A Survey on malware detection techniques

Tibra Alsmadi
Department of Computer Information Systems
Jordan University of Science and Technology
Irbid, Jordan
tralsmadi18@cit.just.edu.jo

Nour Alqudah
Department of Computer Information Systems
Jordan University of Science and Technology
Irbid, Jordan
nsalqudah18@cit.just.edu.jo

2021 International Conference on Information Technology (ICIT) | 978-1-6654-2870-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIT52682.2021.9491765

Abstract __ Malware programs have become a serious threat as it was developed to damage computer systems, spread over the network or Internet connections. Researchers are making great efforts to produce anti-malware systems with practical ways to detect malware protection and malware detection of computer systems.
Two basic approaches were proposed: based on the signature and the heuristics rule detected, we can detect known malware accurately. However, signature-based detection techniques are regarded as ineffective due to cannot identify unknown malware, code obfuscation, packing, polymorphic and metamorphic malware. This survey paper highlights current detection and analysis methodologies used in malicious codes.
Keywords—Malware detection, Security, Survey Malicious programs, Static analysis, Dynamic analysis, Hybrid Analysis.

I. Introduction

Malware is software that aims primarily to infiltrate a computer system without the knowledge and consent of the owner of the systems. Malware can be hostile, destructive, and intrusive, and perhaps its construction is benign, but its presence leads to inconvenience.

Viruses, worms, Trojans horses, Rootkits, spyware are all different forms and types of malicious programs. Then a categorized based on the method of spread and the purpose for which it was built. Its various forms can allow others to spy on the device, launch annoying pop-up ads, send emails from an account without the knowledge of them, and in the worst case, destroy the system almost from the inside [14].

In many studies, it has been to take an alarm keeping in mind the financial implications for institutions because of these threats, for example, the notion of ransomware. The malware hides all folders and encrypts files on PC's C: drive. In 1989, a Trojan called PC Cyborg was distributed, and the script delivered a ransom demanding that $189 be allocated to PC Cyborg Corporation. The affected computer will not function until the ransom is paid, and the malware actions are reversed. Since that time, many improvements have been made to this type of threat, especially in file encryption [12], where ransomware is a type of malware that threatens to publish the victim's data or perpetually block access to it unless a ransom is paid. It typically operates by locking the victim's desktop to render the system inaccessible to the user or by encrypting, overwriting, or deleting the users.

Different types of malware can harm either slowing down the computer or replacing the web page in the browser and allowing hackers to control the system.

Sometimes malware that shows web pages or ads designed to make a profit for the malware creator, for example, Trojans or Rootkits, locks from the computer and passes control of the system to the hacker to control information confidential or sensitive such as bank details, credit card number, personal data and even use the computer to hack or injure others If the computer is controlled by malware, it will not improve by itself, should remove the malware necessary to get the computer back on track and improve it. Therefore, there is a need to use an approach to detect Ransomware and polymorphic programs and other malicious programs to eliminate them.

The researchers note that malware evolves significantly in structure, but there is one fixed property at each stage. All malware is designed to take advantage of an unpleasant way that takes some action to achieve its goals. Assuming malware exists on a Windows device, it will need to use some of the services provided by the Windows operating system (Windows API calls) to generate malicious behavior, so malware detection, classification, and behavior analysis can be performed well based on dynamic analysis to determine their behavior through Windows API Calls [9].
This paper is a survey study of the basics of malware. Section two summarizes related work. Section three discussion of malware analysis techniques. Section four malware detection methods. Section five explains WINDOWS API Calls. In section six, an extensive review of malware detection systems is presented. Section seven concludes the paper with summary comments of content.

II. Related Work

Due to the huge risk of Malware spread on the information and critical threat to the user's computer system, posted taking huge interest from the researchers to benefit from the posted information.
Malware detection developers have taken the lead with the advancement of the solution; because the attackers go on illegal ways to joke or collect profits and threaten others in all fields:
The research work in [1] applied evaluated two approaches, which are the Longest Common Subsequence and the Longest Common Substring algorithms in the context of malware detection rate and false alarm to mitigation from the computational complicate and therefore, cannot scale to large API call sequences, which was result LCSS was improving better than LCS in terms of detection rates and false alarms because it using traceback fashion to construct the subsequence, Another research in[2] analysis

identifying and classifying PE files into malicious and benign to detect malware based on mining behavioral aspects of API calls, through the extraction and interpretation of API calls a feature selection algorithm has been proposed to identify unique APIs, Similarly, research work in [3] framework based on mining API calls which framework includes PE analyzer, feature generator, selector and classifier, after exporting Windows API calls from PEs by PE analyzer, various features are generated and selected form API call sets, which applied classification methods benign or malicious.

One more research work in [4] a method to analyze the API call sequence Instead of relying on the API for each class, that extract API call sequence patterns from malware in different classes focus on common malware functions, which has developed a signature database and APIMDS empirically, with more accurate results and a very low error rate. Also, detection systems rely on signature fixed information for malware, such as file size, process, and effects. Therefore, signatures must always be updated so that the process does not fail; additional research work in [5] propose a clustering framework to detect similar malware behaviors based on the comparison of API call sequences. Which searching the performances of three sequence comparison functions, LCS, namely, OM, and LCP, as a basis for clustering, the LCP function provides the best results in terms of clustering quality and time complexity in API sequence-based malware analysis that can serve as a quick and reliable descriptor for overall similar behaviors in malware.

A study in[6]that API calls are not able to represent all behavior of the applications, So two feature categories: API names and a combination of API names and their input a small set of APIs were extracted in run-time to achieved their impacts in identifying malware and benign applications, also In this study [7] focus on the usage of frequent messages in API call sequences, using the frequent messages of API call sequences can achieve high accuracy for malicious program clustering while reducing the computation time, also proves the importance of frequent item sets in API call sequences for identifying the behavior of malware, An additional research archived the high precision in[8] malware detection which worked a behavioral-based classification of unknown malware by leveraging RNN-LSTM technique based on API call sequences, where used N-gram for feature extraction and TF-IDF for feature selection of the sequences.

The research of Morgan C. Wang et al. [20] in using the static analysis method, using data extraction techniques to extract critical behavior from all programs automatically, used the sequences of instructions extracted from the clean, malware removal an essential classification feature. They formulated the problem as a binary classification problem, the construction of logistic regression, neural networks, and decision tree models.

In another paper [21], a study to use the new feature sets that solve the problem of mobile / Android malware detection, and to analyze the detection of malware; by performing machine learning compilations such as Random Forest classifier, it was more suitable on the side of TPR / FPR. SVM showed similar performance to Random Forest: TP (True positive) for regular type data and FP (False positive) for malware type data.

Also, another search [22] proposed a method for early detection of malware, which runs on the Android system, through parallel machine learning compilations, which use various algorithms of different nature by nature to increase accuracy, where was used the features of fixed applications and was compared results with trained models such as API-linked calls, Linked to commands that were combined with a variety of schemes, to give a composite model that produces a "suspicious" or "benign" judgment to categorize a specific new application.

Some of the research explains the importance of reading the instructions for the permission system in Android apps helps increase awareness of the risks. However, we cannot guarantee that all users read or understand them, so Kabakus Abdullah Talha et al. [23] introduced a malware detection system that runs the APK Auditor, which consists of three components: a signature database to store information generated by application analysis, an Android client that end-users use to grant application analysis requests, and a central server responsible for communicating with both Database, signature and smartphone client, manage the entire analysis process and provides two methods for Android app analysis: search through Play Store, and download the application (.apk file) to analyze applications from other Android sources. Also, the problem of a lack of awareness of the signature-based approach to identifying new malware has been solved by the suggestion of Hardy Jiang et al. [24]using a deep learning method, included in the DLGraph malware detection graph. It uses two auto-encoded devices (SDAs) stacked to learn representation, considering graphs to invoke computer software functions, and Windows APIs, where one of the SDA can learn an underlying representation of graphs and software calling functionality. Other SDA can see a latent representation of Windows API call programs.

III. Malware Detection TechniqueS

Malware analysis is a necessary step to develop an effective malware detection technique. It is the process of analyzing the target and how malware performs its work so that it is easier for developers to detect these programs and the work needed to avoid them, divided into three categories based on time and technology to do the analysis [11, 25].

A. Static analysis detection technique:

For static analysis, correct errors automatically from the source code between coding and unit testing before executing and running the program. The application is divided using disassemble tool, decompile tool, debugger, source code analyzer tools Ollydbg to understand malware and its structure. The static analysis addresses weaknesses in source code to avoid security vulnerabilities, and source code can be manually reviewed, but machine tools are practical.

a) Advantages of Static analysis[28]:

- Is fast and safe.
- It gathers the structure of the code of the program under inspection.

- If static analysis can calculate the malware behavior in the application, this information can then be used.

b) Disadvantages of Static analysis[28]:

- The static analysis does not take a stand for analyzing the unknown malware. Also, the source code of many programs is not readily available.
- For doing static analysis, researchers must have a good knowledge of assembly language and have a deep understanding of the operating system's functioning.

B. Dynamic analysis detection technique:

The process of analyzing the application's behavior during its execution, dynamic code analysis as part of code debugging allows to test the program in any scenario, so do not need to create fake entries or situations with the possibility of producing unexpected effects errors. However, it can detect unexpected problems and required functions during the implementation process that do not appear at the design stage. It is impossible to identify all possible scenarios and is a standard procedure because it reduces the cost and time of testing while facilitating maintenance [27].

Dynamic code analysis is also applied during the testing phase. If there were some errors due to specific scenarios, they are the only option to resolve the problem.

a) Advantages of dynamic analysis[28]:

- It able to detect dependencies that are not possible to detect in the static analysis, ex: dynamic dependencies using reflection, dependency injection, polymorphism.
- Can collect temporal information.
  Deals with real input data, where during the static analysis, it is difficult to know what files will be passed as input, what WEB requests will come, what user will click, Etc.

b) Disadvantages of dynamic analysis[28]:

- May negatively impact the performance of the application.
- Cannot guarantee the full coverage of the source code, as it is running are based on user interaction or automatic tests.

C. Hybrid Analysis detection technique:

This technique is the combination of both static analysis and dynamic analysis; the procedure follows that it first checks for any malware signature if present in the code under inspection, and then it monitors the behavior of the code [15].

IV. Malware Detection MethodS

Malware detection techniques are used to identify, detect the malware, and prevent the computer system from being infected, protecting it from information loss. They can be categorized into signature-based detection, behavior-based detection, and sandbox.

A. Signature-based Detection :

Signal-based detection uses a signature or sequence of bits to identify the malware and its type, programs with a unique code that runs when a file accesses the computer, the malware scanner collects the code into the cloud-based database.

The database contains a wide range of virus codes. Suppose the file icon is matched with an icon in the list. In that case, the database determines that the file is malicious and is rejected by the anti-malware program from the computer and deleted after the antivirus program dismantles the infected file and recognizes its pattern and sequence belongs to determine its type, and in case of mismatch and discover that it New malicious, whose code is added to the list, this is a technique can be static, dynamic or mixed[13]. Also, there are three types of signature analysis for worm detection: network payload signatures; the second type of signature detection is file signatures; the third type is based on log file analysis [26].

a) Advantages *of the signature-based detection[29]:*

- Broadly accessible.
- Easy to run.
- Fast identification.
- Finding comprehensive malware information.

b) The weaknesses of the signature-based detection [29]:

- Failing to detect polymorphic malware.
- Fast identification replicating information in the vast database.

B. Heuristic -based Detection:

It is an approach to detect and distinguish between normal and abnormal behavior of the system to identify known and unknown malware attacks and find a suitable solution. Weight-based rules or systems to determine how much risk a program's function may pose. If these rules exceed the predetermined limit, preventive action is taken based on system settings; for example, the file is quarantined and deleted, or a notification is sent to the server administrator to alert [13].

C. SandBox Detection:

Sandbox is a protected cell within a computer created by anti-malware programs to contain any untested or untreated programs or code. It prevents malware infection because the file works without infecting to harm the host device.

Inside the sandbox, the file is observed and further analyzed to determine whether it is harmful or safe, where tests unverified programs that may contain a virus or other malicious code; if the file is legitimate, it will be released, but if it is harmful, it will be rejected.

V. WINDOWS API CALLS

The Windows API is a broad set of functions that control the way malware interacts with Microsoft libraries, which use specific names, terms, and agreements that must become familiar with before moving to specific functions. Also, Windows generally uses Hungarian APIs [10, 30].

A. API Call Sequence:

Each single API call is an accurate action performed by malicious software or benign activity when the system runs, for example: creating, reading, writing, and deleting files or registry keys.

Also, specifying API strings as features because the order of their calls shows how malware behaves or benign with an operating system, were malicious and benign programs will have their API call patterns or special order of calls [10, 30].

B. Operating system API calls:

The use of operating system API calls is a promising task in detecting PE-type malware in the Windows operating system.

This task is officially defined as running malware in an isolated sandbox environment, recording the API calls made with the Windows operating system, and sequentially analyzing them [9].

VI. SURVEY OF EXISTING WORK

In the current scenario, we have malware detection and analysis as the basis for this paperwork. So, we surveyed malware detection and analysis methods, which are summarized in the table below:

Table.1. Malware detection and analysis methods

S. N	Analysis Algorithm	Analysis /detection techniques	Challenges	Computation cost	Target	Accuracy
1. [31]	Image comparison of malware families and useful data sets is used to visualize the difference in malware behavior patterns. Deep learning techniques are used to facilitate self-learning in order to achieve high accuracy in the proposed classifications.	Image-based techniques for detecting malware.	NA	Lower computational cost.	It developed a hybrid deep learning model to detect and classify malware by using image-based machine learning that is cost-effective and scalable.	99%
2. [32]	The first component is a lightweight host proxy to identify all new files and transfer them to the network service, which receives all files sent by the host proxy, then it analyzes all unwanted and harmful content in the file and instructs to host whether the access file is safe or not. The last step is the archive and forensic service. This component maintains all file records analyzed and creates a query and work alert interface.	Using the cloud computing based on host agent.	NA	Low cost.	The cloud environment more powerful, and protection was being made by overcoming virus problems.	NA
3. [33]	The AMD framework contains a VM operator that detects the known malware by using signature and anomaly detection by creating a fixed feature using the binary bat algorithm and then moving it to Hypervisor to implement anomaly detection using a random forest classifier.	AMD framework.	Virtualization leads to security concerns for cloud computing because of the distributed and dynamic nature of cloud computing.	Affordable computational cost.	VMs by detecting malware was secured executables in high-risk VMs of cloud computing.	>96%
4. [34]	Play a file system scanning role based on Yara signatures from Nessus 6.8, looking for specific file hashes for disk files. Scanning multiple devices is a good alternative without having to install Yara, using ClamAV, that determining malware is located on the device and disposed of, in addition to using YARA rules with ClamAV 0.99 is easy, as YARA rule files are placed in a virus database site ClamAV and run anti-virus.	Yara is an open-source and multi-platform tool.	Nessus is not a free and open-source software tool, it contains a version that has no price.	Decrease cost.	Providing an in-depth analysis of Mirai botnet security, it is malware that affects banking systems' availability and establishes new evidence for a DDoS attack that works with Internet devices. Also, the mitigation of similar malware.	NA

S. N	Analysis Algorithm	Analysis detection techniques	Challenges	Computation cost	Target	Accuracy
5. [35]	Extraction Windows API calls are then parsed using the constructed classification model, which consists of an AutoEncoder tool stacked with Boltzmann's Multilayer Constrained Devices (RBMs), and a layer of relational memory to detect newly unknown malware—after that, labeling each unknown file either benign or malicious.	DeepAM: Heterogeneous deep learning framework.	How sparsity restrictions are imposed on AutoEncoder and how stacked RBMs can be improved for better detection performance.	High cost.	Malware detection based on Windows API calls extracted from the PE files.	98%
6. [36]	First, the XML file for executing malware behavior history is converted to an unbundled array using our proposed application for malware detection's behavioral analysis. This matrix translates into a WEKA input data set to demonstrate the performance efficiency. The proposed methods were applied to a real case study data set using the WEKA tool.	Dynamic analysis method using some classification algorithms such as NaiveBayse, BayseNet, IB1, J48, and regression algorithms.	High complexity. Analyze a real behavioral antivirus platform.	NA	Suggesting a new data mining approach was based on classification methodologies to detect malware behavior.	>86 %
7. [37]	Compute opcodes' suitability by following the sequence of the steps: Dismantle executables based on New Basic Assembler7 to obtain assembly files. Then, create an opcode using the generated assembly files ,that each file contains a list with the activation code and the times when each operating code appears within the malicious and malicious software dataset. The last step calculates the importance of the opcode based on the frequency that appears in each dataset.	Based on the frequency of the appearance of opcode sequences.	Scalability of malware databases. Detection and facing packed executables.	NA	Training machine learning algorithms to detect unknown malware variants.	>90%
8. [38]	EHNFC uses a modified ECM to create evolving fuzzy rules in online mode for classifying Android application (.apk) file as malware or benign program.	Neuro-fuzzy classifier (EHNFC) for Android malware classification using permission-based features.	Not considering the dynamic analysis of Android apps. Run-time overhead.	NA	Android malware detection system was improved. Also, evolving EHNFC structure by learning new malware detection fuzzy rules.	90%
9. [39]	Perform dynamic analysis of the malware data set using API call technology to track API calls by the malware model. Then we are processing more API calls that follow them into groups of API calls that are logically dependent.	Behavior-based detection.	Not suitable for samples of external events. Presence analysis. The extraction technique is not fully automatic.	NA	Description of the malware actions was shown by malware during runtime. Also, utilizing promising API calls to detect malware.	NA
10. [40]	The model measures the similarity between two graphs based on structural similarities and by using the metric similarity to calculate the Euclidean distance between two GR-type graphics.	A graph-based model using System-calls groups.	High time consumption.	NA	Detect malware by taking advantage of relationships between groups of system calls.	94.7%
11. [41]	The API sequences used for malicious codes extracted them through dynamic analysis and used them as feature data for machine learning to effectively distinguish between malicious codes and benign files. Extracted feature data was used previously by performing static and dynamic analysis and applying it to the software using the malware dataset provided by KISIS. The feature selection process was applied to the extracted feature data to reduce the number of features so that the learning model could run effectively.	Machine learning model using XG Boost, RF, and DNN classifiers	Study features that can enormously improve the implementation with light features that take the lowest time to extract.	NA	Feature data was extracted from 7,000 malware and 3,000 benign files using static and dynamic malware analysis tools	96.3%
12. [42]	This model is based on analyzing and categorizing the API call sequence. The dataset is collected from Kaggle, consisting of 42,797 malicious API call sequences and 1,079 harmless API sequences. The KNN algorithm is applied to the dataset to create a model that detects malware.	K-nearest Neighbor (KNN)	NA	NA	The proposed model is significantly essential for detecting real-time encroachment on computer systems.	98.17%

VII. Conclusion

Malware is a global pestilence. Research indicates that the effect of malware is becoming worse. Malware detection devices are the leading equipment to defend against anti-malware. The accuracy of such a detect device is specified by the method it uses. So, it is essential that malware detection techniques were studied and realize their strength and limitations. This paper has highlighted a detailed review of the technical status of malware, presenting malware detection techniques. It summarizes several malware detection systems. Although malware detection systems are overgrowing due to rapid development, this study can be considered as a reference for developers in this field.

REFERENCES

[1] F. Mira and W. Huang, "Performance Evaluation of String Based Malware Detection Methods," 2018 24th Int. Conf. Autom. Comput., no. September, pp. 1–6, 2019.

[2] D. Uppal, R. Sinha, V. Mehra, and V. Jain, "Exploring behavioral aspects of API calls for Malware identification and categorization," Proc. - 2014 6th Int. Conf. Comput. Intell. Commun. Networks, CICN 2014, pp. 824–828, 2014.

[3] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamze, "Malware detection based on mining API calls," Proc. ACM Symp. Appl. Comput., pp. 1020–1025, 2010.

[4] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," Int. J. Distrib. Sens. Networks, vol. 2015, 2015.

[5] Al Shamsi, F., W.L. Woon, and Z. Aung, "Discovering Similarities in Malware Behaviors by Clustering of API Call Sequences. in International Conference on Neural Information Processing," 2018. Springer.

[6] Z. Salehi, M. Ghiasi, and A. Sami, "A miner for malware detection based on API function calls and their arguments," AISP 2012 - 16th CSI Int. Symp. Artif. Intell. Signal Process., no. Aisp, pp. 563–568, 2012.

[7] Y. Qiao, Y. Yang, L. Ji, and J. He, "Analyzing malware by abstracting the frequent itemsets in API call sequences," Proc. - 12th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2013, pp. 265–270, 2013.

[8] Mathew, J. and M.A. Kumara, "*API Call Based Malware Detection Approach Using Recurrent Neural Network—LSTM.* in *International Conference on Intelligent Systems Design and Applications*," 2018. Springer.

[9] Catak, F.O. and A.F. Yazı, "*A Benchmark API Call Dataset for Windows PE Malware Classification*," arXiv preprint arXiv:1905.01999, 2019.

[10] M. Sikorski, *Praise for Practical Malware Analysis*. 2012.

[11] R. Brewer, "Ransomware attacks: detection, prevention and cure," Netw. Secur., vol. 2016, no. 9, pp. 5–9, 2016.

[12] D. Uppal, V. Mehra, and V. Verma, "Basic survey on Malware Analysis, Tools and Techniques," Int. J. Comput. Sci. Appl., vol. 4, no. 1, pp. 103–112, 2014.

[13] R. Tahir, "A Study on Malware and Malware Detection Techniques", Int. J. Educ. Manag. Eng., vol. 8, no. 2, pp. 20–30, 2018.

[14] S. R. Singh, "International Journal of Innovative Research in Computer and Communication Engineering The Internet of Things in Education (IoTE): An Overview," pp. 5042–5054, 2017.

[15] M. Jain and P. Bajaj, "Techniques in Detection and Analyzing Malware Executables: A Review," Int. J. Comput. Sci. Mob. Comput., vol. 35, no. 5, pp. 930–935, 2014.

[16] J. Landage and M. Wankhade, "Malware and Malware Detection Techniques: A Survey," Int. J. Eng. Res. Technol., vol. 2, no. 12, pp. 61–68, 2013.

[17] H. S. Ham and M. J. Choi, "Analysis of Android malware detection performance using machine learning classifiers," Int. Conf. ICT Converg., pp. 490–495, 2013.

[18] M. Siddiqui, M. C. Wang, and J. Lee, "Data mining methods for malware detection using instruction sequences," Proc. IASTED Int. Conf. Artif. Intell. Appl. AIA 2008, no. December, pp. 358–363, 2008.

[19] K. A. Talha, D. I. Alper, and C. Aydin, "APK Auditor: Permission-based Android malware detection system," Digit. Investig., vol. 13, pp. 1–14, 2015.

[20] R. Brewer, "Ransomware attacks: detection, prevention and cure," Netw. Secur., vol. 2016, no. 9, pp. 5–9, 2016.

[21] H. S. Ham and M. J. Choi, "Analysis of Android malware detection performance using machine learning classifiers," Int. Conf. ICT Converg., pp. 490–495, 2013.

[22] M. Jain and P. Bajaj, "Techniques in Detection and Analyzing Malware Executables: A Review," Int. J. Comput. Sci. Mob. Comput., vol. 35, no. 5, pp. 930–935, 2014.

[23] H. Jiang, T. Turki, and J. T. L. Wang, "DLGraph: Malware Detection Using Deep Learning and Graph Embedding," Proc. - 17th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2018, pp. 1029–1033, 2019.

[24] J. Landage and M. Wankhade, "Malware and Malware Detection Techniques: A Survey," Int. J. Eng. Res. Technol., vol. 2, no. 12, pp. 61–68, 2013.

[25] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, 2017.

[26] R. Rizwan, G. C. Hazarika, and G. Chetia, "Malware threats and mitigation strategies: A survey," *J. Theor. Appl. Inf. Technol.*, vol. 29, no. 2, pp. 69–73, 2011.

[27] J. Landage and M. Wankhade, "Malware and Malware Detection Techniques: A Survey," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 61–68, 2013.

[28] R. Tahir, "A Study on Malware and Malware Detection Techniques," *Int. J. Educ. Manag. Eng.*, vol. 8, no. 2, pp. 20–30, 2018.

[29] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Comput. Inf. Sci.*, vol. 8, no. 1, 2018.

[30] T. Notes, "a Vamar C Lient for W Indows on," no. May, pp. 1–13, 2011.

[31] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *J. Inf. Secur. Appl.*, vol. 47, pp. 377–389, 2019.

[32] A. Bedi, N. Pandey, and S. K. Khatri, "Analysis of Detection and Prevention of Malware in Cloud Computing Environment," *Proc. - 2019 Amity Int. Conf. Artif. Intell. AICAI 2019*, pp. 918–921, 2019.

[33] R. Patil, H. Dudeja, and C. Modi, "Designing in-VM-assisted lightweight agent-based malware detection framework for securing virtual machines in cloud computing," *Int. J. Inf. Secur.*, 2019.

[34] L. E. S. Jaramillo, "Malware Detection and Mitigation Techniques: Lessons Learned from Mirai DDOS Attack," *J. Inf. Syst. Eng. Manag.*, vol. 3, no. 3, 2018.

[35] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: a heterogeneous deep learning framework for intelligent malware detection," *Knowl. Inf. Syst.*, vol. 54, no. 2, pp. 265–285, 2018.

[36] M. Norouzi, A. Souri, and M. Samad Zamini, "A Data Mining Classification Approach for Behavioral Malware Detection," *J. Comput. Networks Commun.*, vol. 2016, pp. 20–22, 2016.

[37] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci. (Ny).*, vol. 231, pp. 64–82, 2013.

[38] A. Altaher, "An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 4147–4157, 2017.

[39] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 12, no. 2, pp. 59–67, 2016.

[40] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malware detection and classification using system-call groups," *J. Comput. Virol. Hacking Tech.*, vol. 13, no. 1, pp. 29–46, 2017.

[41] J. Kang and Y. Won, "A Study on Variant Malware Detection Techniques Using Static and Dynamic Features," vol. 16, no. 4, pp. 882–895, 2020.

[42] T. A. Assegie, "An Optimized KNN Model for Signature-Based Malware Detection," no. 2, pp. 46–49, 2021.