MVDroid: an Android malicious VPN detector using neural networks

Saeed Seraj[1], Siavash Khodambashi[2], Michalis Pavlidis[1], and Nikolaos Polatidis[1]*

[1]School of Architecture, Technology and Engineering, University of Brighton, Brighton, BN2 4GJ, U.K
[2]Department of Computer Engineering, Yadegar-e-Imam Khomeini (RAH) Shahr-e-Rey Branch, Islamic Azad University, Tehran, Iran

*Correspondence: N.Polatidis@Brighton.ac.uk

Abstract: The majority of Virtual Private Networks (VPNs) fail when it comes to protecting our privacy. If we are using a VPN to protect our online privacy, many of the well-known VPNs are not secure to use. When examined closely, VPNs can appear to be perfect on the surface but still be a complete privacy and security disaster. Some VPNs will steal our bandwidth, infect our computers with malware, install secret tracking libraries on our devices, steal our personal data, and leave our data exposed to third parties. Generally, Android users should be cautious when installing any VPN software on their devices. As a result, it's important to identify malicious VPNs before downloading and installing them on our Android devices. This paper provides an optimised deep learning neural network for identifying fake VPNs, and VPNs infected by malware based on the permissions of the apps, as well as a novel dataset of malicious and benign Android VPNs. Experimental results indicate that our proposed classifier identifies malicious VPNs with high accuracy, while it outperforms other standard classifiers in terms of evaluation metrics such as accuracy, precision, and recall.

Keywords: Android Malware Detection; Virtual Private Network; VPN; Neural Networks

1. Introduction

VPNs were first designed for employees to virtually connect to their office network from home or while on a business trip. These days, VPNs are used more frequently for privacy and security-related reasons, as well as to conceal online internet traffic, circumvent censorship, or access geo-blocked content. To make it more difficult for anyone on the internet to see which websites or apps a user is accessing, VPNs work by funnelling all user's internet traffic through an encrypted pipe to the VPN server. However, VPNs don't automatically provide anonymity or privacy protection. VPNs merely redirect all user' internet traffic to the systems of the VPN provider rather than the systems of the users' internet service provider.

There are even fake VPN services popping up with the growing interest in VPNs. Since the launch of Android 4 in October 2011, the Android VPN Service class has allowed mobile app developers to create VPN clients through native support. Android app developers only need to request BIND_VPN_SERVICE permission to use the native support for VPN purposes. It enables an app to intercept user traffic and seize complete control over it according to security concerns highlighted by Android's official documentation [1]. Many legitimate apps may use the BIND_VPN_SERVICE permission to provide online anonymity or access censored

content [2]. Android warns users of the potential dangers of the BIND_VPN_SERVICE permission by showing notifications and system dialogues since malicious apps may abuse users' personal information [1]. However, many users might not have the technical knowledge necessary to fully comprehend the risks. Many popular, VPN services will leak a user's IP address or DNS requests, thereby exposing the user's data to third parties. Some VPNs can intrude on privacy and steal private information, install hidden tracking libraries on target devices, infect a user's computer with malware, and even steal the user's bandwidth.

Given the availability of free VPN client apps, which can be readily downloaded from the Google Play store, and their vital role during the pandemic, users must be aware of the inherent risks. Not all VPNs that aid in bypassing censorship or gaining access to geographically blocked content offer privacy and security. It is hard to endorse that a VPN is not malicious. However, a significant number of free VPNs have been identified as being malicious and using risky levels of permissions. In this article, we deliver an accurate approach that identifies malicious Android VPNs with a possible malware presence. We have provided a dataset of 1300 Android VPNs with 184 specific permissions which might be questioned during installation and identified the malware in our dataset using the report from VirusTotal and the recognition of numerous reputable antivirus vendors [3]. Furthermore, we have presented a Convolutional Neural Network (CNN) optimized for identifying malware VPNs using the proposed dataset. The experimental results demonstrate that our proposed method outperforms other well-known machine learning classifiers in terms of accuracy, precision, and other evaluation metrics. In addition, the proposed approach is feasible, easy to use and due to its robust and well-optimized neural network, performs fast. The classification with reasonable computational resources is performed quickly since only the permissions that VPN requests are taken into consideration.

The proposed approach can be used to identify and detect Android malicious VPNs accurately on the target device before installation. The specific contribution of this paper is as follows:

- A novel dataset of android VPNs with their permissions and their risks is available in [3].
- A CNN classifier that can be used to detect the state of the security of VPN clients.

This paper focuses on the detection of Android VPNs contaminated by malware, which may put user devices at risk. The entire paper has been classified as follows; Section 2 discusses the related work of identifying Android malwares, section 3 illustrates the proposed dataset, section 4 explains the introduced classifier, section 5 describes and elaborates the experimental evaluation, and section 6 concludes the paper.

2. Background

So far, various techniques have been introduced in the literature for Android malware identification in this section. Three methods have been proposed to identify malware on Android devices, which are static, dynamic, and hybrid. An in-depth analysis of 283 Android VPN clients was carried out by Ikram et al. in [1] from a population of 1.4M Google Play apps. They characterised the behaviour of VPN apps and their impact on users' privacy and security. According to the main findings of their analysis, 82% ask for permission to access sensitive resources like user accounts and text messages while using third-party tracking libraries, while over 38% of them contain some malware presence according to VirusTotal [4]. Nevertheless,

25% of the analysed VPN apps receive at least a 4-star rating, and 37% have more than 500K installs. Their research showed that only a few VPN users have publicly expressed any privacy or security concerns in their app reviews. According to their study, a user's IP address will be leaked by 84% of Android VPN apps, sensitive data will be attempted to be accessed by 82%, third-party tracking is used by 75%, there is malware in 38% of apps, and 18% of apps don't even encrypt data, leaving the user completely exposed.

An active measurement system was developed by Taha Khan and his team [5] to test various infrastructure-related and privacy-related VPN service features and assess 62 commercial providers. According to their research results, although paid VPN services appear to be less likely than free ones to intercept or tamper with user traffic, many VPNs do inadvertently leak user traffic through several different channels. They also discovered that a sizeable portion of VPN providers transparently proxy traffic, and many of them misrepresent the actual location of their vantage points; 5 to 30% of the vantage points associated with 10% of the providers they studied were hosted on servers in nations other than those that were advertised to users. Wilson et al. [6] examined several selected iOS VPN apps from the Apple App Store to determine the level of security and privacy provided by VPN providers. Installing VPN software on a target device, simulating network traffic for a set amount of time, and capturing the traffic was all part of their testing methodology. According to their research, with many applications still using HTTP and not HTTPS for transmitting specific data, there were several tested VPN applications that had common security issues. A large majority of the VPN applications failed to route additional user data through the VPN tunnel. Moreover, it was discovered that only fifteen of the tested applications had correctly implemented the tunnelling protocol that was most highly recommended for user security. Additionally, they provided a set of recommendations for best practices that will help iOS developers create safe and secure VPN clients.

Wangchuk and Rathod [7] revealed that most Android-based VPNs have intrusive permissions which can be dangerous for the users, while some of the VPNs were flagged for the presence of possible malware content. Furthermore, some free VPNs even failed the DNS leakage and traffic encryption tests. They analysed the permissions and the malware content of 229 Android VPN apps and tried to study the possibility of finding the forensic artefacts which could be left by the VPNs on the devices. Korty et al. [8] discussed the difficulties Indiana University (IU) faced in balancing the two risk factors to ensure the continuation of its mission throughout the COVID-19 pandemic. IU had to make pressing decisions as the pandemic struck and teaching went online worldwide that weighed cybersecurity against other factors such as usability, cost, and health and safety. Using VPNs for all activities would have been unsustainable at IU with 130,000 faculty members, employees, and students would be able to teach, learn, conduct research, and work from home. The network staff of IU employed a VPN feature called split tunnelling to decrease the load, while the pros and cons of their methodology were discussed.

While the exact function of the fake VPN app is still unknown, SideWinder has published rogue apps under the pretext of utility software. This time, a phishing link downloads a VPN application called Secure VPN ("com.securedata.vpn"') from the official Google Play store in an attempt to impersonate the genuine Secure VPN app ("com.securevpn.securevpn") [9].

2.1 Static methods

Malware analysis can be performed using the signature, permissions, Dalvik bytecode analysis, or a combination of these methods [10, 11]. In a static method, an application is examined for malicious behaviour without being executed. Using reverse engineering tools such as Apktool, dex2jar, etc. We can disassemble a mobile app into its source code and extract the features to generate patterns or signatures. Thus, the features that have been extracted are compared to signatures that have previously been identified as malware. The signature-based analysis uses either cryptographic or similarity measurement algorithms. Nevertheless, they suffer from code obfuscation. SDHash [12] measures the level of similarity between two files; it is typically used with image and video files. Permission-based approaches are simple for implementation. However, Dalvik bytecode-based analysis uses power, resources, and storage space while failing to execute native code. Also, zero-day malwares cannot be reliably detected using this method, especially when the malware signature database is limited. Furthermore, to overcome limitations and restrictions and to classify applications, the YARA project provides a public repository with malware signatures by a community of people. [13]. Wang et al. [14], Talha et al. [15,] and Li et al. [16] each took a different approach to identify malwares through extracting high-risk permissions from the manifest file and based on the permissions, classified the apps as benign or malicious. Milosevic et al. [17] presented an Android antimalware based on permissions and source code analysis using machine learning techniques. Kang et al. [18] introduced an n-gram opcode feature extraction-based machine learning malware detector system. Another recent work in [19] presented a malware detection methodology based on linear regression by applying Android permissions to an application. MLDroid [20] is a recently introduced Android malware detection framework using app ratings, API calls, and machine learning (ML) techniques.

Most of the static methods are unable to analyse encrypted, obfuscated, or unknown malwares which results in false positives. However, they are fast and secure with low resource consumption. It is a good idea to use static analysis along with other security models for better efficiency. Recently, HamDroid [21] presented a very efficient way to detect malicious anti-malwares. According to HamDroid authors' methodology, for some Android apps with specific purposes like anti-malware, very accurate results are achieved through permission analysis together with machine learning. Unlike older methods which deal with all types of apps the same way, HamDroid is only capable of analysing Android anti-malware apps that can detect malicious anti-malwares in a short time before installation on the target device.

2.2 Dynamic methods

Unlike static methods, which analyse the apps before installation on Android platforms, dynamic approaches demand monitoring of all system calls and resources during the execution of the apps on a test platform, such as network traffic, CPU and battery usage, and several active processes. Dynamic methods can deal with obfuscated and encrypted malware due to their runtime behaviour and interactions with the system. Although dynamic methods usually detect both known and unknown malwares more accurately, due to the restricted reachability of the code, they are slow, resource-consuming, and vulnerable. Thus, they could occasionally be unsafe or even dangerous and require certain expertise and a considerable amount of time yet suffer from runtime detection methods [22]. Although it can be challenging to spot a malicious outgoing flow, Taintdroid. [23] monitored real-time data accesses and labelled sensitive data to track data leakage. Particularly, when a legitimate

application invokes too many system calls, monitoring the system calls may lead to a false alarm while requiring a lot of resources.

2.3 Hybrid methods

Hybrid techniques benefit from both static and dynamic approaches together for better accuracy. They first analyse an application by a static method. Then use dynamic analysis to overcome both static and dynamic limitations [21]. In general, hybrid methods obtain the best results most of the time. Nevertheless, they are very resource and time-consuming due to their complexity. EspyDroid [24] was a hybrid method that precisely reflected the analysis of Android apps and overcame the drawbacks of static methods and runtime-dependent parameters.

2.4 Drawbacks of current methods

Various methods have been proposed for Android malware detection so far. However, they approach all applications in the same manner, including Android VPNs. As far as we know, no efficient methodology has been proposed yet, particularly for identifying Android VPNs with malware presence that may threaten Android users by gaining their trust. This research aims to present a quick and robust technique called MVDroid to detect Android VPNs containing malware before installation on users' devices. With regards to the high risk that users may face after the installation of VPNs, MVDroid utilises a machine learning technique as well as an updated dataset to combat malware VPNs. MVDroid can provide a robust, reliable, and secure approach able to identify malicious VPNs on Android devices with a permission-based analysis. The dataset contains over 1300 Android VPN records, including permissions and the risk of being infected by malware for each VPN record. The risk of each record in the dataset has been evaluated by a VirusTotal scan of that VPN record. MVDroid uses an optimised CNN.

While VirusTotal is a content aggregator the main difference is that it is not an intelligent detection method such the proposed solution which can identify malicious VPNs that employ similar permissions and thus, detect those in real time and not as VirusTotal does it by connecting to other antivirus engines to scan an app.

3. Dataset

This section demonstrates our proposed method for identifying, detecting, and characterising Android VPN apps on Google Play and any other unofficial websites that distribute Android apps. In the literature it has been shown that it is particularly important to be able to identify the family that a malware belongs. Recent research has shown that to be able to act against a malicious app it is necessary to identify it, and this happens with very high accuracy [47, 48]. Considering that it is an active area of research but there are cases that malware cannot be classified to a family correctly there is recent research that builds on top of that, and datasets have been developed for individual malware families. For example, there are recent works in the literature that have collected data and developed methodologies to detect individual types of malwares such as anti-malware, malware that pretend to remove malware but do not, Trojans and botnets [49, 50, 51, 52]. Regular works can identify if an app is malicious or not with very high accuracy as well but are limited in allowing family detection and further research to be easily built on top of that [53, 54].

Furthermore, in industry there has been recent active research on malicious VPNs by Kaspersky, NordVPN and researchers in the ESET research group [55, 56, 57]. This shows that

there is an increasing interest in detecting such apps since these are popular among users who want to change their IP address and since there are many such apps available regular users are more susceptible in installing one. In view of the above facts, a novel dataset for malicious VPN detection will help the research community to identify the properties of such malicious apps and will allow further research to be developed, thus improving malware detection approaches.

Moreover, searching and downloading VPN apps on Google Play and other websites to develop a dataset is not a trivial task. Considering that a given app's profile is available on Google Play and the list of permissions requested for the specific app are available, in the permission live is not necessarily included the use of BIND_VPN_SERVICE permission by the app. A given Android app contains an AndroidManifest file including all the permissions required by the app in relation to the app, service, or specific activity. A service performs long-running operations in the background, while activities are app elements that run in the foreground on a single screen and require user interaction. The BIND_VPN_SERVICE permission will not appear in the list of Android permissions available on Google Play when it is declared by an app developer within the "service" tag. Hence, we have crawled Google Play to download each apk file and then decompiled them to inspect their AndroidManifest files in detail. This way, we make sure that we correctly identify VPN apps at scale. We used Google Play's search feature with keywords like "VPN", "anonymity", "privacy", "censorship", "security" or "virtual private network" to find apps containing that keyword in their description. We applied a 'breadth-first-search' method for any other app considered by Google Play as "similar" as well as for other apps created and published by the same developer.

We collected a total of over 1300 Android apk files of VPN apps, mostly from Google Play and other websites during a one-month period in November 2021. We extracted all features from apk files, including internet access and other required app permissions, and analysed all the files using VirusTotal [4], which has received reports from more than 70 well-known antivirus detection engines. We classified the apk files into two categories; legitimate VPN apps and malicious VPN apps that pose as legitimate apps but are harmful to the target device and we have labelled these as 0 (legitimate/begin) and 1 (malicious) in the dataset. Assessments showed that 9.3% of collected VPNs had any type of malware, while the other portion was safe. Moreover, a VPN might ask for 184 specific permissions at most on an Android platform. To make the dataset accessible to a wide range of software applications, we put all the data in a file in the.csv format. There are 185 columns in the proposed dataset, and 184 of them are specific app permissions plus the risk flag indicating the presence of malware. The first row of the dataset is permission titles, and the rest of the entries are 1300 apk files, which are Android VPN apps. All columns have binary values, so when a VPN requests specific permission, the corresponding column of the dataset entry has a value of "1," and unnecessary permissions requested by an app have a value of "0." Typically, a user can assume that the risk flag of a VPN app, which has been recognised as malware by most of the antimalware engines used by VirusTotal and presented in a final report after the scan of an apk file which in the dataset has a value of '1', meaning that it is malicious. Similarly, risk flags for other safe VPN apps are equal to '0'. However, in the literature this is not the case since the work in [58] sets the minimum number as 1, in [59] as 2, in [60] as 4 and in [61] as 10 or more, thus since there isn't a standardized approach to follow in our experiments we decided to set the minimum threshold as 1 but all statistical information obtained from VirusTotal is included in the dataset for this value to be changed, The complete version of the

dataset is available and publicly accessible on [3] for other researchers. For the sake of clarity, Figure 1 indicates a small portion of the dataset. Our proposed dataset is reliable since the classifications have been made through the VirusTotal mechanism, a popular web-based antimalware scanning tool which incorporates several reputed antivirus engines to identify malicious patterns. In the dataset we scanned each app using VirusTotal and if at least 1 antivirus engine states that it is malicious we set the respective entry in the dataset to 1 (malicious) and only if all antivirus engines do not find it as malicious, we set it as 0 (non-malicious). If the baseline changes this could possibly affect the overall accuracy of the algorithm and possibly a malicious app can be classified as non-malicious.

The motivation behind permissions is that an app can request many permissions from a user before it is installed and the user in many cases won't be bothered what each one is, thus making it easier to install a malicious app. By creating a dataset based on all permissions we show that malicious VPNs can be detected with very high accuracy when there is ground truth that shows which permissions can lead to malicious activity. However, it should be also noted that a specific permission can be unnecessary, but not malicious but similar malicious apps have similar permissions that request to use which can be identified using machine learning algorithms. Although, it is not always possible to detect 100% all malicious apps due to this fact and that's why machine learning-based detection methods are affected by false negatives (FNs) or False Positives (FPs).

The permissions are identified after we scan each APK file in VirusTotal. All possible combinations of permissions are based on a maximum number of 184 permissions, but each app is using a subset of those. The permissions requested from any of these 184 permissions by each app are represented with 1 and if not requested then are represented by 0.

Moreover, VirusTotal, owned by a Google subsidiary, is an excellent choice to collect such data since it is controlled by experts in the area, and it aggregates results from many antivirus products and online web engines which makes it an effective choice of detecting malicious activities. VirusTotal does not use any form of Intelligent algorithms, thus it is not able to identify zero-day vulnerabilities. VirusTotal connects to over 70 antivirus applications and submits a report to the user which shows how many of the antivirus applications identifies an app as malicious [4].

1	Name	MD5	INTERNET	READ PHONE STATE	WRITE EXTERNAL STRORAGE	SYSTEM ALERT WINDOW	CHANGE WIFI STATE	GET TASKS	ACCESS FINE LOCATION	USES POLICY FORCE LOCK	WAKE UP STOKER	Risk score
2	orchid-vpn-secure-networking_0.9.10	3ceb065e489ba3d70627cf8617751cd0	1	0	0	0	0	0	0	0	0	0
3	free-vpn-bearvpn-fast-and-secure-vpn_1.9	2bf7b13a0f41d8bd6c488c7a04e0867d	1	1	0	0	0	0	0	0	0	0
4	smart-lock-vpn-proxy-master-the-best-shield_1.0.33	9814cae6ee3c1e43ac44d6712a1293bd	1	1	0	0	0	0	0	0	0	1
5	com.opera.vpn_v1.5.0-30_Android-4.0.3	a74f3579cd6a1944cb96165d245e39fa	1	0	0	0	0	0	0	0	0	0
6	avira-phantom-vpn-free-fast-vpn-client-proxy_3.6.2	f7f28dd4af9436d17963e0c5673b2051	1	1	1	0	0	1	1	0	0	0

Figure 1. Dataset sample

Details of the proposed Convolutional Neural Network (CNN), which we designed and trained from scratch as an optimised classifier for detecting malware VPNs, are given in the next section of the paper. The CNN has been trained and verified using our VPN dataset and simulation results have been presented in section 5. Section 5 also shows the classification results performed by other algorithms using the same dataset.

4. The proposed classifier

In previous works in the literature that are based on permissions it has been show that supervised machine learning algorithms, including neural networks can be very accurate classifiers [21, 49, 51, 53, 54]. Thus, we used a CNN neural network to detect malware VPNs in our dataset. A CNN is a good estimator in our case due to the immense flexibility of the math performed in the overall function. It is an entirely mathematical system that uses a lot of data to gradually approximate complex input-output relationships. According to our VPN dataset, we designed our neural network as illustrated in Figure 2.

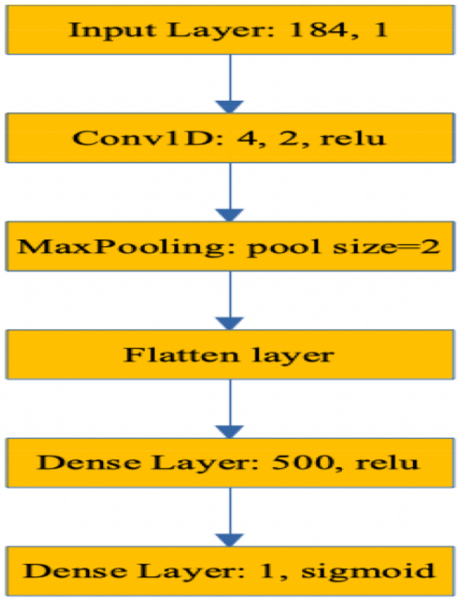


Figure 2. CNN architecture

The number of input nodes must match the number of permissions in the dataset, which is exactly 184. In addition, even with so many input nodes, only one output node is needed since the classification here is a yes/no decision-maker. Initially due to the high imbalance of the data we used the SMOTE library to oversample the minority class. The dataset contains 1179 non malicious and 121 malicious apps and after applying SMOTE the 121 malicious entries are oversampled to 1179 to balance the dataset. After using SMOTE to balance the dataset we dropped all columns that contain 85% or more 0s (i.e., we deleted all columns where the respective permission is not required by 85% or more of the entries). At the next step we identified the optimum settings to be a Convolutional 1D layer with a filter of 4,2 and by using the relu activation function, at the next step we identified that a MaxPooling layer with a size 2 provides the optimum result, then at the next step a flatten layer follows. In the next step, there is a dense layer with 500 perceptrons and then the final dense layer for the classification using sigmoid and Adam with a 0.01 learning rate which provided the most accurate results in this case, while other optimizers and learning rates were tested. Finally, the number of epochs was 20 and the batch size was set to 16 which provided the most accurate results. For hyperparameter tuning we used a grid search approach in combination with trial and error. The algorithm starts with the convolution which takes place as shown below in equation 1. Where y is the output, n is the length of the convolution represented by x and the kernel represented by h. S is the number of positions the kernel shifts.

$$y(n) = \begin{cases} \sum_{id=0}^{k} x(n+i)h(i), & if\ n = 0 \\ \sum_{i=0}^{k} x\big(n+i+(s-1)\big)h(i), & otherwise \end{cases} \quad (1)$$

The relu function used in the convolution layer is shown below in equation 2. Where for the output y and the input x a value is returned from 0 to infinite.

$$y\,(x) = \max(0, x) \quad (2)$$

At the next step the pooling layer takes place to reduce the dimensionality with a size of 2, which helps to reduce any possible overfitting. Then the flatten layer concatenates the output to a flat structure that can be used as an input to a fully connected Multi-Layer Perceptron as shown in equation 3 where Zm is the function output, f is the function name, followed by the function inputs, bias b, and the summary of the inputs.

$$Zm = f\,(x_n\,, w_{mn}) = b + \sum_m x_n\, w_{mn} \quad (3)$$

At the next step the output uses a one hot encoding where the output is either 0 or 1 for an input x based on the sigmoid function shown in equation 4.

$$\sigma(\mathrm{x}) = \frac{1}{1 + e^{-x}} \quad (4)$$

5. Experimental evaluation

The Python programming language was used as well as Keras and scikit-learn libraries to train and verify our neural network classifier using the proposed VPN dataset. For array operations and reading data from files, the NumPy and Pandas libraries are required. The simulation is divided into four phases: reading the VPN dataset, training the neural network with a portion of the dataset, defining the network's parameters, such as node numbers, learning rate, and so on; and finally, verifying the neural network with the rest of the dataset. The simulation software was a single-threaded Python application executed on an Intel Pentium core i5 @ 2.6 GHz with 4GB of RAM using the Microsoft Windows 10 operating system. In this paper, we use a 5-fold cross-validation procedure.

Validation is a true-or-false type of classification. We used well-known evaluation metrics such as Accuracy, Precision, Recall and F1, which are defined in equations 5 to 8 respectively, where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative. The Accuracy shows the overall performance, while the Precision describes what portion of predicted malicious VPNs are truly malware. The Recall metric is the portion of actual malware that is correctly classified and the F1 is a number between 0 and 1 and is the harmonic mean of Precision and Recall. Precision and Recall are important evaluation metrics in malware detection that are based on the confusion matrix values which identify False Positives and False Negatives.

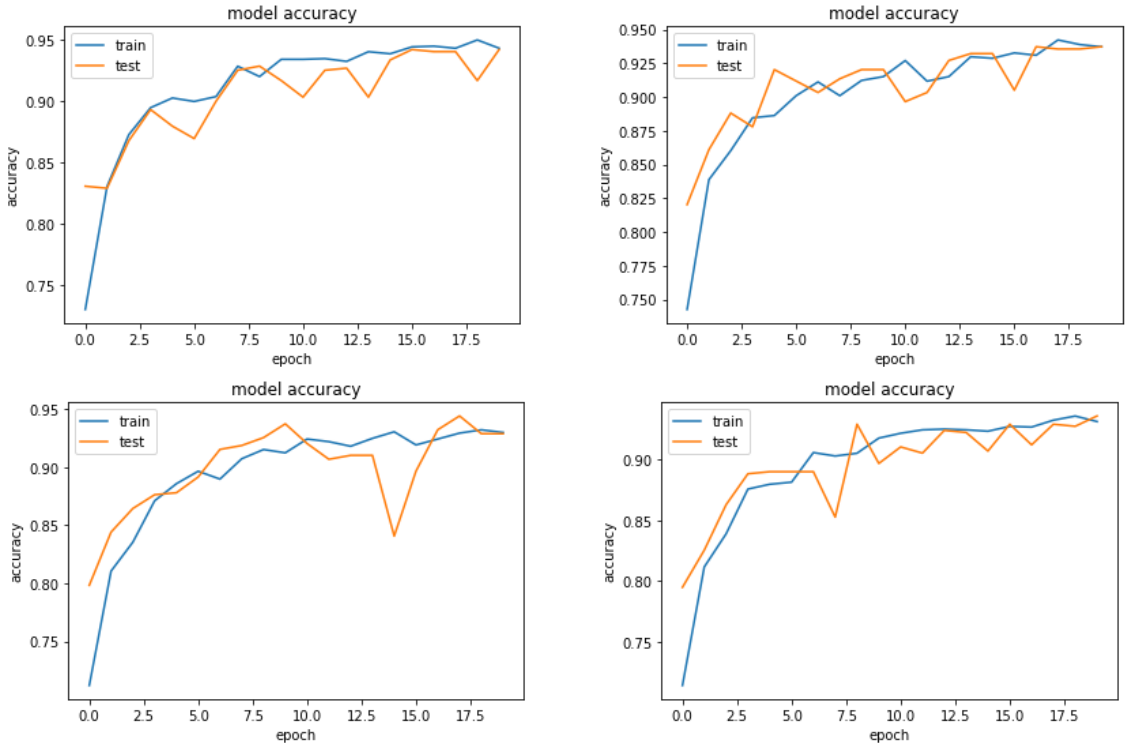
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (5)$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

The results indicate that MVDroid (the proposed classifier) can identify and detect malicious VPNs with high accuracy and outperforms all the other classifiers in the comparisons. More specifically, figures 3 and 4 present the training/test accuracy and the loss over the 5 epochs for each of the five folds. In figure 3 it is shown that as the number of epochs grows both the train and test accuracy grow until they stabilize which means that the classifier cannot become any better. While in figure 4 the loss is represented, and it is shown that as the number of epochs grows a larger error for the test accuracy is shown. The results in the figures show that the training accuracy grows stably while the testing accuracy is close which shows that the model performs well and towards the end the last epochs the accuracy remains similarly in figure 4 the training accuracy has a small loss compared to the testing accuracy in which case the loss grows as the number of epochs grow. Additionally, table 1 presents the accuracy, precision, recall, and f1 scores over 5 different cross-fold executions and table 2 presents a comparison of the proposed classifier with other well-known classifiers and the settings used for the MLP, Random Forest, Decision Tree and KNN, which is the default from the sci-kit learn library.



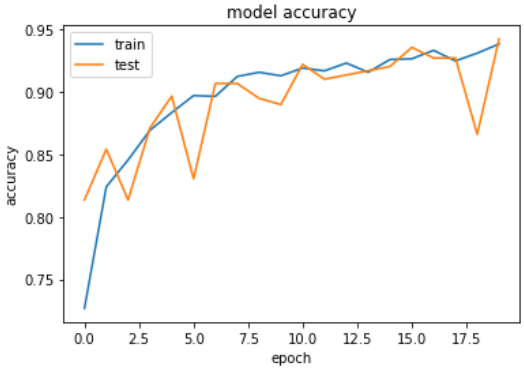


Figure 3. Train/Test accuracy over epochs

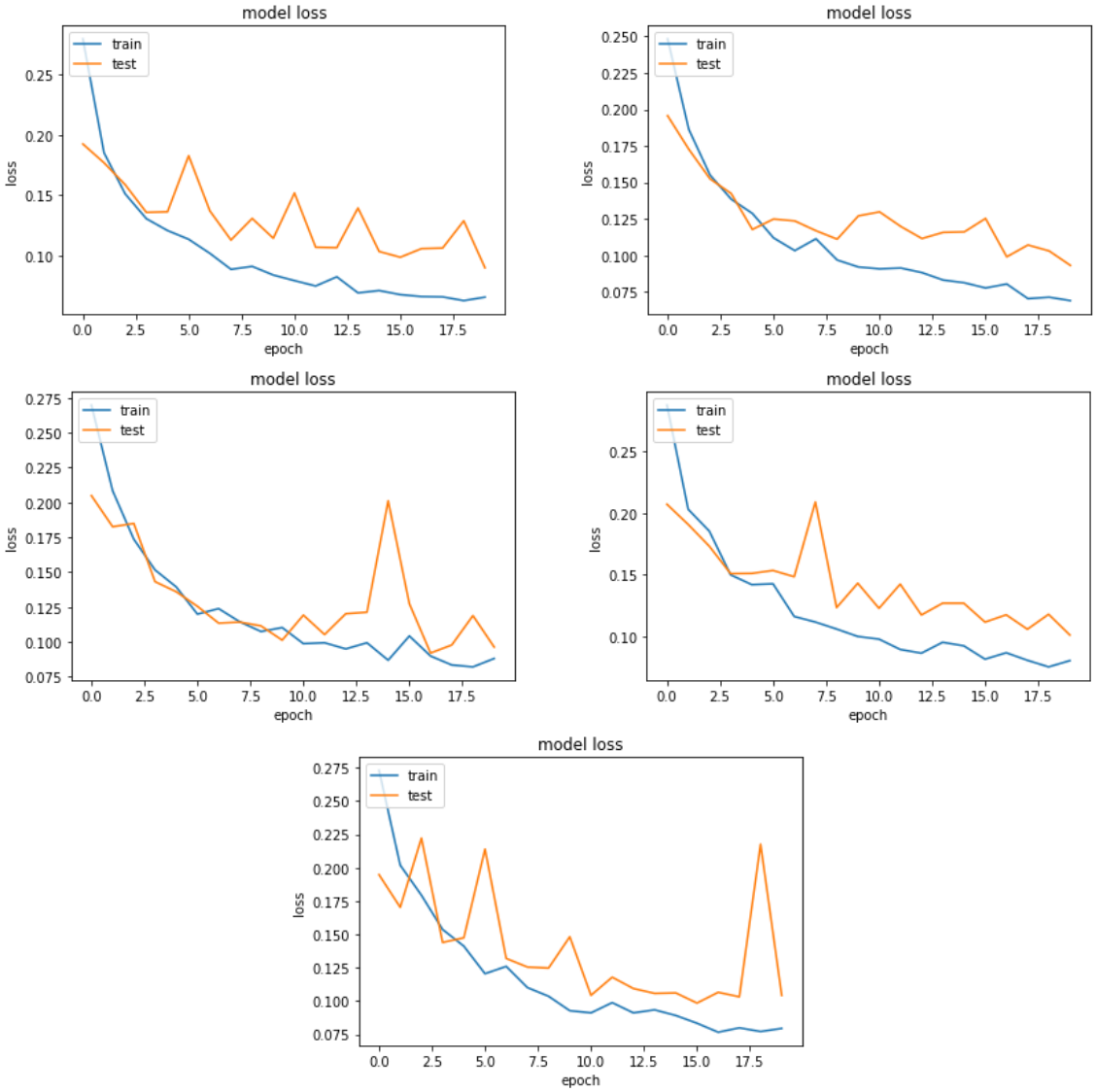


Figure 4. Loss over epochs

	Accuracy %	Precision %	Recall %	F1 %
Run 1	93.73	92.34	95.39	93.8
Run 2	92.88	91.55	94.58	93.0
Run 3	91.97	91.06	93.15	92.09
Run 4	92.68	92.74	92.81	92.77
Run 5	92.81	92.18	93.63	92.89
Average	92.81	91.97	93.91	92.91

Table 1. MVDroid evaluation results

	MVDroid %	MLP %	Random Forest %	Decision Tree %	KNN %
Accuracy %	92.81	89.1	88.1	89.1	91.1
Precision %	91.97	88.9	88.4	88.1	88.5
Recall %	93.91	89.8	88.8	87.0	88.9
F1 %	92.91	89.3	88.5	87.5	88.6

Table 2. Comparisons with other classifiers

The importance of identifying malware families has been demonstrated in the literature where there are several works in malware classification and in other areas of single malware family such as Trojans and Botnets. To this extent, the above results indicate the first step towards malicious VPN detection for Android. Malicious VPN detection has started gaining significant interest in the industry most importantly because typical users can easily install a VPN client through the app store. Using the proposed data and the classifier malicious VPN can be detected with high accuracy and will allow further research to take place in this area in the future by other research teams. On the other hand, a limitation is that due to the constraint of only including malicious VPNs the dataset is highly unbalanced, thus we used SMOTE to oversample the malicious entries to balance the data. More investigation in the future will allow to identify more malicious VPN clients and develop techniques to overcome this issue which will result in even higher detection accuracy.

6. Conclusions

In this paper, we discussed VPN malware identification techniques on the Android operating system, which are static, dynamic, and hybrid as explained above. We declared the importance of malicious VPNs threatening many Android users around the world. We hypothesised that a permission-based analysis can deliver accurate results for the categorization of Android VPNs and perform admirably in a reasonable amount of time. Thus, we collected many Android VPN apps from Google Play and other websites and provided a dataset including permissions and the risk of being malware for all collected VPNs through scanning by VirusTotal. Furthermore, we presented an optimised CNN neural network that can detect unsafe Android VPNs quickly before installation on the target device.

We trained and then verified our proposed classifier using our VPN dataset and compared the classification results with other well-known classifiers regarding the important evaluation metrics. As far as we know, there is no publication yet on malicious Android VPN detection. Unlike the previous Android malware detectors, which use complex or ensemble classifiers, MVDroid is straightforward, using an optimised neural network to produce very accurate results. MVDroid is practical because it is straightforward to extract permissions from the manifest file before the installation of VPNs and the classification results are ready by the neural network. In the future, we plan to collect a dataset of iOS VPNs and study how neural networks work regarding this problem area for the iOS platform.

Author Contributions: S.S. and N.P. Data collection. S.S. and S.K. Algorithm development. S.S., S.K. and M.P Wrote the main manuscript. S.S and N.P. Prepared the figures. S.S., S.K., M.P., and N.P Evaluation. N.P. Reviewed the manuscript.

Funding: This research received no external funding.

Data Availability The data have been made available by the authors through the Kaggle platform at https://www.kaggle.com/datasets/saeedseraj/mvdroid-a-malicious-android-vpn-detector-dataset

Declarations

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ikram M, Vallina-Rodriguez N, Seneviratne S, Kaafar M A, Paxson V (2016) An analysis of the privacy and security risks of android vpn permission-enabled apps. Proceedings of the 2016 Internet Measurement Conference. https://doi.org/10.1145/2987443.2987471
2. Khattak S, Javed M, Khayam S A, Uzmi Z A, Paxson V (2014) A Look at the Consequences of Internet Censorship Through an ISP Lens. Proceedings of the 2014 Conference on Internet Measurement Conference, Vancouver 271-284.
3. https://www.kaggle.com/datasets/saeedseraj/mvdroid-a-malicious-android-vpn-detector-dataset. Accessed 20 March 2022
4. VirusTotal. https://www.virustotal.com . Accessed 20 March 2022
5. Taha Khan M, DeBlasio J, Voelker G M, Snoeren A C, Kanich C, Rodriguez NV (2018) An Empirical Analysis of the Commercial VPN Ecosystem. In Proceedings of the Internet Measurement Conference 2018 (IMC '18). https://doi.org/10.1145/3278532.3278570
6. Wilson J, McLuskie D, Bayne E (2020) Investigation into the security and privacy of iOS VPN applications. In Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20). https://doi.org/10.1145/3407023.3407029
7. Wangchuk T, Rathod D (2021) forensic and behavior analysis of free android VPNs. Journal of Applied Engineering, Technology and Management 1(1):91-101. https://doi.org/10.54417/jaetm.v1i1.27
8. Korty A, Calarco D, Spencer M (2021) Balancing risk with virtual private networking during a pandemic, Business Horizons 64(6):757-761. https://doi.org/10.1016/j.bushor.2021.07.011
9. https://thehackernews.com/2022/06/sidewinder-hackers-use-fake-android-vpn.html?&web_view=true
10. Sihaga V, Vardhan M, Singh P (2021) A survey of android application and malware hardening. Computer Science Review 39. https://doi.org/10.1016/j.cosrev.2021.100365
11. Arshad S, Ali Shah M, Khan A, Ahmed M (2016) Android malware detection & protection: a survey. Int. J. Adv. Comput. Sci. Appl. 7(2): 466. https://doi.org/10.14569/IJACSA.2016.070262
12. Roussev V (2010) Data fingerprinting with similarity digests. IFIP Advances in information and communication technology 337:207-226. https://doi.org/10.1007/978-3-642-15506-2_15
13. YaraRules: yara-rules/rules ; https://github.com/Yara-Rules/rules. Accessed 28 March 2022
14. Wang W, Wang X, Feng D, Liu J, Han Z, Zhang X (2014) Exploring permission-induced risk in android applications for malicious application detection. IEEE Trans. Inform. Forensics Security 9(11):1869-1882. https://doi.org/10.1109/TIFS.2014.2353996
15. Talha K A, Alper DI, Aydin C (2015) APK auditor: permissionbased android malware detection system. Digital Investigation 13:1-14. https://doi.org/10.1016/j.diin.2015.01.001
16. Li J, Sun L, Yan Q, Li Z, Srisa-An W, Ye H (2018) Significant permission identification for machine-learning-based android malware detection. IEEE Trans. Indu. Inform. 14(7):3216-3225. https://doi.org/10.1109/TII.2017.2789219

17. Milosevic N, Dehghantanha A, Choo K K R (2017) Machine learning aided Android malware classification. Computers & Electrical Engineering, Elsevier 61:266-274
18. Kang B J, Yerima S Y, McLaughlin K, Sezer S (2016) N-opcode analysis for android malware classification and categorization. In Proceedings of IEEE International Conference on Cyber Security And Protection Of Digital Services (Cyber Security) 1-7
19. Sahin D O¨ , Kural O E, Akleylek S et al (2021) A novel permission-based Android malware detection system using feature selection based on linear regression. Neural ComputAppl 29:245-26
20. Mahindru A, Sangal AL (2021) MLDroid: framework for Android malware detection using machine learning techniques. Neural ComputAppl 33(10):5183-5240
21. Seraj S, Khodambashi S, Pavlidis M, Polatidis N (2022) HamDroid: permission-based harmful android anti-malware detection using neural networks. Neural Comput & Applic. https://doi.org/10.1007/s00521-02106755-4
22. Vidas T, Christin N (2014) Evading android runtime analysis via sandbox detection. In Proceedings of the 9th ACM symposium on Information, computer and communications security 447-458. https://doi.org/10.1145/2590296.2590325
23. Enck W, Gilbert P, Han S, Tendulkar V, Chun B G, Cox L P, Jung J, McDaniel P, Sheth A N (2014) TaintDroid. ACMTrans. Comput. Syst. 32(2):1-29. https://doi.org/10.1145/2619091
24. Gajrani J, Agarwal U, Laxmi V, Bezawada B, Gaur M S, Tripathi M, Zemmari A (2020) EspyDroid+: Precise reflection analysis of android apps. Computers & Security, 90:101688
25. Zhou W, Zhou Y, Jiang X, Ning P (2012) Detecting repackaged smartphone applications in third-party android marketplaces, In Second ACM Conference on Data and Application Security and Privacy 317-326. http://dx.doi.org/10.1145/2133601.2133640.
26. Faruki P, Ganmoor V, Laxmi V, Gaur M S, Bharmal A (2013) AndroSimilar: robust signature for detecting variants of android malware. In Proceedings of the 6th International Conference on Security of Information and Networks 152-159. https://doi.org/10.1145/2523514.2523539
27. Kim J, Yoon Y, Yi K, Shin J, Center S (2019) ScanDal: Static analyzer for detecting privacy leaks in android applications. In MoST 12.
28. Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y (2012) Andromaly: a behavioral malware detection framework for android devices. J. Intell. Inform. Syst. 38(1):161-190. https://doi.org/10.1007/s10844-010-0148-x
29. Burguera I, Zurutuza U, Nadjm-Tehrani S (2011) Crowdroid: behaviorbased malware detection system for Android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices 15. https://doi.org/10.1145/ 2046614.2046619
30. Yan L K, Yin H (2012) DroidScope: seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In Proceedings of the 21st USENIX conference on Security symposium 1-16
31. Lindorfer M, Neugschwandtner M, Weichselbaum L, Fratantonio Y, Veen V V D, Platzer C (2016) ANDRUBIS – 1,000,000 Apps Later: A View on Current Android Malware Behaviors. In IEEE Proceedings-3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security 3-17. https://doi.org/10.1109/ BADGERS.2014.7
32. Zheng M, Sun M, Lui J C S (2013) DroidAnalytics : A Signature Based Analytic System to Collect , Extract , Analyze and Associate Android Malware. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications 163-171. https://doi.org/10.1109/TrustCom.2013.25
33. Sato R, Chiba D, Goto S (2013) Detecting Android Malware by Analyzing Manifest Files 36:23-31
34. SanzB , Santos I, Laorden C, Ugarte-Pedrero X, Bringas P G, Alvarez G (2013) PUMA: Permission usage to detect malware in android. Adv. Intell. Syst. Comput. 189: 289-298
35. Wognsen E R, Karlsen H S, Olesen M C, Hansen R R (2014) Formalisation and analysis of Dalvik bytecode. Sci. Comput. Program. 92:25-55
36. Aafer Y, Du W, Yin H (2013) DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. Secur. Priv. Commun. Networks 127:86-103
37. Chen T, Mao Q, Yang Y, Lv M, Zhu J (2018) TinyDroid: A lightweight and efficient model for Android malware detection and classification, Mob. Inf. Syst. http://dx.doi.org/10.1155/2018/4157156.
38. Chen J, Alalfi M H, Dean T R, Zou Y (2015) Detecting Android malware using clone detection. J. Comput. Sci. Technol. 30(5):942-956. http://dx.doi.org/10.1007/s11390-015-1573-7.

39. Liu P, Wang W, Luo X, Wang H, Liu C (2020) NSDroid: Efficient multiclassification of android malware using neighborhood signature in local function call graphs. Int. J. Inf. Secur. http://dx.doi.org/10.1007/s10207020-00489-5.
40. Suarez-Tangil G, Dash S K, Ahmadi M, Kinder J, Giacinto G, Cavallaro L (2017) Droidsieve: Fast and accurate classification of obfuscated Android malware. In proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy 309-320. http://dx.doi.org/10.1145/3029806.3029825
41. Qiao M, Sung A H, Liu Q (2016) Merging permission and API features for Android malware detection. In 5th IIAI International Congress on Advanced Applied Informatics 566-571. AAI.2016.237
42. Wu D, Mao C, Wei T, Lee H, Wu K (2012) DroidMat: Android malware detection through manifest and API calls tracing. In 7th Asia Joint Conference on Information Security 62-69. http://dx.doi.org/10.1109/AsiaJCIS.2012.18
43. Arp D, Spreitzenbarth M, Hu¨bner M, Gascon H, Rieck K (2014) DREBIN: Effective and explainable detection of Android malware in your pocket. In 21st Annual Network and Distributed System Security Symposium. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.676.2139 Accessed 20 March 2022
44. Feng P, Ma J, Sun C, Xu X, Ma Y (2018) A novel dynamic Android malware detection system with ensemble learning. IEEE Access 6:3099631011
45. Xie N, Zeng F, Qin X, Zhang Y, Zhou M, Lv C (2018) RepassDroid: Automatic detection of Android malware based on essential permissions and semantic features of sensitive APIs. IEEE International Symposium on Theoretical Aspects of Software Engineering 52-59
46. Bugiel S, Davi L, Dmitrienko A, Fischer T, Sadeghi A R (2011) XManDRoid: A New Android Evolution to Mitigate Privilege Escalation Attacks. Tech. Rep. TR-2011-04, TechnischeUniversit¨at Darmstadt
47. Mahdavifar, S., Alhadidi, D., & Ghorbani, A. A. (2022). Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder. *Journal of network and systems management*, *30*, 1-34.
48. D'Angelo, G., Palmieri, F., & Robustelli, A. (2022). A federated approach to Android malware classification through Perm-Maps. *Cluster Computing*, 1-14.
49. Seraj, S., Pavlidis, M., & Polatidis, N. (2022, June). TrojanDroid: Android Malware Detection for Trojan Discovery Using Convolutional Neural Networks. In *Engineering Applications of Neural Networks: 23rd International Conference, EAAAI/EANN 2022, Chersonissos, Crete, Greece, June 17–20, 2022, Proceedings* (pp. 203-212). Cham: Springer International Publishing.
50. Ullah, S., Ahmad, T., Buriro, A., Zara, N., & Saha, S. (2022). TrojanDetector: A Multi-Layer Hybrid Approach for Trojan Detection in Android Applications. *Applied Sciences*, *12*(21), 10755.
51. Yerima, S. Y., Alzaylaee, M. K., & Shajan, A. (2021). Deep learning techniques for android botnet detection. *Electronics*, *10*(4), 519.
52. Moodi, M., Ghazvini, M., & Moodi, H. (2021). A hybrid intelligent approach to detect android botnet using smart self-adaptive learning-based PSO-SVM. *Knowledge-Based Systems*, *222*, 106988.
53. Amer, E. Permission-Based Approach for Android Malware Analysis Through Ensemble-Based Voting Model. In Proceedings of the 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 26-27 May 2021; pp. 135-139.
54. Wang, H.; Zhang,W.; He, H. You are what the permissions told me! Android malware detection based on hybrid tactics. J. Inf. Secur. Appl. 2022, 66, 103159.
55. Bahar, Z. (2022) *Your free VPN app could be a trojan: How to spot fake vpns*, *NordVPN*. Available at: https://nordvpn.com/blog/fake-vpn/ (Accessed: January 23, 2023).
56. Glover, C. (2022) *Sandstrike Fake VPN is latest in wave of new Android malware*, *Tech Monitor*. Available at: https://techmonitor.ai/technology/cybersecurity/android-malware-sandstrike-fake-vpn (Accessed: January 23, 2023).
57. Editor (2022) *Eset Research: Bahamut Group targets android users with fake VPN apps; spyware steals users' conversations*, *ESET*. Available at: https://www.eset.com/int/about/newsroom/press-releases/research/eset-research-bahamut-group-targets-android-users-with-fake-vpn-apps-spyware-steals-users-convers/ (Accessed: January 23, 2023).
58. Li, L., Li, D., Bissyandé, T. F., Klein, J., Le Traon, Y., Lo, D., & Cavallaro, L. (2017). Understanding android app piggybacking: A systematic study of malicious code grafting. *IEEE Transactions on Information Forensics and Security*, *12*(6), 1269-1284.

59. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. E. R. T. (2014, February). Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss* (Vol. 14, pp. 23-26).
60. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., & Cavallaro, L. (2019). {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)* (pp. 729-746).
61. Salem, A., Banescu, S., & Pretschner, A. (2021). Maat: Automatically analyzing virustotal for accurate labeling and effective malware detection. *ACM Transactions on Privacy and Security (TOPS)*, *24*(4), 1-35.