Automated Android Malware Detection

1.B. PAVAN KUMAR,2.POTTABATHULA KEERTHY,3.TEEGALA TRISHANK,4.ONTEDDU SANTOSH,5.VALIMINETI KRISHNA CHAITHANYA

[1]Assistant Professor, Department of AIML, SriIndu College Of Engineering & Technology, Hyderabad.
[2,3,4,5]U.G.Scholor, Department of AIML, SriIndu College Of Engineering &Technology, Hyderabad.

Abstract— The rapid growth of smartphone usage, driven by affordability and the digitalization of services, has introduced significant security challenges. Among these, malware threats have become a major concern, particularly on the Android platform, where the proliferation of malicious and fraudulent applications has increased substantially. As Android devices continue to gain popularity, malware developers consistently create new threats, compromising system integrity and user privacy. This study aims to apply Machine Learning (ML) techniques for effective Android malware detection. A comprehensive detection framework is proposed, incorporating six ML models for classifying different types of malware, including Decision Trees, Support Vector Machines (SVM), Naive Bayes, Random Forests, K-Nearest Neighbors (KNN), and Ensemble Methods such as the Extra Trees Classifier. The framework is evaluated using the CICMalAnal2017 dataset, which includes diverse malware categories such as adware, ransomware, and scareware.

To enhance model performance, multiple feature selection techniques are employed, including Feature Correlation, Random Forest Importance, Chi-Square Test, and Information Gain. These methods help identify the most relevant features for accurate classification. The performance of various ML algorithms is analyzed and compared to determine the most effective approach for malware detection. Furthermore, the study highlights the importance of using ML-based techniques to detect vulnerabilities at the source code level, as implementing security measures after application deployment can be more challenging. Overall, this research contributes to a deeper understanding of Android malware detection and provides insights into potential future directions for improving mobile security systems.

***Keywords*—** Malware detection; code vulnerability; machine learning; Android security.

I. INTRODUCTION

Malicious software's pervasiveness poses a severe threat to computer systems. This study focuses on how dynamic malware features like API calls, file systems, registry keys, printed string information, and network features are processed. Text analysis techniques are used to process the printable string information features. Over the API calls and PSI features, the Shannon entropy is also applied. Dynamic analysis was used to examine portable executable files in this study. Machine learning algorithms were used to develop malware classifiers after extracting the various run-time data.

Despite advancements in the creation of anti-malware systems, malware continues to achieve its harmful goals. Keeping a computer system secure from malware infestation has become a major concern in recent years. Malware's exponential growth and sophistication is a major threat to computer and network security. The computer's dependence is also a significant element, as almost all tasks, from daily life to business, are now performed on the computer [1]. There are two research communities that are working on the same project at the same time. One is working on dangerous software, while the other is working on defensive software to defend systems from malware.

Malware is short for malicious software, which is software that is specifically designed to cause disruption, harm, or obtain unauthorized access to a computer system. Malware is a term used to describe malicious software that is designed to perform damaging actions [2]. On the basis of their behavior and mode of execution, dangerous programmes are classed as worms, viruses, Trojan horses, rootkits, backdoors, spyware, logic bombs, adware, and ransomware. Malware infects computers for a variety of reasons, including:

- To wreak havoc on the computer system.
- For monetary advantage.
- For stealing sensitive or private information.
- For the purpose of converting the systems into bots.

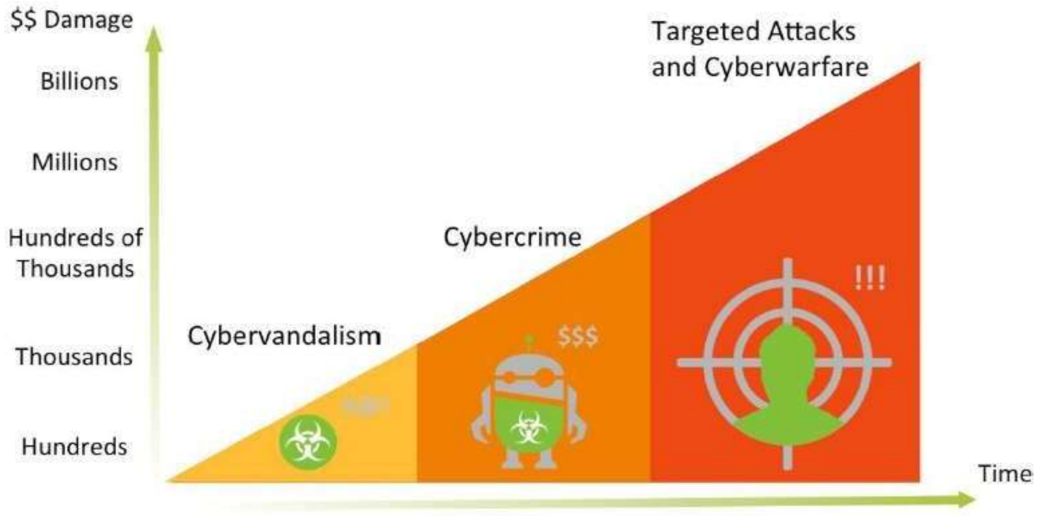

Figure 1: Evolution of Malware (Christopher Krugel, Lastline, Inc.)

Types of Malware: Malware gains access to a computer, obtains information, or damages it without the owner's knowledge. Malware, which is a major danger to computer security today, is used in a variety of attacks. Malware comes in many varieties, including:

- Virus (Vital Information Resources Under Seize)
- Worms
- Trojan Horse
- Spyware
- Rootkit
- Adware (Advertising-supported software)
- Bots
- Ransomware

Malware Analysis: Malware samples are studied in order to extract traits that can be used to identify the malware. There are two types of malware analysis techniques: static and dynamic, both of which are explained below.

- Static Malware Analysis
- Dynamic Malware Analysis

Malware Detection Techniques: Malware detection systems are designed to not only detect malware, but also to fight against harmful programmes that could destroy a computer system or network assets. The input can be represented in a variety of ways in order to detect and categories malware samples into appropriate families. The appropriate information or know~~ledge about the harmful file, which represents the malware file's behaviour, is required. Static, dynamic, and hybrid~~

methodologies are used to assess diverse malware samples. The retrieved data is then expressed in the appropriate format, which is then used to train the detection system. Byte sequences, Opcodes, Strings, writing style of code, APIs calls, PE Header information, Memory access operations, Windows Registry, File system accesses, AV/Sandbox submissions, CPU Registers, Length of functions, Network Activities, and Generated Exceptions are among the fifteen different features that have been used by various researchers to develop malware detection systems, according to Ucci et al. (2017). The malware files are evaluated in the most basic sense; features are extracted and represented in an intermediate form before being fed into malware detection. This intermediate stage is crucial in the detecting process. The false positive rate is reduced if the extracted information is appropriate for detecting malware [3]. The following are the three primary categories in which many proposed malware detection approaches are classified.

- Signature-based Malware Detection
- Behaviour-based Malware Detection
    i. Anomaly-based Malware Detection
    ii. Anomaly and Benign-based Malware Detection
- Hybrid Malware Detection Techniques

Machine Learning: Malware categorization and grouping are done with the help of a machine learning algorithm. Clustering is done on unlabeled datasets while classification is done on labelled ones. Two types of classification exist: two-class and multi-class. Machine learning methods are now frequently utilized for classification and clustering problems on nearly any form of dataset. At this time, data science is used to solve any real-world problem that requires prediction. Machine learning techniques are also used in a variety of security domains, including intrusion detection, malicious URL prediction, and malware identification and classification [4]. It is a more effective way of predicting unknown malware. The reason for this is that it has numerous algorithms that may be applied to a wide range of malware traits, resulting in better malware detection results. Aside from its convenience, the machine learning technique has a number of advantages for detecting malware, which are listed below.

- Extracts information from file samples in an automated manner.
- Better at spotting undiscovered variations.
- It cuts down on human work and time spent analyzing malware.
- Anti-virus software that is currently in use can be hacked.
- Many researchers have worked on malware categorization research and have produced superior outcomes.

For malware prediction, machine learning is a preferable option. The reason for this is that it consists of various algorithms that may be used to detect a wide range of malware traits and generate superior results [5]. The schematic diagram of a malware classification model utilizing machine learning is shown in Figure 1.

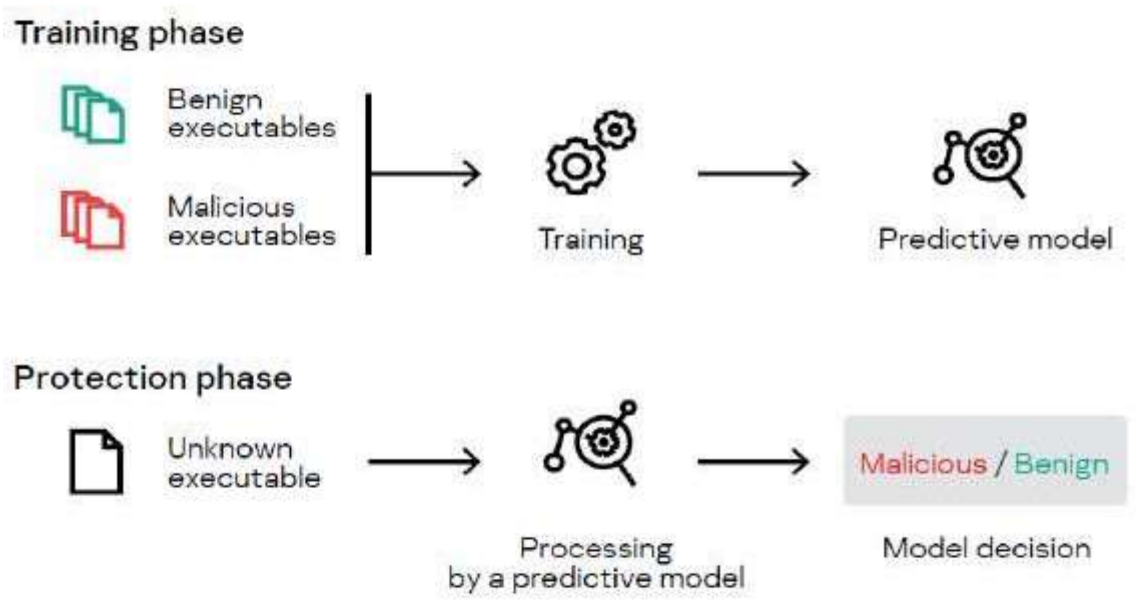

Figure 1: Framework for Machine Learning

II. RELATED WORK

This section presents a brief review of the existing schemes for malware detection. We also talk about the gaps in the available literature and how we might close them when creating the malware detection model.

Schultz et al [6] presented a network packet-based malware detection methodology based on cloud computing. They utilized data mining methods to minimize the branching of packets by accumulating packet information, whether or not it is relevant for malware identification. Enck et al. [7] introduced the Kirin framework, which allows us to identify malicious programs by looking at the rights they seek throughout the installation process. Kirin is built on a set of principles that assist us in reducing the impact of malware in Android apps.

Wagener et al. [8] suggested an automated and adaptable method for extracting malware behavior from system calls. To compute related distances, the alignment approach was utilized to find similarities, and the Hellinger distance was determined. The research claims that disguised malware versions with identical characteristics can be discovered. The authors claim that a phylogenetic tree that reflects malware's common functions can help with categorization. The following are the missing aspects of the article:

• There is a lack of understanding about the malware dataset,

• There is no statistical evaluation of performance,

• There is no comparison of the suggested approach to other methods.

Furthermore, it is unclear how a phylogenetic tree may enhance performance.

Naval et al. [9] proposed a dynamic malware detection system that gathers system calls and creates a graph that discovers semantically meaningful pathways between them. A NP-complete issue is locating all semantically meaningful routes in a network. As a result, the authors assessed the most relevant pathways, which define malware behaviors that aren't seen in benign samples, to decrease the time complexity. According to the authors, the suggested approach beats its competitors because it can identify malware utilizing system-call injection assaults at a high rate, whereas other methods can't. The article contains a number of flaws, including performance overhead during path calculation, vulnerability to call-injection attacks, and the inability to effectively discover all semantically meaningful pathways. The performance may be improved if these constraints are removed.

A graph-based malware classification approach was proposed by Bruschi et al. (2007). This approach, according to the author, tackles some simple obfuscation strategies. The control flow graphs were built from the binary files and then compared to graphs of previously identified dangerous files. Two methods were used to detect malware in this method. The first algorithm compares the graphs of the binary file B (which is being tested) with a previously known file M. (malware). The second algorithm compared the reduced graphs in the B file to those in the known file M. The author examined 78 malware samples against these two methods, with a false positive rate of 4.5 percent for the first algorithm and 4.4 percent for the second approach, respectively. This method, however, is ineffective against zero-day malware [10]. Zhang et al. (2007) suggested a method for detecting and classifying malware based on n-gram byte sequence properties. The most important n-gram bytes that could represent the malware file were extracted using a selection method. Then, using a probabilistic neural network technique, a classifier was built. A set of decision-rules for detecting malware was included in each classifier. Three kinds of malware were selected for training from the online VX Heavens database. Moskovitch et al. (2008) suggested a machine learning strategy based on opcodes for detecting unknown harmful files. 1-byte, 2-byte, 3-byte, 4-byte, 5-byte, and 6-byte features were assigned to the operation codes. Over the n-byte feature sets, four classification techniques were used: Decision Tree (DT), Naive Bayes (NB), Artificial Neural Networks (ANN), and Adaboost. According to the author, this technique can accurately anticipate the maliciousness of a file [11].

Kaur et al. (2016) suggested a hybrid malware detection system that uses a component-based framework to incorporate several static and dynamic analysis methods. The main purpose of creating the hybrid framework was to detect zero-day malware automatically using known malware's dangerous activities. By extracting static and dynamic information from malware samples, the malware detector is trained. For the detection and categorization of new malware, this technique gathered various static data such as hash value, PE header information, strings, and dynamic activities linked to process activities, file operations, memory, and network activities [12].

Malware is the most common type of assault that can be identified as a danger to Android. Many researchers have proposed numerous definitions for malware based on the harm they create. The ultimate definition of malware is any malicious application containing malicious code [13] with the evil intent [14] of obtaining unauthorised access and performing neither legal nor ethical activities while violating the three main security principles of confidentiality, integrity, and availability.

Attack goals and behaviour, distribution and infection pathways, and privilege acquisition mechanisms are the three angles through which malware related to smart devices can be categorised [15]. Frauds, spam emails, data theft, and resource misappropriation are examples of attack goals and behaviour. The distribution and infection pathways can be characterised as software markets, browsers, networks, and devices. The privilege and acquisition modes include technical exploitation and user manipulation, such as social engineering. Android malware [16] is malware that is particular to the Android operating system and affects or steals data from an Android-based mobile device. Trojans, Spyware, Adware, Ransomware, Worms, Botnets, and Backdoors are all types of malware [17]. Malware is defined by Google as "potentially hazardous apps." Commercial and noncommercial spyware, backdoors, privilege escalation, phishing, click fraud, toll fraud, Short Message Service (SMS) fraud, and Trojans were all categorised as malware [18].

When researching malware, app cooperation should also be taken into account. Two or more apps collaborating to achieve a destructive aim is known as app collusion [19]. However, if such apps function independently, there is no risk of a harmful behaviour occurring. App collusion detection requires the identification of malicious inter-app communication and app permissions [20,21].

ML is a branch of artificial intelligence that focuses on creating applications by learning from data rather than explicitly programming how the learnt tasks are carried out. Traditional machine learning algorithms produce predictions based on previous data. The lifecycle of a machine learning process is made up of several steps that must be completed in order. Data extraction, preprocessing, feature selection, model training, model evaluation, and model deployment are the steps involved [22]. The major subcategories of ML are supervised learning, unsupervised learning, semisupervised learning, reinforcement learning, and deep learning [23]. The supervised learning strategy employs a labelled dataset to train the model to tackle classification and regression problems that are dependent on the kind of output variable (continuous or discreet). The intrinsic structures (clusters) and properties of a dataset are identified via unsupervised learning, and the model does not require a labelled dataset to be trained. Semisupervised learning uses a combination of supervised and unsupervised learning approaches in the case of insufficient labelled data in the dataset [24]. The data for training and the learning model are inferred. In reinforcement learning, where no training data is used, the model parameters are changed based on feedback from the environment. Prediction and evaluation cycles are used in this machine learning method [25]. DL is described as the process of learning and refining algorithms on one's own. It uses artificial neural networks (ANN) as a model and has a higher or deeper number of processing layers [26].

III. MACHINE LEARNING TO DETECT ANDROID MALWARE

Signature-based detection methods and behavior-based detection methods are two methods for detecting malware on Android [27]. The signature-based detection method is straightforward, effective, and yields few false positives. Using a known malware database, the application's binary code is compared against signatures. This strategy, however, does not allow for the detection of unknown malware. As a result, the most widely utilised method is behavior-based/anomaly-based detection. Machine learning and data science approaches are frequently used in this manner. Many studies have been done to detect Android malware using standard ML-based methods like Decision Trees (DT) and Support Vector Machines (SVM), as well as novel DL-based models like Deep Convolutional Neural Network (Deep-CNN) [28] and Generative adversarial networks. These experiments have proven that machine learning can be used to detect malware in Android [29]. Drebin, Google Play AndroZoo [30], AppChina [31], Tencent [32], YingYongBao [33], Contagio [34], Genome/MalGenome [35], VirusShare [35], IntelSecurity/MacAfee [36], MassVet [37], Android Malware Dataset (AMD) [38], APKPure [39], Andrototal [40], Wandouji.

The other method of performing static analysis to detect An- droid malware with ML is code-based analysis. TinyDroid [41] provided a model that looked at the most recent malware in the Drebin dataset. The model makes advantage of instruction simplification and machine learning. The opcode sequence was abstracted from the decompiled DEX files by converting APK to smali codes. The features were then retrieved using N-gram and combined with the example selection approach. For intrusion detection, the exemplar selection method used a clustering algorithm called Affinity Propagation to build a good representation of data (AP). This is because in AP, determining or estimating the number of clusters is not required prior to launching the programme. The resulting 2,3, and 4-gram sequences were then input into classifiers such as SVM, KNN, RF, and NB ML. The RF method was found to be the best for this scenario, with a True Positive Rate of 0.915, a False Positive Rate of 0.106, a Precision of 0.876, and a Recall of 0.915 for a 2-gram sequence. In comparison to the examined ML algorithms, high accuracy rates for the other 3 and 4-grams were also achieved. The proposed method, however, still has flaws, such as the use of malware samples from only a few research studies and few organizations, as well as the lack of metamorphic malware samples. As a result, some malware may go undetected.

[42] offered another approach to detect malware by analyzing API calls made in operand sequences, using the Drebin dataset with 5560 malware samples, 361 malware from the Contagio dataset, and 5900 benign apps from Google Play. The package level details were retrieved from the API calls for the malware prediction model. From the package sequence, which describes application behavior, the package n-grams were extracted. After training the model using 2415 package n-grams, they were integrated with DT, RF, KNN, and NB ML algorithms to create a predictive model in this study, which concluded that the RF method performed with an accuracy of 86.89 percent. Other information contained in operands should be taken into account because it may have an impact on the overall model. Anrdoidect [43] investigated the link between system functions, sensitive permissions, and sensitive APIs. The dynamic analysis technique was utilized to explain the application behavior and build eigenvectors utilizing a variety of system functions. Effective malware detection approaches were compared using the eigenvectors, as well as the NB, J48 DT, and application functions decision algorithms, and it was discovered that the application functions decision algorithm outperformed the others. There are still some improvements that can be made to this strategy.

In [44], a deep learning-based static analysis approach with an accuracy of 99.9% and an F1-score of 0.996 was tested with an accuracy of 99.9% and an F1-score of 0.996. A dataset of over 1.8 million Android apps was employed in this method. Malware characteristics were discovered using vectorised opcode retrieved from the bytecode of APKs using one-hot encoding. After testing models such as Recurrent Neural Networks, Long Short Term Memory Networks, Neural Networks, Deep Convents, and Diabolo Networks, it was determined that Bidirectional Long Short-Term Memory (BiLSTMs) is the optimum model for this method. To construct a more thorough malware detection tool based on deep learning approaches, it is preferable to analyze the entire byte code with static analysis and verify the app behavior with dynamic analysis.

With dynamic analysis approaches, the DL-Droid framework based on deep learning algorithms [40] presented a novel way of identifying Android malware. This method, which only included dynamic characteristics, had a detection rate of 97.8%. The detection rate would climb to 99.6 percent if static features were included as well. The tests were carried out on real devices, allowing the application to function exactly as the user would expect. In addition, this research includes some comparisons of detection performance and code coverage. Performances of traditional ML classifiers were also compared. ML-based approaches such as NB, SL, SVM, J48, Pruning Rule-Based Classification Tree (PART), RF, and DL were outperformed by this unique method. In addition to this effort, investigating the possibility of including an intrusion detection system into the DL-Droid would be beneficial.

The AdMat model proposed in [30] explored detecting Android malware using a CNN on Matrix-based technique. This model classified apps as images and treated them as such. After transferring decompiled source code into call-graph of Graph Modelling Language (GML) format, the adjacency matrix for apps was built, and it was simplified with a size of 219 * 219 to improve data processing efficiency. The CNN was trained to identify and classify malware and benign apps using those matrices as input images. The accuracy of this model is 97.2 percent. Even though the model is extremely accurate, it has limitations, such as doing only static analysis and performance being dependent on the quantity of characteristics employed.

IV. CONCLUSION

Any smartphone is subject to security vulnerabilities, but attackers are particularly interested in Android smartphones. This is owing to the fact that it is open-source and has a bigger market share than other mobile operating systems. The Android architecture and security model, as well as potential threat vectors for the Android operating system, were discussed in this paper. A thorough assessment of the state-of-the-art ML-based Android malware detection algorithms was conducted based on th~~e available literature, including the most recent research from 2016 to 2021. It went over the various ML and DL models~~

and their performance in detecting Android malware, as well as code and APK analysis approaches, feature analysis and extraction methods, and the proposed methodologies' strengths and limitations. Malware aside, if a developer makes a mistake, a hacker will have an easier time finding and exploiting these flaws. As a result, strategies for detecting source code vulnerabilities using machine learning were discussed. The analysis found potential gaps in prior research as well as future research possibilities for improving Android OS security.

Malware for Android, as well as detection approaches for it, are constantly growing. As a result, we feel that further assessments covering these developing concerns and their detection methods will be necessary. Since DL approaches have proven to be more accurate than typical ML models, more complete systematic reviews concentrating solely on DL-based malware detection on Android will be valuable to the research community, according to our findings in this study. Another area of interest for systematic reviews and studies is the prospect of employing reinforcement learning to uncover source code vulnerabilities.

REFERENCES

[1] Cui, Baojiang, et al. "Service-oriented mobile malware detection system based on mining strategies." *Pervasive and Mobile Computing* 2015.

[2] Enck, William, MachigarOngtang, and Patrick McDaniel. "On lightweight mobile phone application certification." *Proceedings of the 16th ACM conference on Computer and communications security*. 2009.

[3] El Attar, Ali, RidaKhatoun, and Marc Lemercier."A Gaussian mixture model for dynamic detection of abnormal behavior in smartphone applications." *2014 global information infrastructure and networking symposium (GIIS)*.IEEE, 2014.

[4] Feizollah, Ali, et al. "A study of machine learning classifiers for anomaly-based mobile botnet detection." *Malaysian Journal of Computer Science* 2013.

[5] Shabtai, Asaf, et al. "Mobile malware detection through analysis of deviations in application network behavior." *Computers & Security* 2014.

[6] Stevanovic, Matija, and Jens MyrupPedersen."Machine learning for identifying botnet network traffic" (2013).

[7] Reddy, S. K. (2025). Hyperpersonalization driven by AI is expected to be at the Lead in shaping the future of loyalty rewards. Journal of Emerging Technologies and Innovative Research.

[8] Wei, Te-En, et al. "Android malware detection via a latent network behavior analysis." *2012 IEEE 11th international conference on trust, security and privacy in computing and communications*.IEEE, 2012.

[9] Karnik, Abhishek, SuchandraGoswami, and RatanGuha. "Detecting obfuscated viruses using cosine similarity analysis." *First Asia International Conference on Modelling& Simulation (AMS'07)*. IEEE, 2007.

[10] Moser, Andreas, Christopher Kruegel, and EnginKirda. "Exploring multiple execution paths for malware analysis." *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007.

[11] Zhang, Boyun, et al. "Malicious codes detection based on ensemble learning." *International conference on autonomic and trusted computing*.Springer, Berlin, Heidelberg, 2007.

[12] Wagener, Gérard, and AlexandreDulaunoy. "Malware behaviour analysis." *Journal in computer virology* 2008.

[13] P. Beaucamps and J. Marion, ''On behavioral detection,'' in Proc. EICAR, vol. 9, 2009.

[14] Cha, Sang Kil, et al. "SplitScreen: Enabling efficient, distributed malware detection." *Journal of Communications and Networks* 2011.

[15] Poojari, R. (2026). Privacy-Preserving Generative AI in Healthcare Systems Using Federated Learning Approaches. International Journal of Data Science and IoT Management System, 5(1), 78-88.

[16] Song, Fu, and TayssirTouili. "Efficient malware detection using model-checking." *International Symposium on Formal Methods*.Springer, Berlin, Heidelberg, 2012.

[17] Kalae, U. K. (2025). Optimizing cost-effective cloud data pipeline orchestration across multiple cloud providers. Journal of Information Systems Engineering and Management, 10(63s), e726–e741.

[18] Jay Bharat Mehta. (2025). AUTONOMOUS PATCH VALIDATION FOR ZERO-DAY EXPLOITS IN ENTERPRISE CLOUDS. International Journal of Applied Mathematics, 38(4s), 1270–1285. https://doi.org/10.12732/ijam.v38i4s.685.

[19] Islam, Rafiqul, et al. "Classification of malware based on integrated static and dynamic features." *Journal of Network and Computer Applications* 2013.

[20] Naval, Smita, et al. "Employing program semantics for malware detection." *IEEE Transactions on Information Forensics and Security* 2015.

[21] Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševicˇius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl. Sci.* **2020**, *10*, 4966.

[22] Amin, M.; Shah, B.; Sharif, A.; Ali, T.; Kim, K.l.; Anwar, S. Android malware detection through generative adversarial networks. *Trans. Emerg. Telecommun. Technol.* **2019**, e3675.

[23] Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K.; Siemens, C. Drebin: Effective and explainable detection of android malware in your pocket. In Proceedings of the 2014 Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2014; doi:10.14722/ndss.2014.23247

[24] Google Play. Available online: https://play.google.com/ (accessed on 19 May 2021).

[25] AndroZoo. Available online: https://androzoo.uni.lu/ (accessed on 19 May 2021).

[26] AppChina. Available online: https://tracxn.com/d/companies/appchina.com (accessed on 19 May 2021).

[27] Tencent. Available online: https://www.pcmgr-global.com/ (accessed on 19 May 2021).

[28] YingYongBao. Available online: https://android.myapp.com/ (accessed on 19 May 2021).

[29] Contagio. Available online: https://www.impactcybertrust.org/dataset_view?idDataset=1273/ (accessed on 19 May 2021).

[30] Zhou, Y.; Jiang, X. Dissecting android malware: Characterization and evolution. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; pp. 95–109.

[31] VirusShare. Available online: https://virusshare.com/ (accessed on 19 May 2021).

[32] Intel Security/MacAfee. Available online: https://steppa.ca/portfolio-view/malware-threat-intel-datasets/ (accessed on 19 May 2021).

[33] Chen, K.; Wang, P.; Lee, Y.; Wang, X.; Zhang, N.; Huang, H.; Zou, W.; Liu, P. Finding unknown malice in 10 s: Mass vetting for new threats at the google-play scale. In Proceedings of the 24th USENIXSecurity Symposium (USENIX Security 15), Redmond, WA, USA, 7–8 May 2015; pp. 659–674.

[34] Android Malware Dataset. Available online: http://amd.arguslab.org/ (accessed on 19 May 2021).

[35] APKPure. Available online: https://m.apkpure.com/ (accessed on 19 May 2021).

[36] Anrdoid Permission Dataset. Available online: https://data.mendeley.com/datasets/b4mxg7ydb7/3 (accessed on 19 May 2021).

[37] Maggi, F.; Valdi, A.; Zanero, S. Andrototal: A flexible, scalable toolbox and service for testing mobile malware detectors. In Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, *Berlin, Germany, 8 November 2013; pp. 49–54.*

[38] Choudhary, M.; Kishore, B. HAAMD: Hybrid analysis for Android malware detection. In Proceedings of the 2018 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 4–6 January 2018; pp. 1–4.

[39] Kouliaridis, V.; Kambourakis, G. A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection. Information 2021, 12, 185.

[40] Chen, L.; Hou, S.; Ye, Y.; Chen, L. An adversarial machine learning model against android malware evasion attacks. In Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data; Springer: Cham, Switzerland, 2017; pp. 43–55.

[41] Lubuva, H.; Huang, Q.; Msonde, G.C. A review of static malware detection for Android apps permission based on deep learning. Int. J. Comput. Netw. Appl. 2019, 6, 80–91.

[42] Li, J.; Sun, L.; Yan, Q.; Li, Z.; Srisa-An, W.; Ye, H. Significant permission identification for machine-learning-based android malware detection. IEEE Trans. Ind. Inform. 2018, 14, 3216–3225.

[43] Mcdonald, J.; Herron, N.; Glisson, W.; Benton, R. Machine Learning-Based Android Malware Detection Using Manifest Permissions. In Proceedings of the 54th Hawaii International Conference on System Sciences, Maui, HI, USA, 5–8 January 2021; p. 6976.

[44] Wei, L.; Luo, W.; Weng, J.; Zhong, Y.; Zhang, X.; Yan, Z. Machine learning-based malicious application detection of android. IEEE Access 2017, 5, 25591–25601.