experience and knowledge of expert analysts.

Several studies have been done to develop malware detection models by which decisions have been taken based on the automated behavioral rules that are created using machine learning techniques and the YARA tool [22][25][27]. On the other hand, based on statically extracted string data, ref. [14] was concerned with generating manually general rules to recognize the existence of malicious activities that might be achieved by malware using HTML elements and JavaScript functions. Moreover, (DNS) domain name system-based rules were developed by [49] to build a botnet attack detection model. The proposed model took the final decision based on the manually developed general rules which can detect abnormalities in DNS queries and responses.

2.4. Malware Detection Discussion

According to the literature, the researchers applied string, opcode, and derived features to construct static signatures, while only API calls and opcodes were used to create dynamic signatures. In general, the ability to detect malware utilizing previously derived signatures using both static and dynamic signatures is insufficient to combat malicious software enhancement due to obfuscation techniques popularly used by malware writers to create different malware variants as well as new malware, because each malware variant and new malware seems to have a different fingerprint, which must be obtained before the detection process can begin.

Furthermore, malware detection models based on static signature patterns have been defeated during the tackling of encrypted or compromised malware. Regarding the behavioral-based approach and the preferable data types, machine activities data is the most used feature in the literature to depict malware functionality using continuous behaviors, whereas API calls data is the most frequently utilized feature to build malware detection models using sequential behaviors. Furthermore, the most used data types when authors try to get the common behaviors among malware groups are API calls and opcodes. The behavioral-based approach is a promising solution to overcome the weaknesses of the signature-based approach but relying on behaviors leads the suggested models to misclassify malware that performs functions that are similar to benign functions or mimic legitimate behaviors, and then those models suffer from a high false-positive rate.

Moreover, malware is capable of recognizing the nature of its execution surroundings using evasion techniques and then changing its behaviors to be like benign behaviors or terminating its execution, resulting in representing them through unrepresentative behaviors. In addition, extracting a sufficient feature set is a tough process that has a massive effect on the malware detection and classification models. Furthermore, representing malware behaviors based on the names, sequence, or frequency of extracted characteristics results in malware detection and classification models that are more vulnerable to obfuscation techniques, which are employed to update the names, sequences, and frequencies of the extracted characteristics. Furthermore, several researchers trained their models by employing malicious behaviors that were extracted from the most recent malware to provide the classifiers the capacity to recognize trends in malicious activity. On the other hand, developed malware detection models have become vulnerable to older malicious behaviors.

By focusing on the heuristic-based approach, there is no single type of data that is commonly used with this approach in the literature, but the researchers have used practically almost all the types of data at the same rate, including API calls, network data, registry data, import DLL, and others. However, the creation of general rules that play a significant role in the final decision is required when building a malware detection and classification model based on the heuristic approach. Therefore, generating the investigation rules manually consumes time and effort and needs malware behavior experts with enough experience. Even though the required rules can be generated automatically, the suggested rule-based model is limited to detecting only the malicious activities that are represented in the critical general rules.

References

1. Khodamoradi, P.; Fazlali, M.; Mardukhi, F.; Nosrati, M. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. In Proceedings of the 2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), Tehran, Iran, 7–8 October 2015; pp. 1–6.
2. Zakeri, M.; Daneshgar, F.F.; Abbaspour, M. A static heuristic approach to detecting malware targets. Secur. Commun. Netw. 2015, 8, 30.
3. Kumar, R.; Vaishakh, A.R.E. Detection of Obfuscation in Java Malware. Procedia Comput. Sci. 2015, 78, 521–529.
4. Wael, D.; Sayed, S.G.; AbdelBaki, N. Enhanced Approach to Detect Malicious VBScript Files Based on Data Mining Techniques. Procedia Comput. Sci. 2018, 141, 552–558.
5. Hashemi, H.; Azmoodeh, A.; Hamzeh, A.; Hashemi, S. Graph embedding as a new approach for unknown malware detection. J. Comput. Virol. Hacking Tech. 2017, 13, 153–166.
6. Liu, L.; Wang, B.; Yu, B.; Zhong, Q. Automatic malware classification and new malware detection using machine learning. Front. Inf. Technol. Electron. Eng. 2017, 18, 1336–1347.
7. Fuyong, Z.; Tiezhu, Z. Malware Detection and Classification Based on N-Grams Attribute Similarity. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 793–796.

8. Khalilian, A.; Nourazar, A.; Vahidi-Asl, M.; Haghighi, H. G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families. Expert Syst. Appl. 2018, 112, 15–33.
9. Zelinka, I.; Amer, E. An Ensemble-Based Malware Detection Model Using Minimum Feature Set. Mendel 2019, 25, 1–10.
10. Denzer, T.; Shalaginov, A.; Dyrkolbotn, G.O. Intelligent Windows Malware Type Detection based on Multiple Sources of Dynamic Characteristics. Nis. J. 2019, 12, 20.
11. Lu, R. Malware Detection with LSTM using Opcode Language. arXiv 2019, arXiv:1906.04593.
12. Li, X.; Qiu, K.; Qian, C.; Zhao, G. An Adversarial Machine Learning Method Based on OpCode N-grams Feature in Malware Detection. In Proceedings of the 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), Hong Kong, China, 27–30 July 2020; pp. 380–387.
13. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of ransomware families with machine learning based on N-gram of opcodes. Futur. Gener. Comput. Syst. 2019, 90, 211–221.
14. PSeshagiri, P.; Vazhayil, A.; Sriram, P. AMA: Static Code Analysis of Web Page for the Detection of Malicious Scripts. Procedia Comput. Sci. 2016, 93, 768–773.
15. Ling, Y.T.; Sani, N.F.M.; Abdullah, M.T.; Hamid, N.A.W.A. Nonnegative matrix factorization and metamorphic malware detection. J. Comput. Virol. Hacking Tech. 2019, 15, 195–208.
16. Kumar, A.; Kuppusamy, K.; Aghila, G. A learning model to detect maliciousness of portable executable using integrated feature set. J. King Saud Univ.-Comput. Inf. Sci. 2019, 31, 252–265.
17. Euh, S.; Lee, H.; Kim, D.; Hwang, D. Comparative Analysis of Low-Dimensional Features and Tree-Based Ensembles for Malware Detection Systems. IEEE Access 2020, 8, 76796–76808.
18. Belaoued, M.; Mazouzi, S. A chi-square-based decision for real-time malware detection using PE-file features. J. Inf. Process. Syst. 2016, 12, 644–660.
19. Choudhary, S.; Vidyarthi, M.D. A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining. Procedia Comput. Sci. 2015, 54, 265–270.
20. Galal, H.S.; Mahdy, Y.B.; Atiea, M.A. Behavior-based features model for malware detection. J. Comput. Virol. Hacking Tech. 2016, 12, 59–67.
21. Banin, S.; Shalaginov, A.; Franke, K. Memory access patterns for malware detection. Nor. Inf. 2016, 96, 107.
22. Mosli, R.; Li, R.; Yuan, B.; Pan, Y. Automated malware detection using artifacts in forensic memory images. In Proceedings of the 2016 IEEE Symposium on Technologies for Homeland Security (HST), Waltham, MA, USA, 10–12 May 2016; pp. 1–6.
23. Norouzi, M.; Souri, A.; Zamini, M.S. A Data Mining Classification Approach for Behavioral Malware Detection. J. Comput. Netw. Commun. 2016, 2016, 1–9.
24. Burnap, P.; French, R.; Turner, F.; Jones, K. Malware classification using self organising feature maps and machine activity data. Comput. Secur. 2018, 73, 399–410.
25. Jerlin, M.A.; Marimuthu, K. A New Malware Detection System Using Machine Learning Techniques for API Call Sequences. J. Appl. Secur. Res. 2018, 13, 45–62.
26. Fasano, F.; Martinelli, F.; Mercaldo, F.; Santone, A. Energy Consumption Metrics for Mobile Device Dynamic Malware Detection. Procedia Comput. Sci. 2019, 159, 1045–1052.
27. Belaoued, M.; Boukellal, A.; Koalal, M.A.; Derhab, A.; Mazouzi, S.; Khan, F.A. Combined dynamic multi-feature and rule-based behavior for accurate malware detection. Int. J. Distrib. Sens. Netw. 2019, 15, 155014771988990.
28. Kim, H.; Kim, J.; Kim, Y.; Kim, I.; Kim, K.J.; Kim, H. Improvement of malware detection and classification using API call sequence alignment and visualization. Cluster Comput. 2019, 22, 921–929.
29. Hwang, J.; Kim, J.; Lee, S.; Kim, K. Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques. Wirel. Pers. Commun. 2020, 112, 2597–2609.
30. Arabo, A.; Dijoux, R.; Poulain, T.; Chevalier, G. Detecting Ransomware Using Process Behavior Analysis. Procedia Comput. Sci. 2020, 168, 289–296.
31. Ali, M.; Shiaeles, S.; Bendiab, G.; Ghita, B. MALGRA: Machine Learning and N-Gram Malware Feature Extraction and Detection System. Electronics 2020, 9, 1777.
32. Singh, J.; Singh, J. Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. Inf. Softw. Technol. 2020, 121, 106273.

33. JVidal, M.; Orozco, A.L.S.; Villalba, L.J.G. Malware Detection in Mobile Devices by Analyzing Sequences of System Calls. Int. J. Comput. Electr. Autom. Control. Inf. Eng. 2017, 11, 588–592.
34. Shijo, P.; Salim, A. Integrated Static and Dynamic Analysis for Malware Detection. Procedia Comput. Sci. 2015, 46, 804–811.
35. Fraley, J.B.; Figueroa, M. Polymorphic malware detection using topological feature extraction with data mining. In Proceedings of the SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–7.
36. Darshan, S.L.S.; Jaidhar, C.D. Windows malware detection system based on LSVC recommended hybrid features. J. Comput. Virol. Hacking Tech. 2019, 15, 127–146.
37. Huang, X.; Ma, L.; Yang, W.; Zhong, Y. A Method for Windows Malware Detection Based on Deep Learning. J. Signal Process. Syst. 2021, 93, 265–273.
38. Darshan, S.L.S.; Jaidhar, C.D. An empirical study to estimate the stability of random forest classifier on the hybrid features recommended by filter based feature selection technique. Int. J. Mach. Learn. Cybern. 2020, 11, 339–358.
39. Kang, J.; Won, Y. A study on variant malware detection techniques using static and dynamic features. J. Inf. Process. Syst. 2020, 16, 882–895.
40. Naz, S.; Singh, D.K. Review of Machine Learning Methods for Windows Malware Detection. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019; pp. 1–6.
41. Ahmed, Y.A.; Koçer, B.; Huda, S.; Al-Rimy, B.A.S.; Hassan, M.M. A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. J. Netw. Comput. Appl. 2020, 167, 102753.
42. Ndibanje, B.; Kim, K.H.; Kang, Y.J.; Kim, H.H.; Kim, T.Y.; Lee, H.J. Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction. Appl. Sci. 2019, 9, 239.
43. Zhong, W.; Gu, F. A multi-level deep learning system for malware detection. Expert Syst. Appl. 2019, 133, 151–162.
44. Damodaran, A.; Di Troia, F.; Visaggio, C.A.; Austin, T.; Stamp, M. A comparison of static, dynamic, and hybrid analysis for malware detection. J. Comput. Virol. Hacking Tech. 2017, 13, 1–12.
45. Wael, D.; Shosha, A.; Sayed, S.G. Malicious VBScript detection algorithm based on data-mining techniques. In Proceedings of the 2017 International Conference on Advanced Control. Circuits Systems (ACCS) Systems & 2017 International Conference on New Paradigms in Electronics & Information Technology (PEIT), Alexandria, Egypt, 5–8 November 2017; pp. 112–116.
46. Ki, Y.; Kim, E.; Kim, H.K. A Novel Approach to Detect Malware Based on API Call Sequence Analysis. Int. J. Distrib. Sens. Netw. 2015, 11, 659101.
47. Sihwail, R.; Omar, K.; Ariffin, K.A.Z.; Al Afghani, S. Malware Detection Approach Based on Artifacts in Memory Image and Dynamic Analysis. Appl. Sci. 2019, 9, 3680.
48. Saxena, S.; Mancoridis, S. Malware Detection using Behavioral Whitelisting of Computer Systems. In Proceedings of the 2019 IEEE International Symposium on Technologies for Homeland Security (HST), Greater Boston, MA, USA, 5–6 November 2019; pp. 1–6.
49. Alieyan, K.; Almomani, A.; Anbar, M.; Alauthman, M.; Abdullah, R.; Gupta, B.B. DNS rule-based schema to botnet detection. Enterp. Inf. Syst. 2021, 15, 545–564.
50. Kakisim, A.G.; Nar, M.; Sogukpinar, I. Metamorphic malware identification using engine-specific patterns based on co-opcode graphs. Comput. Stand. Interfaces 2019, 71, 103443.
51. Du, D.; Sun, Y.; Ma, Y.; Xiao, F. A Novel Approach to Detect Malware Variants Based on Classified Behaviors. IEEE Access 2019, 7, 81770–81782.
52. Catak, F.O.; Yazı, A.F.; Elezaj, O.; Ahmed, J. Deep learning based Sequential model for malware analysis using Windows exe API Calls. PeerJ Comput. Sci. 2020, 6, e285.