Journal of Theoretical and Applied Information Technology
30th June 2024. Vol.102. No. 12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DEEP LEARNING BASED MALWARE DETECTION

T. SUSHMA[1], SIRISHA NARKEDAMILLI[2], MADHAVA RAO CHUNDURU[3], VADDEMPUDI SUJATHA LAKSHMI[4], G. BALU NARASIMHA RAO[5], PRABHAKAR KANDUKURI[6]

[1]Department of ECE, Prasad V Potluri Siddhartha Institute of Technology, Vijayawada, India

[2]Department of EEE, Aditya College of Engineering and Technology, Surampalem, India

[3]Department of CSA, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India

[4]Department of Computer Applications, RVR&JC College of Engineering, Guntur, India

[5]Department of CSE, Vignan's Foundation of Science, Technology & Research, Guntur, India

[6]Department of AI&ML, Chaitanya Bharathi Institute of Technology, Hyderabad, India

tsushmaece@gmail.com , sirisha.narkedamilli@acet.ac.in, cmadhavarao@kluniversity.in, sujathavdmpudi@gmail.com, balunarasimharao@gmail.com , prabhakarcs@gmail.com

ABSTRACT

Malware detection is a critical aspect of cybersecurity, aiming to identify and mitigate malicious software designed to harm or exploit any programmable device or network. Traditional methods of malware detection, such as signature-based techniques, have limitations in dealing with the sophisticated and rapidly evolving nature of modern malware. This paper explores the application of deep learning, a subset of artificial intelligence, in enhancing malware detection capabilities. By leveraging deep learning models, which can automatically learn and extract features from data, we can improve detection accuracy and adapt to new, unseen malware. This research reviews various deep learning architectures and methodologies employed in malware detection, evaluates their effectiveness, and discusses future directions and challenges in the field.

Keywords: *Malware detection, deep learning, cybersecurity.*

1. INTRODUCTION

In the digital era, a significant number of computing devices have been impacted by malware. Malware, often known as malevolent software, is specifically designed to fulfill the harmful objectives of a malicious attacker. Malicious software, or malware, has the ability to infiltrate networks, cause harm to critical infrastructure, compromise the security of computers and smart devices, and unlawfully obtain confidential information [1].

The concept of an information society has developed due to the emergence of the Internet of Things (IoT) and its various uses. Nevertheless, the benefits of this industrial progress are impeded by security concerns, as hackers selectively target individual computers and networks to illicitly obtain confidential data for monetary purposes and disrupt operations [2]. These attackers employ harmful software, also known as "malware," to exploit system weaknesses and present significant risks. Malware, also referred to as malicious software, is a type of computer software specifically designed to cause harm to an operating system [3]. The frequency of malware attacks has greatly risen due to the substantial changes in our daily contacts caused by the advancements in mobile technologies. Mobile devices connected to the Internet provide many services such as online learning, social networking, online banking, online shopping, and web browsing. Mobile devices have thus played a pivotal role and have transformed into an essential component of everyday life [4]. As of 2020, the global mobile device user count stands at 4.78 billion [5]. While mobile devices offer convenience to consumers, they are susceptible to virus infiltration and attacks due to their connection to online social networks and services. Mobile malware has the ability to masquerade as regular code and

thereafter modify any intended program in order to corrupt and impede the functioning of the system [5,6,7].

Google Play has implemented a permission-based approach as a security safeguard to prevent applications from accessing private data. This permission prompts users to install the program after considering the assets that have been accessed. Prior to proceeding with the installation, it is imperative that the users explicitly acknowledge and agree to the terms of the agreement. Regrettably, the Google Play technique does not provide complete protection for the customer since they often have a propensity to approve the agreement without thoroughly perusing the permission [5,8]. Another potential threat arises from the exploitation of profitable Android applications, as seen by the significant rise in the detection of Android malware, which increased more than tenfold from 2012 to 2018 [9]. In addition, a total of more than 12,000 new instances of Android malware were discovered per day during the year 2018. The newly discovered Android malware samples exhibit greater sophistication compared to those that emerged a few years ago, particularly in their ability to evade antivirus detection through coding and encryption. Additionally, there has been a significant increase in the spread of malware [10,11].

The utilization of machine learning in malware detection studies is becoming increasingly popular due to its effectiveness in achieving a high level of accuracy in detecting malware [12]. Prior research has employed machine learning (ML) algorithms, which have the ability to make decisions based on learned patterns from data. Machine learning refers to the idea of reducing the need for human involvement in computer systems [13]. Machine learning utilizes computer learning algorithms and historical data to make predictions. Supervised and unsupervised learning approaches [14,15] are utilized to examine the characteristics and monitor the model. In both scenarios, the machine acquires the ability to differentiate between harmful and harmless actions. In supervised learning, the machine learning model is provided with both the input data and the desired outputs. It then learns to accurately classify malware patterns as "malware" and normal behaviors as "normal". The training phase is iterated until the model achieves perfect accuracy in predicting all samples [5]. Various machine learning algorithms, such as support vector machines (SVM) [16,17,18], K-nearest neighbor (KNN) [19,20], Bayesian estimates [21,22], genetic algorithms [23], have been employed to construct malware detection systems. Unsupervised learning approaches involve providing inputs without any predetermined targets, allowing the machine learning system to learn how to differentiate between malware and benign samples. Nevertheless, certain investigations integrated the approaches of supervised and unsupervised learning [24].

Malware detection is a crucial aspect of security that is closely linked to the legal, reputational, and economic interests of companies. Utilizing deep learning as a technique for developing and enhancing detection methods is an effective approach to address many challenges associated with malware detection. However, in the realm of deep learning, there are numerous complex factors that must be taken into account when considering detection strategies. Correlation-based feature selection, the dense layer model, and the LSTM model are three complex and symmetrical approaches that can significantly impact performance.

Two distinct datasets will be utilized in the ongoing research. One of the datasets contains a substantial quantity of records, whereas the other dataset comprises a significant number of predictors (attributes). The process of selecting the most optimal qualities will be employed in various situations to determine the most effective combination of features. The correlation between the target property "classification" will be utilized as the way for selecting features. The training phase will involve the utilization and comparison of Dense and LSTM models. Multiple training scenarios will be set up based on various feature selection criteria, splitting criteria, and dataset topologies. Our primary innovation lies in using the efficacy of deep learning and feature selection techniques in the domain of malware detection to construct a resilient, high-performing, computationally efficient malware detection system.

2. METHODOLOGY

This research employs both static and dynamic analysis methods using deep learning models. The dataset comprises a mix of known malware samples and benign software, sourced from public repositories like VirusTotal and Malimg dataset. Multiple deep learning (DL) methods are suggested and employed in this

work. To train the deep learning models using the two chosen datasets, it is necessary to preprocess these datasets. This preprocessing phase involves encoding (numbering) the classification (target) columns and handling any special characters or missing values. Due to the distinct nature of the two datasets, the preparation stages will vary.
Once the datasets have undergone preprocessing, they are divided into separate training and test sets. Feature selection is conducted prior to the training process in certain training scenarios to reduce data dimensionality and computational time.

Subsequently, the DL models will be constructed and trained using several training scenarios, encompassing diverse splitting criteria, distinct DL architectures, and the option of feature selection. Figure 1 depicts the recommended technique for both datasets.

The objective of feature selection is to identify the most optimal characteristics relevant to the topic being examined, with the purpose of minimizing computational time. Our study proposes a correlation-based technique to address the issues of large dimensionality and long processing time. This approach also aims to pick the most effective combinations of features, hence improving the performance of the training and evaluation process.

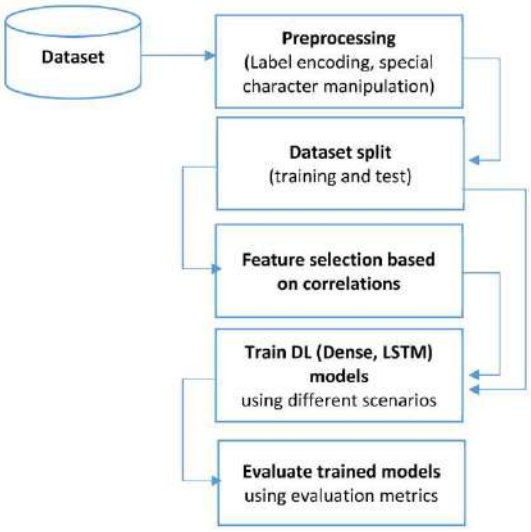

Figure 1. Illustrates the methodology that is being suggested

Next, a compilation of probable columns to be dropped is generated. Various selection scenarios can be generated as the correlation spans from 0 to 1. The selection process is determined by the desired number of columns. We will extract the K necessary features and discard the remaining ones. For the second dataset, the identical method will be utilized, with the exception of the selection phase. Columns will be eliminated in the second dataset based on specific correlation thresholds (T), given that there are 214 columns in total. In the second dataset, the number of selected features is contingent upon the chosen threshold, unlike the first dataset where the threshold is not specified.

3. RESULTS AND DISCUSSION

This section presents and discusses the findings of the experiments done to assess the effectiveness of the autoencoder-based malware detection approach. The current section is dedicated to providing pertinent information on the experimental setup. It is divided into three sections to address the following aspects: the configuration of the experiment setup, the gathering of data, and the specifics of the training process. To obtain additional details regarding the experimental setting, go to table 1. The tests were conducted on a machine equipped with an Intel CoreTM i5-8300 processor, 16GB of RAM, and a GeForce GTX 1060 MQ graphics card. The computer operated on a 64-bit iteration of the Windows 10 operating system. In our programming, we employed Keras, Tensorflow 2.1, and Python 3.7. We categorized the datasets according to their intended purpose. (1) Dataset-1 consists of 8,121 malicious programs and 2,000 benign programs. It is utilized for training and evaluating AE-1 models. The AE-2 model is trained, validated, and tested using Dataset-2, which consists of 8121 dangerous applications and 7015 safe ones. The AE-2 model is tested using Dataset-3, which consists of 5,384 malicious applications and 5,000 safe programs, to evaluate its ability to detect unfamiliar malware. It is important to note that when we divided Dataset-2 and Dataset-3, we intentionally incorporated older software samples in Dataset-2 for the purpose of training, such as malware from 2016.

In Dataset-3, we included more recent releases, such as those from 2017 and 2018. Simulating the condition of identifying newly published software samples will facilitate future analysis of the model's performance. In order to evaluate the autoencoder's ability to reconstruct feature images, we utilize the AE-1 network. The specific attributes of the AE-1 model are presented in Table 2. During the training process, we utilize the Adam optimization technique with a total of 100 epochs and a learning rate of 1e-4.

The AE-1 network undergoes training using the DTrain dataset and subsequently undergoes testing using the DTest_mal and DTest_benign datasets, which contain malicious and benign software, respectively. In order for a test set to have a low reconstruction error, the new input must be similar to the input of the training dataset. Conversely, if the new inputs deviate from the inputs used in the dataset during training, a noticeable reconstruction error will be observed in this test set. The primary objective of our experiment is to investigate the significant disparity in error data produced by these two test sets after AE-1. In practical terms, the significant duplication features in the software dataset and the distinct functional traits exhibited by malware families in the malware dataset can result in experimental outcomes showing substantial fluctuations. This is because our hypothesis is founded on the notion that malware is universally similar, while benign software is not. Consequently, we place less importance on the exact errors exhibited by the two test sets and instead focus more on the comparative disparities between them.The responsibility of evaluating the performance of the detection model lies with the AE-2 network. We partitioned Dataset-2 into two equal parts, allocating 80% for training and 20% for testing. During the training process, we employed k-fold cross-validation with a value of k equal to 6 in order to train and validate the training set. Consequently, we allocated 5/6 of the training set for training purposes and reserved 1/6 for validation. We conducted this procedure on six occasions prior to calculating the average. The test set is used for testing purposes during the entire testing procedure. The duration of training is quantified in units of minutes. The training of AE-2 utilized the Adam optimization technique with a learning rate of 0.0001 and 100 epochs.We evaluate the effectiveness of this strategy by analyzing the overall error distribution in both malicious and benign reconstructions of malware images. Figure 2 shows the error distributions of the combined test sets. The Y-axis indicates the normalized reconstructed error value generated by each program following the encoder network. The error value of each pixel point corresponding to the malware feature image is aggregated and subsequently divided by the total to achieve image normalization. The line statistics graph illustrates the general error trend of DTest_mal with a blue line, whereas the overall error trend of DTest_benign is represented by a yellow line. The inherent unpredictability of the dataset plus the redundancy of the software files result in a non-zero error. Figure 5 illustrates a significant disparity in the average error values between the two datasets. The blue line indicates a consistent and steady error trend for the malware dataset, while the yellow line represents an erratic and fluctuating error trend for the benign software test set. This supports our perspective.

Based on this experiment, we can show that the automatic encoder can identify complex characteristics of both harmless and harmful software and successfully reconstruct the pre-processed malware data. Next, we proceed to carry out the task of differentiating between harmful and benign software.

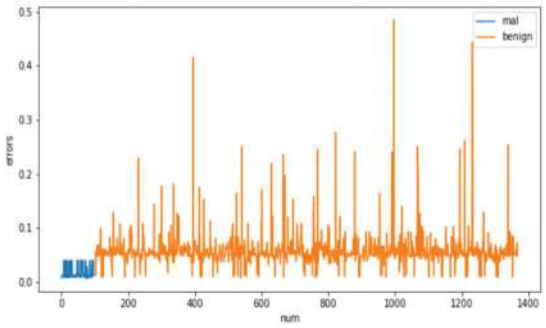

Figure 2. Mistake in reconstruction for two sets of data.

The evaluation of the autoencoder model is conducted using many measures, such as accuracy, precision, recall, F1-score, false positive rate, and false negative rate. These metrics offer a holistic perspective on the model's capacity to accurately detect instances of malware while minimizing both false positives and false negatives. The results are compared with conventional signature-based methods to emphasize the possible enhancements provided by the autoencoder methodology. Signature-based approaches are intrinsically constrained by their dependence on pre-established patterns, rendering them vulnerable to evasion by polymorphic and metamorphic malware. The autoencoder's capacity to acquire knowledge from the inherent characteristics of data without pre-established patterns situates it as a more flexible and adaptable solution.The ROC curves depicted in Figure 3 illustrate the impact of the model on the training set. It is evident that the model demonstrates a consistently reliable performance on the training set. The ROC curves depicted in Figure 4 illustrate the model's performance on the test set, specifically Dataset-2. It is evident that our model surpasses the other

two in performance. In order to thoroughly evaluate the ability of our model to detect previously undiscovered malware, we utilize Datasets-3 as the test set for AE-2. The ROC curves are displayed in Figure 5, revealing that our model exhibits commendable accuracy and a certain degree of viability in detecting previously undetected malware. However, it also exhibits certain limitations as the software evolves over time.

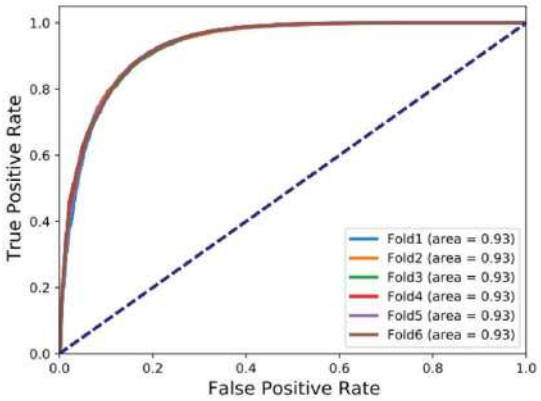

Figure 3. The ROC curve of AE-2 on training set.

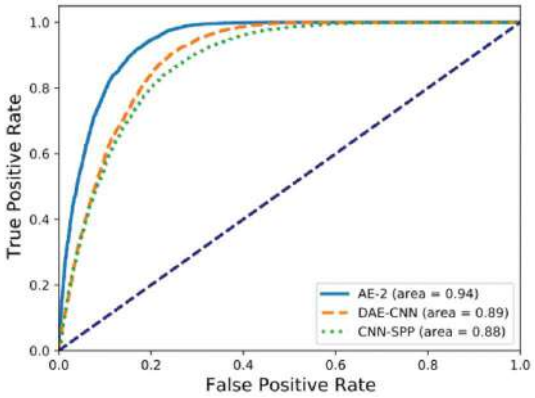

Figure 4. The ROC curve of different models on the test set.

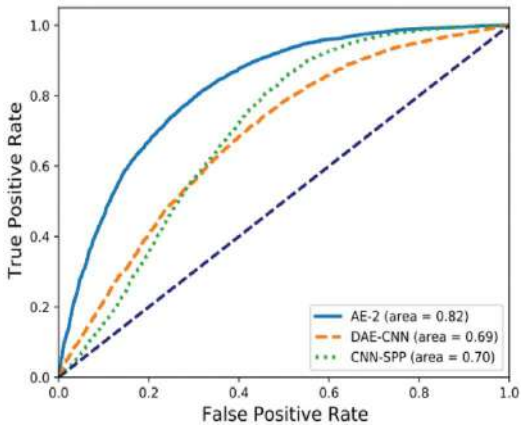

Figure 5. The ROC curve of different models on the unseen software

The results and discussion section illuminates the performance of the autoencoder-based malware detection approaches, offering valuable insights into its strengths, limitations, and implications for the broader cybersecurity landscape.

4. CONCLUSION

The findings of this study emphasize the capacity of autoencoders to enhance the skills of malware detection. Autoencoders provide a promising approach to improving the adaptability and effectiveness of cybersecurity defenses against emerging malware threats by adopting the dynamic and unsupervised learning paradigm. The study findings obtained from the evolving digital landscape contribute to the ongoing effort to develop new and robust solutions for protecting digital ecosystems. The testing results confirm the effectiveness of our proposed approach, which entails converting the bytecode of each software method into a grayscale image that graphically depicts the characteristics of a software sample. Our approach exhibits a notably higher level of accuracy in identifying malicious software compared to methods built using traditional machine learning algorithms. Our technique exhibits decreased training and detection durations when compared to competing malware detection systems that depend on deep learning models. The text outlines suggestions for future research approaches, including investigating ensemble methods that combine autoencoders with other deep learning architectures, including temporal factors to improve dynamic malware detection, and utilizing adversarial training to boost the robustness of models. These recommendations

are intended to provide guidance for future investigations in the continual pursuit of developing more efficient and adaptable malware detection systems.

REFERENCES

[1]. Rathore, H.; Agarwal, S.; Sahay, S.; Sewak, M. Malware detection using machine learning and deep learning. In Proceedings of the International Conference on Big Data Analytics, Seattle, WA, USA, 10–13 December 2018; pp. 402–411. [Google Scholar]

[2]. Nasif, A.; Othman, Z.; Sani, N.S. The deep learning solutions on lossless compression methods for alleviating data load on IoT nodes in smart cities. Sensors 2021, 21, 4223. [Google Scholar] [CrossRef] [PubMed]

[3]. Vinayakumar, R.; Alazab, M.; Soman, K.; Poornachandran, P.; Venkatraman, S. Robust intelligent malware detection using deep learning. IEEE Access 2019, 7, 46717–46738. [Google Scholar] [CrossRef]

[4]. Singh, A.; Kumar, R. A two-phase load balancing algorithm for cloud environment. Int. J. Softw. Sci. Comput. Intell. 2021, 13, 38–55. [Google Scholar] [CrossRef]

[5]. Mat, S.R.T.; Razak, M.A.; Kahar, M.; Arif, J.; Firdaus, A. A Bayesian probability model for Android malware detection. ICT Express 2022, 8, 424–431. [Google Scholar] [CrossRef]

[6]. Yen, S.; Moh, M.; Moh, T.-S. Detecting compromised social network accounts using deep learning for behavior and text analyses. Int. J. Cloud Appl. Comput. 2021, 11, 97–109. [Google Scholar] [CrossRef]

[7]. Shabudin, S.; Sani, N.; Ariffin, K.; Aliff, M. Feature selection for phishing website classification. Int. J. Adv. Comput. Sci. Appl. 2020, 11, 587–595. [Google Scholar] [CrossRef]

[8]. Liu, C.-H.; Zhang, Z.-J.; Wang, S.-D. An android malware detection approach using Bayesian inference. In Proceedings of the 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, Fiji, 8–10 December 2016; pp. 476–483. [Google Scholar]

[9]. GDATA Mobile Malware Report—No let-up with Android malware. 2019. Available online: https://www.gdatasoftware.com/news/2019/07/35228-mobile-malware-report-no-let-up-with-android-malware (accessed on 22 November 2022).

[10]. Qiu, J.; Zhang, J.; Luo, W.; Pan, L.; Nepal, S.; Xiang, Y. A survey of android malware detection with deep neural models. ACM Comput. Surv. 2020, 53, 1–36. [Google Scholar] [CrossRef]

[11]. Sihwail, R.; Omar, K.; Ariffin, K.A.Z. An effective memory analysis for malware detection and classification. Comput. Mater. Contin. 2021, 67, 2301–2320. [Google Scholar] [CrossRef]

[12]. Mat, S.R.T.; Razak, M.A.; Kahar, M.; Arif, J.; Mohamad, S.; Firdaus, A. Towards a systematic description of the field using bibliometric analysis: Malware evolution. Scientometrics 2021, 126, 2013–2055. [Google Scholar] [CrossRef]

[13]. Bassel, A.; Abdulkareem, A.; Alyasseri, Z.; Sani, N.; Mohammed, H.J. Automatic Malignant and Benign Skin Cancer Classification Using a Hybrid Deep Learning Approach. Diagnostics 2022, 12, 2472. [Google Scholar] [CrossRef]

[14]. 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. [Google Scholar] [CrossRef]

[15]. Abdallah, A.; Ishak, M.K.; Sani, N.S.; Khan, I.; Albogamy, F.R.; Amano, H.; Mostafa, S.M. An Optimal Framework for SDN Based on Deep Neural Network. Comput. Mater. Contin. 2022, 73, 1125–1140. [Google Scholar] [CrossRef]

[16]. Han, H.; Lim, S.; Suh, K.; Park, S.; Cho, S.; Park, M. Enhanced android malware detection: An svm-based machine learning approach. In Proceedings of the 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Republic of Korea, 19–22 February 2020; pp. 75–81. [Google Scholar]

[17]. Singh, P.; Borgohain, S.; Kumar, J. Performance Enhancement of SVM-based ML Malware Detection Model Using Data Preprocessing. In Proceedings of the 2022 2nd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), Patna, India, 24–25 June 2022; pp. 1–4. [Google Scholar]

[18]. Droos, A.; Al-Mahadeen, A.; Al-Harasis, T.; Al-Attar, R.; Ababneh, M. Android Malware Detection Using Machine Learning. In Proceedings of the 2022 13th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 21–23 June 2022; pp. 36–41. [Google Scholar]

[19]. Baldini, G.; Geneiatakis, D. A performance evaluation on distance measures in KNN for mobile malware detection. In Proceedings of the 2019 6th international conference on control, decision and information technologies (CoDIT), Paris, France, 23–26 April 2019; pp. 193–198. [Google Scholar]

[20]. Assegie, T.A. An optimized KNN model for signature-based malware detection. Tsehay Admassu Assegie. Int. J. Comput. Eng. Res. Trends (IJCERT) 2021, 8, 2349–7084. [Google Scholar]

[21]. Castillo-Zúñiga, I.; Luna-Rosas, F.; Rodríguez-Martínez, L.; Muñoz-Arteaga, J.; López-Veyna, J.; Rodríguez-Díaz, M.A. Internet data analysis methodology for cyberterrorism vocabulary detection, combining techniques of big data analytics, NLP and semantic web. Int. J. Semant. Web Inf. Syst. 2020, 16, 69–86. [Google Scholar] [CrossRef]

[22]. Yilmaz, A.B.; Taspinar, Y.; Koklu, M. Classification of Malicious Android Applications Using Naive Bayes and Support Vector Machine Algorithms. Int. J. Intell. Syst. Appl. Eng. 2022, 10, 269–274. [Google Scholar]

[23]. Yildiz, O.; Doğru, I.A. Permission-based android malware detection system using feature selection with genetic algorithm. Int. J. Softw. Eng. Knowl. Eng. 2019, 29, 245–262. [Google Scholar] [CrossRef]

[24]. Arora, A.; Peddoju, S.; Chouhan, V.; Chaudhary, A. Hybrid Android malware detection by combining supervised and unsupervised learning. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, New Delhi, India, 29 October–2 November 2018; pp. 798–800. [Google Scholar