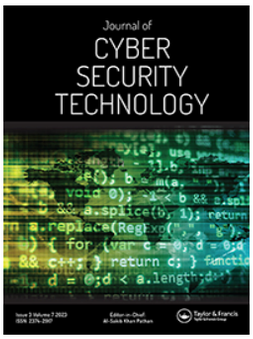

Journal of Cyber Security Technology

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tsec20

MadDroid: malicious adware detection in Android using deep learning

Saeed Seraj, Michalis Pavlidis, Marcello Trovati & Nikolaos Polatidis

To cite this article: Saeed Seraj, Michalis Pavlidis, Marcello Trovati & Nikolaos Polatidis (2023): MadDroid: malicious adware detection in Android using deep learning, Journal of Cyber Security Technology, DOI: 10.1080/23742917.2023.2247197

To link to this article: https://doi.org/10.1080/23742917.2023.2247197

Published online: 22 Aug 2023.

JOURNAL OF CYBER SECURITY TECHNOLOGY
https://doi.org/10.1080/23742917.2023.2247197

Taylor & Francis
Taylor & Francis Group

MadDroid: malicious adware detection in Android using deep learning

Saeed Seraj[a], Michalis Pavlidis[a], Marcello Trovati[b] and Nikolaos Polatidis[a]

[a]School of Architecture, Technology, and Engineering, University of Brighton, Brighton, UK;
[b]Department of Computer Science, Edge Hill University, Omrskirk, UK

ABSTRACT

The majority of Android smartphone apps are free. When an application is used, advertisements are displayed in order to generate revenue. Adware-related advertising fraud costs billions of dollars each year. Adware is a form of advertising-supported software, that turns into malware when it automatically installs additional malware and adware on an infected device, steals user data, and exposes other vulnerabilities. Better techniques for detecting adware are needed due to the evolution of increasingly sophisticated evasive malware, particularly adware. Even though significant work has been done in the area of malware detection, the adware family has received very little attention. This paper presents a deep learning-based scheme called MadDroid to detect malicious Android adware based on static features. Moreover, this paper delivers a novel dataset that consists of malicious Adware and benign applications and an optimised Convolutional neural network (CNN) for detecting Adware infected by malware based on the permissions of the applications. The results indicate an average classification rate that is higher than previous work for individual adware family classification in terms of well-known evaluation metrics.

ARTICLE HISTORY
Received 5 May 2023
Accepted 8 August 2023

KEYWORDS
Android; malware detection; adware; neural networks; new dataset

1. Introduction

The global smartphone market has experienced exceptional growth, thanks to the vast number of available applications and an open-source code platform that encourages app developers to create Android apps for free. These applications are considered essential to Android phones, as they drive innovation, leisure, accessibility, and compatibility with mobile devices. However, adware or mobile advertising in the form of banner ads, rich media, or interstitial ads is increasingly infiltrating Android

CONTACT Nikolaos Polatidis N.Polatidis@Brighton.ac.uk School of Architecture, Technology, and Engineering, University of Brighton, Brighton BN2 4GJ, UK

applications, posing a threat to users. The use of mobile devices has become widespread in recent years, and they are now essential to our daily lives, resulting in an exponential increase in mobile users and apps. Android, with over 2 billion users, is the most popular operating system worldwide and is thus a common target for malicious actors. Adware, a type of malware that displays unwanted and often offensive advertisements, is a prevalent threat to Android users [1] [2].

Advertising is a marketing strategy that promotes products, ideas, and services. With the emergence of the internet and smartphones, digital advertising has become increasingly popular. Digital ads are displayed while browsing websites or using mobile applications, and since many smartphones run on the Android platform, there is a vast ecosystem of Android apps. However, digital advertising is plagued by fraudulent activity, with adware being a prevalent security threat used to collect marketing data or display ads for profit. Adware is more common and effective than traditional malware and goes beyond the judicious advertising found in freeware or shareware. Adware is often installed alongside other programmes and can continue generating ads even when the user is not running the desired programme [3].

Smartphones have become essential tools in our daily lives, and the amount of sensitive information we store on them has made them prime targets for hackers. Malicious code can be installed on smartphones, allowing hackers to steal personal information and profit from advertising and micropayment systems. The number of mobile malware infections has grown exponentially, and Android devices are particularly vulnerable due to the openness of the Android market and their high market share. Malicious adware-based hacking attacks have become more intense and diverse over time, with the most common type infiltrating and controlling users' Android devices. Malicious adware has infected almost all current Android versions, making many Android devices worldwide vulnerable to threats. Advertisements in some free Android apps have become more aggressive, and users may not be aware of the changes on their devices. Even popular apps may contain adware, and users may not object to it or be aware of its effects [4].

Machine learning is an advanced technique used in cybersecurity to identify and classify malware, including adware, based on patterns and behaviours rather than just matching signatures. Machine learning algorithms are trained on large datasets of known malware and can detect new and previously unknown malware. These algorithms can learn to recognise and classify adware based on its behaviour, such as its use of network connections, data transmission, and system modification, among other things. By detecting adware through behaviour analysis, machine learning can identify new strains of adware that traditional signature-

based methods might miss. Machine learning can also help security experts quickly develop new rules and signatures for identifying adware and other malware, allowing for faster and more effective responses to new threats. Machine learning is expected to play an increasingly important role in cybersecurity as new and more complex forms of malware continue to emerge [5].

Therefore, a different approach is proposed to detect and classify adware-based permission analysis using deep learning. This study describes MadDroid, a permission-based adware detection algorithm that uses a Convolutional neural network (CNN). This research analyses the use of machine learning as a possible defence against mobile adware. We classify Android apps based on the features obtained from static analysis. The static features, which are permissions, are obtained from the VirusTotal Scanner website [6]. The complete process is fully explained in Section 4. To detect Android adware using permissions, we first created a new self-made dataset as explained in Section 5 and then utilised a tuned CNN algorithm. We employ a deep-learning technique to analyse Android adware and benign apps, based on the dataset that we have collected. Specifically, we consider experiments involving neural networks. This study compared several ML algorithms, namely, Decision Tree (DT), Random Forest (RF), K-Nearest Neighbour (K-NN), Support Vector Machine (SVM), Naive Bayes (NB), Multilayer Perceptron (MLP), Logistic Regression (LR), and Linear Discriminant Analysis (LDA).

The following contributions are delivered:

- A novel dataset is introduced 440 permissions to detect malicious adware on android devices.
- We propose a tuned convolutional neural network (CNN) to detect malicious android adware.
- We evaluate our approach using well-known evaluation metrics, with the results demonstrating that malicious adware can be detected with a very high degree of accuracy.

The rest of the paper is organised as follows: Section 2 is the research background; Section 3 is the related work; Section 4 illustrates the dataset creation process; Section 5 describes the dataset; Section 6 explains the proposed method; Section 7 delivers the experimental evaluation; Section 9 indicates the results; and Section 9 contains the conclusions and future work.

2. Background

2.1. Overview of adware

Adware is often distributed alongside other software, especially free software, or software that users may be tempted to download. Once installed on a user's device, the adware can track their online activities and display targeted advertisements, which can be annoying or even harmful. Adware can also slow down a device's performance and cause it to crash. To avoid adware, users should be cautious when downloading software and only download from trusted sources. They should also keep their devices and software up to date with the latest security patches and use reputable antivirus software. Additionally, users can install ad-blocking software or browser extensions to block unwanted advertisements [7].

Adware is installed on a user's device through a security flaw in an existing application. Users can unknowingly download it as well. This can occur when users perform one of the following actions:

- Download an adware-infected application.
- Use software that contains flaws that adware authors can exploit.

The aim of adware is to entice users to click on or engage with the advertisements displayed or downloaded by the software. Adware authors

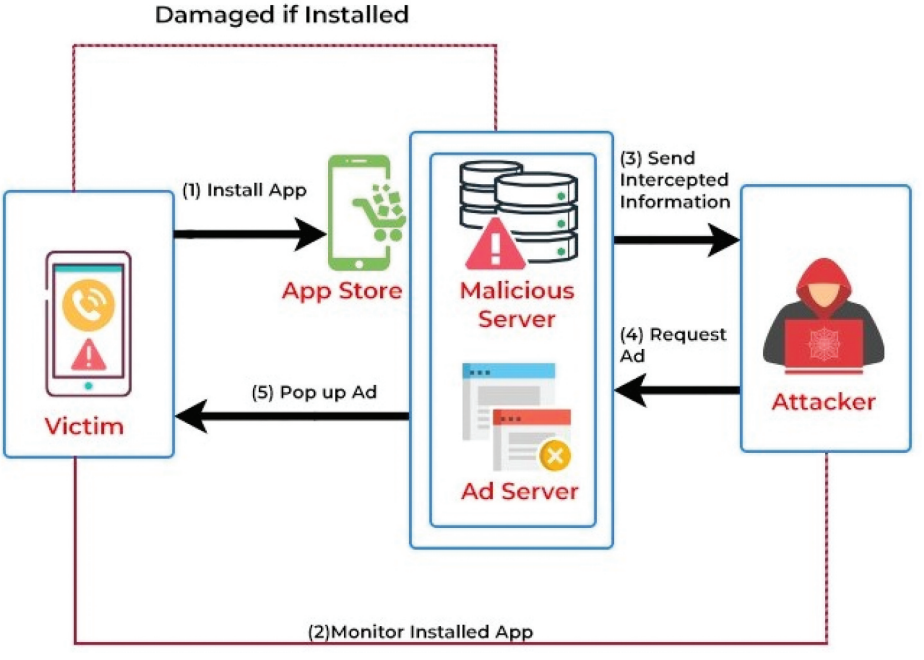

Figure 1. Demonstration of how malicious adware works [8].

and distributors earn money when users click on the online ads served by their adware. Although some adware is legitimate and agreed to by users, it is often unwanted. Adware can be a nuisance, but it may also carry harmful risks. Figure 1 demonstrates how malicious adware works.

2.2. Adware and history

Adware is any software programme, whether malicious or not, that can display advertisements on a personal computer. Malicious programmes that display misleading advertisements, blinking pop-up windows, massive digital billboards, and full-screen auto-play advertisements within an internet browser are the most common examples. The term is a compound of the words 'advertising' and 'software'. The developer earns money every time someone clicks on an advertisement displayed by adware. Some types of adware may interfere with your web browsing experience by directing you to malicious websites. Then, without your knowledge, some collect your browsing information and use it to serve you advertisements that are more tailored to your preferences and thus more likely to be clicked on.

When adware first became popular in 1995, many industry professionals assumed it was all spyware, which is software that allows someone to obtain covert data from a computer without the user's knowledge. Adware was demoted to the status of a 'potentially undesirable application', or PUA, as its credibility grew. As a result, despite its widespread use, little was done to ensure its legality. Adware makers did not begin monitoring and blocking problematic behaviour until the peak adware years of 2005–2008.

2.3. Adware and illegal use

Many people mix up adware and malware, malicious software designed to harm a computer or server. Malware includes viruses, spyware, worms, and some types of adware. Pop-up ads, inaccessible panels, and other types of malicious adware can infect computers. Once dangerous adware has infiltrated a computer, it may perform a variety of malicious activities, such as tracking the user's location, query history, and web browser viewing history, which the malware programmer can then monetize by selling to third parties.

2.4. Adware and the dark side

Adware is a cyber-security term that refers to adware applications that exhibit dangerous or irregular behaviour. Adware is classified as spyware

when it tracks users' activities without permission. Fraudsters take advantage of flaws in the validation process of ad networks or flaws in a consumer's browser. When a user visits an infected website, malicious adware can spawn pop-ups, pop-unders, and persistent windows that allow for drive-by installations. Visitors who disable ad blockers may be at risk of infection. Adware applications have been discovered that prevent antivirus software from running. Because some adware software is legal or does not have uninstallation processes, security software may be unable to identify which adware applications are truly dangerous.

2.5. Types of adware

Adware is most commonly associated with annoying pop-up windows and advertisements, but it can also take other forms. It is critical to distinguish between harmless and dangerous adware. The following are the most common types of adware:

2.5.1. Legitimate fc

Adware of this type allows you to subscribe to advertisements and software promotions, allowing developers to distribute their programmes for free by offsetting their expenses. Users instal this type of adware on purpose to obtain a free item. You can also choose to allow it to collect marketing data. All programmers, including reputable ones, create legitimate adware because providing clients with a free product is a legitimate and fair method of gaining adoption. However, not all software downloads are agreed upon by both the distributor and the user. In this situation, the line between legal and illegal blurs.

2.5.2. Potentially unwanted applications (PUAs)

PUAs are unwanted software packages that are bundled with legitimate complementary software applications. PUPs, or potentially unwanted programmes, are another name for them. Although not all PUAs are malicious, some may exhibit intrusive behaviours such as displaying pop-up advertisements or slowing down your device. It can slow down a computer's performance and potentially introduce security issues such as spyware and other unwanted software.

2.5.3. Legal abusive adware PUA

PUAs, both legal and abusive, are designed to bombard you with advertisements. Excessive advertising can be found in packaged software, internet browser toolbars, and other places. This is also legal because no malware is involved. Ads for fitness pills, for example, are common in adware like this.

2.5.4. Legal deceptive adware PUA

This category includes legal adware that deceives the user in some way. This type of PUA may make it difficult to uninstall secure third-party software. This strategy is occasionally used by legal adware, and it is lawful if the developer did not put malware-infected advertising or software there on purpose. Unfortunately, certain adware can inadvertently infect devices with malware.

2.5.5. Illegal malicious adware PUA

This category includes malicious adware that is either illegal to use or distribute. The PUA earns money by distributing malicious programmes to machines such as spyware, viruses, and other malware. The malware could be hidden within the adware, the websites it promotes, or other software programmes. The authors and distributors are spreading this threat on purpose and may employ aggressive tactics [9].

According to [8], deceptive and abusive adware is designed to manipulate users into interacting with advertisements or to obtain consent through deceptive means, such as bombarding them with unwanted ads or making it difficult to uninstall unwanted software. While adware is not inherently malicious, it can create vulnerabilities that may be exploited by malicious software. Only adware that is specifically designed to deliver harmful software to the user is considered malicious. Some types of malicious adware include the following:

(1) Spyware: Adware can contain code that tracks and records a user's personal information and internet browsing habits. If this data is collected without the user's knowledge or consent and sold to third parties, it is considered spyware. Many privacy advocates are critical of these practices.
(2) Potentially unwanted programs (PUPs): Malicious adware or spyware can be bundled with free or shareware software downloaded from the internet. Users may unknowingly download adware from an infected website. Antimalware programs often flag adware as a potentially unwanted program, regardless of whether it is malicious or not.
(3) Man-in-the-middle (MitM) attacks: Adware can also be used in MitM attacks, where the attacker routes user traffic through the adware vendor's system, even over secure connections. The communicating parties believe they are exchanging information securely, but the attacker can collect and manipulate sensitive information during the conversation.

Table 1. Attacks/Threats and their impact on user's privacy.

Attacks/Threats	Adversary End	Developer End	User End	Android End
Malware	Unauthorized Data Collection	Code Level Security Issues	Victim	Data Exposure to other apps
Malicious Ads	Attack Generator (e.g. DDoS attack)	Victim	Victim	Lack of Security Control
Malicious Ad-Libraries	Attack Generator	Money making & Lack of Security Check	Victim	No Privilege Escalation
Permission Misuse	Developers accessed Permissions misuse	Ad-Libraries accessed Permissions misuse	Victim	No Privilege Escalation
MitM	Attack Generator	Targeted Applications	Victim	Lack of Security Control
Certificate Compromise	Targeted Sites via Valid certificate	Lack of Security Check	Victim	Lack of Security Control
Click-Fraud Attack	Victim	Make Money or Exhaust Adversary	Security Exploitation	Lack of Security Control

2.6. Types of attacks/threats

Some common advertisement attacks are described and explained in the sections that follow. The impact of threats and attacks on an adversary, developers, users, and platform ends is described in Table 1, which summarises their relationship.

Attack/Threats are described below

- ***Adversary End***: This refers to an attack that occurs due to the presence of malicious advertisement libraries or networks. These libraries or networks may be designed to serve ads that contain malware or to redirect users to phishing or other malicious websites. When users interact with these ads, they may inadvertently download malware or give away sensitive information.
- ***Developer End:*** The two directions in which developers launch attacks are against advertisers, to exhaust their budget and abuse power, and against users, to steal personal information or make money. The terms 'Adversary End' and 'Developer End' are interrelated because app developers and ad libraries collaborate with each other. They share permissions and are located in the same code piece with the same UID. These two categories are separated only for the purpose of better understanding their behaviour.
- ***User End:*** refers to the end-users of a system who may be vulnerable to attacks due to their lack of security awareness, knowledge, and implementation of existing defensive measures. This means that users may not be aware of potential security risks and how to protect themselves from them, such as using strong passwords, avoiding clicking on suspicious links, and

regularly updating their software. As a result, they become easy targets for attackers who can steal their personal information, and financial data, or use their devices for malicious purposes.

- ***Android End:*** The security control of the Android platform is a significant factor, which has some vulnerabilities, including the absence of privilege escalation, ad SDKs, and application code permissions for developers.

As described in Table 1 summarizes the relationship between them where the impact of attacks/threats on 'Adversary developers', users and platforms are described

- ***Malware:*** The term malvertising is used to describe online advertisements that distribute malware [9]. Malware can be transmitted to an Android device through malicious software or advertisements. Users may be directed to other pages where they can download additional software or malicious applications by clicking on these ads while using the app. The majority of Android malware is in the form of Trojans.
- ***Malicious Ads:*** It is related to malware injection in some ways. The separation from malware serves only to categorise malicious advertisements. Malware can, however, be injected through the code of developers or malicious libraries.
- ***Click-Fraud Attack***: Adware attacks can occur not only on Android devices but also on ad networks. These attacks aim to exploit security vulnerabilities in the Android platform and other systems. Click fraud is an example of a cyber-criminal activity that has become increasingly common. Attackers use adware to generate fraudulent clicks, with the goal of either increasing revenue for developers or depleting the budget of advertisers.
- ***Malicious Ad-Libraries:*** One type of attack involves the use of malicious ad-libraries that have access to sensitive information. These libraries can collect data through the permission mechanism and send targeted ads to users. A developer can use up to 65 of these libraries simultaneously, giving them access to a significant amount of personal data. This type of attack can be especially dangerous as it can result in the theft of sensitive information.
- ***Permission Misuse:*** Permissions are crucial for application security, but their misuse can lead to various attacks. Malware injection, malicious ads, malicious advertisement libraries, and authorities misusing permissions are the primary causes of Android phone rooting. However, rooting is only beneficial if done by the user to gain complete access to the device. Malicious software can also root the phone and take complete control of it, which is a significant security concern. Therefore, applications should not be granted 'super-user access' even if the phone is rooted.
- ***Man-in-the-Middle Attack:*** a type of attack that targets smartphone users. This attack is classified as MiTM because it involves intercepting

communication between two parties, with the attacker positioned in the middle, allowing them to read or modify data in transit. Examples of this type of attack include SSL hijacking, SSL stripping, and DNS spoofing.

3. Related work

The AdStop system [2] is a machine learning-based approach to identifying Android adware by analysing network traffic features with high accuracy, speed, and generalisability beyond the training dataset. To enhance Adware detection accuracy and reduce time overhead, the feature reduction stage was implemented. Another study proposes a machine learning-based method for detecting Android adware based on static and dynamic features. Static features are obtained from the manifest file, while dynamic features are obtained from network traffic. This approach classifies Android apps as adware or benign and further categorises each adware sample into a specific family [10]. This paper [5] also examines individual adware families and performs feature selection using information gain and machine learning classification. The best attributes for classifying each of the individual adware families are presented using network traffic samples. In addition to previous works, this paper introduces a new detection model for safeguarding smart devices against adware attacks by monitoring network traffic. To identify adware samples in the given dataset, several data preprocessing, feature selection techniques, and machine learning algorithms were employed. Seven performance metrics were used to compare ML classifiers such as Random Forest (RF), k-Nearest Neighbors (k-NN), Decision Tree (DT), Multi-Layer Perceptron (MLP), XGBoost (XGB), and Logistic Regression (LR) to determine the best approach for adware detection [11].

The objective of the paper referenced in [1] is to raise awareness regarding the potential threat to end-user privacy caused by in-app advertisements. It proposes an attack model and investigates attacks triggered by such ads. The paper also offers a theoretical framework and quantitative analytical approach to measuring the impact of embedded adware on the privacy of Android users. The authors developed the AdDetect framework [7], which utilises semantic analysis and machine learning for automatic semantic detection of in-app ad libraries. To identify and recover primary and non-primary app modules, a module decoupling technique based on hierarchical clustering is employed. The semantic features are then transformed into vectors representing each module. An SVM classifier trained with these feature vectors is utilised to detect ad libraries. This study compares 17 different supervised learning techniques for machine classification analysis. The performance of these classifier algorithms was evaluated using various metrics such as Accuracy, Precision, Recall, F-Measure, Root Mean Squared Error, Receiver Operation Characteristics Area, Root Relative Squared Error, False Positive Rate, and True Positive Rate, utilising

Figure 2. Overview of creating malicious adware dataset.

the WEKA data mining tool [3]. The paper focuses on exploring the relevance of machine learning-based solutions for detecting Android malware, specifically Adware. Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), Classification and Regression Trees (CART), and Naive Bayes (NB) machine-learning algorithms are trained and tested [12]. Another study proposes a deep logistic regression and support vector machine (DLR-SVM) trained with multiple input clusters and a malicious or benign API. It detects a malicious pattern in the unknown IoT firmware of deep LR by labelling a single output unit as malicious or benign [10]. The authors also investigate the characteristics of adware, a growing Android-based mobile malicious code, propose a learning algorithm for detecting malicious adware attacks, and present attack detection rates [7].

The system presented in this study [13] allows for the identification of crucial features in Adware and Malware, which can aid in further analysis by security researchers. The system has been tested and validated on well-known and widely used datasets, and it outperforms the leading solutions available in the market. In related work, a system has been proposed that detects adware and spyware by using classification and association mechanisms, preventing specific data theft. While the application is running, the system performs an analysis of the application data, making it a challenging task for security experts [14]. This study [15] investigated computer privacy perceptions among internet users and companies' access to privately owned computer information. It also explored consumer awareness of internet marketing practices like cookies and adware, and their potential consequences. The research suggests adware can compromise privacy and security, highlighting the need for better user awareness. The paper [16] presents a large-scale empirical study that analyses over 5 million Android apps to examine the diversity and evolution of Android malware. They use labels from 57 anti-malware vendors and propose a dissimilarity measure for clustering these labels based on scanning reports. This [17] paper introduces an intelligent honeypot that utilizes reinforcement learning to proactively engage with automated malware and optimize data collection. It shows that the intelligent honeypot captures larger datasets compared to traditional high interaction deployments. Final paper [18] discusses internal interface diversification as a proactive software security method to prevent

malware from exploiting fundamental operating system services. They diversified three main internal interfaces and found that it enhances security without significant performance costs.

4. Dataset creation

This section discusses the process of creating a malicious adware dataset. There are four phases to the process. They include data collection, feature extraction, feature selection, and finally data labelling. Figure 2 illustrates the overview of creating a malicious adware dataset.

4.1. Phase A: Data collection

The initial phase of creating a dataset is data collection. Android apps are usually collected from multiple sources. These apps are stored in the Android application package (apk) file format. The 500 malicious adware apps were downloaded from the Canadian Institute of Cybersecurity, University of New Brunswick [19]. The 1500 benign apps in various categories were collected from Google Play.

4.2. Phase B: Feature extraction

The static analysis consists of collecting features that do not require the execution of the code. We extracted permissions as static features. In this research, we used the VirusTotal online scanner [6] to extract permissions by uploading an apk file to this website and adding every permission to our dataset.

4.3. Phase C: Feature selection

The selected features play a critical role in determining a machine learning model's accuracy. It is also referred to as attribute selection. It is used to reduce

Table 2. Android permission level of protection.

Level of protection	Description	Permission Examples
Normal	Users and apps are not at risk. The permission was automatically granted, and the user did not revoke it.	ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, ACCESS_LOCATION_EXTRA_COMMANDS, ACCESS_NOTIFICATION_POLICY,
Dangerous	The user is at high risk. Apps must prompt the user and wait for the approval.	ACCESS_MEDIA_LOCATION, ACCESS_BACKGROUND_LOCATION, ACCESS_FINE_LOCATION, ACCEPT_HANDOVER
Signature	When the same certificate signs the apps, the system grants them.	BIND_AUTOFILL_SERVICE, BIND_ACCESSIBILITY_SERVICE
Signature Or System	The applications in a dedicated folder that are signed with the same certificate are granted.	BIND_CALL_REDIRECTION_SERVICE, BATTERY_STATS

irrelevant and redundant features, which aids in the selection of relevant features. Irrelevant and redundant features can degrade the classification model's quality and accuracy. Higher-dimensional datasets necessitate more storage space and computation time. Selecting relevant features will help reduce space and time complexity while also increasing accuracy. The primary goal of permissions is to protect users' privacy. Apps must ask for permission to access user-sensitive data and system features. The system may grant permission itself at times or may prompt users to accept the request. Permissions are primarily declared in the AndroidManifest.xml file. Permissions play an important role in detecting malicious Android apps. Table 2 elaborates on the protection level of Android permissions, their descriptions, and examples [20].

4.4. Phase D: Data labelling

The Android apps (apk files) obtained from the previous phase are scanned using the VirusTotal [6] tool for labelling purposes. It means that once we upload the apk file into the VirusTotal Scanner, the antimalware companies that incorporate VirusTotal need to flag the apk file as malware so that we ensure the apk file is malicious, and then we can label it as '1' which is malware. The benign apps are labelled as '0' and the malicious apps (infected with malicious adware) are labelled as '1' in the dataset. It is important to note that every antimalware company operates with its own unique databases and policies for determining whether a file should be classified as malicious or benign. This inherent variability is the primary reason behind the divergent results when an apk file is uploaded to the VirusTotal portal. Depending on the specific criteria employed by different antimalware companies, some may classify the file as risky while others may not.

Resolving conflicts when different antimalware engines disagree on the classification of an APK file is an important aspect of using VirusTotal for labelling purposes. When multiple antivirus engines are used, it's common to encounter situations where they provide conflicting results or have different opinions on whether a file is malicious or benign. It's important to note that resolving conflicts between antimalware engines is not always straightforward. Different engines may use different detection techniques, heuristics, or databases, leading to variations in their results. Additionally, false positives and false negatives can occur, where an engine may incorrectly label a file as malicious or

Figure 3. The process of extracting and selecting features.

Table 3. List of malicious Android adware families and the number of samples.

Malicious Adware Family	Year of Discovery	Number of Samples
DOWGIN	2013	10
EWIND	2015	10
FEIWO	2015	14
GOOLIGAN	2015	14
KEMOGE	2015	11
KOODOUS	2015	3
MOBIDASH	2015	10
SELFMITE	2014	4
SHUANET	2015	10
YOUMI	2014	9
Various other families	2014–2020	405
TOTAL		500

1	INTERNET	CLEAR APP CACHE	GET TASKS	CHANGE WIFI STATE	READ PHONE STATE	SYSTEM ALERT WINDOW	WRITE EXTERNAL STORAGE	CALL_ PHONE	CAMERA	READ CALL LOG		USES POLICY FORCE LOCK	WAKE UP STOKER	Risk score
2	1	1	0	1	0	1	1	0	0	0		0	0	0
3	1	1	1	1	1	1	1	0	1	0		0	0	0
4	1	0	0	0	0	0	0	0	0	0		0	0	1
5	1	1	1	1	1	1	1	0	0	0		0	0	0
6	1	1	1	1	1	1	1	0	1	0		0	0	0

Figure 4. Small portion of the malicious adware dataset.

miss a genuine threat. Therefore, it's crucial to consider multiple perspectives, rely on reputable engines, and, if possible, leverage manual review for challenging cases to achieve the most accurate labelling outcome.

5. Dataset

We present a new self-made dataset based on Android app permissions for malicious adware detection on Android platforms. As a result, we created an Android malicious adware dataset with 2000 entries. To do this, we downloaded 500 malicious Android adware files and 1500 benign apps from different categories from Google Play. To examine all apk files and extract app permissions, we used the VirusTotal online scanner [6]. In addition, we classified the apk files using over 70 trusted anti-malware detection engines. The process of extracting and selecting features is shown in Figure 3. The malicious Android adware dataset includes 10 adware families, including Dowgin, Ewind, Feiwo, Gooligan, Kemoge, Koodous, Mobidash, Selfmite, Shuanet, and Youmi. The list of malicious Android adware families and the number of samples are listed in Table 3. We put all the information in a file to make the dataset usable. CSV file format, which is simple to open and process. The dataset contains 441 columns, including 440 specific permissions and the label, which can be found in the last

column. The initial row of the dataset describes column titles, and the remaining rows contain features from 2000 malicious Android adware and benign applications. All values are in binary format, which means they are either '0' or '1'. When an app requires permission, the value in the corresponding dataset entry is '1', and when an app does not require permission, the value is '0'. Based on the VirusTotal [6] report, an Android app that is recognised as malware by most antimalware companies is possibly risky, and the respective value in the label column is set to '1', indicating malicious adware. Figure 3 shows the process of extracting and selecting features, and Figure 4 indicates a small portion of the dataset. These adware families are still actively used in research [2]. The entire dataset is available at: https://www.kaggle.com/datasets/saeedseraj/malicious-adware-detection-in-android-using-dl.

- ***DOWGIN*** is a malicious advertising module that is distributed and bundled with other (usually legitimate) programmes. The advertising module is used to display advertising content while also silently gathering and forwarding information from the device. Dowgin provides users with advertising content. If the user is unaware of the module's presence or objects to the nature of the advertising materials displayed, this behaviour may be considered unwanted. The module may also silently leak or harvest sensitive device information such as the device's International Mobile Equipment Identity (IMEI) number, location, contacts, and so on.
- ***EWIND*** is an adware trojan that was first discovered in mid-2016 and is capable of displaying unwanted ads, collecting device data, and sending SMS messages to the attacker. The trojan is distributed by decompiling legitimate Android apps, adding malicious code, and re-packaging them for distribution through third-party Russian-language Android app stores. These trojanized apps target popular apps such as Grand Theft Auto (GTA) Vice City, AVG cleaner, Minecraft – Pocket Edition, and Avast! Ransomware Removal, VKontakte, and Opera Mobile. It is important for Android users to be cautious when downloading from third-party app stores and use reputable antivirus software to detect and remove potential threats.
- ***FEIWO*** is a malicious adware for Android devices that sends the victim's phone number, IMEI, and list of installed apps to its servers. This is a common unwanted SDK that should be removed from devices. Furthermore, the adware employs several techniques to complicate its analysis.
- ***GOOLIGAN*** is a type of malware that poses as a legitimate Android app in order to trick users into installing it, thereby infecting their Android device. It can also spread by infecting apps that are downloaded from untrusted sources. Once the device is infected, the malware installs multiple unwanted apps that are difficult to remove. These apps remain on the device even after performing a factory reset, making it challenging to completely eradicate the malware.

- ***KEMOGE*** is an adware that masquerades as a popular app; it has spread so widely because it takes the names of popular apps and repackages them with malicious code before making them available to the user.
- ***MOBIDASH*** A special programme module that cybercriminals use to monetize Android games and applications. It displays various types of advertisement messages to the user. The unique feature of Adware.MobiDash.1. origin is that its unwanted activity begins after some time, rather than immediately after the malicious applications containing this module are installed or run. This period is sufficient for the user to forget about the potential source of annoying notifications and advertisements, allowing the malware to remain on the device.
- ***SELFMITE*** Security researchers have discovered a rare Android worm that spreads to other users via links in text messages. When Selfmite malware is installed on a device, it sends a text message to 20 contacts in the device owner's address book.
- ***SHUANET*** behaves more like malware and shares some ancestry with two other adware families, Kemoge and Shedun, which also root devices and provide system-level persistence to their respective payloads.
- ***YOUMI*** steals a large amount of personal information from an Android device. This includes its GPS and cell tower location, as well as phone identifiers such as the IMEI number and phone number. This differs from the data that affected stolen Apple apps, which included a list of all apps installed on the device as well as the Apple ID email address associated with the device. Symantec discovered Android's Youmi to be downloading new applications as well.

6. Proposed methodology

Controlled learning machines are capable of using labelled examples to make predictions about future events and apply knowledge learned from previous data. By analysing a specific training dataset, the learning algorithm generates a function to predict output values. With sufficient training, the programme can predict outcomes for any new input. The algorithm can also detect errors and adjust the model accordingly by comparing its output to the correct output. Machine learning and deep learning techniques are useful for analysing massive amounts of data, yielding faster and more accurate results in identifying cost-effective opportunities or risky threats. However, properly training these models may require additional time and resources. Deep learning is a particularly effective technique for efficiently processing large amounts of data [10]. Figure 5 illustrates a general overview of our process analysis from start to end.

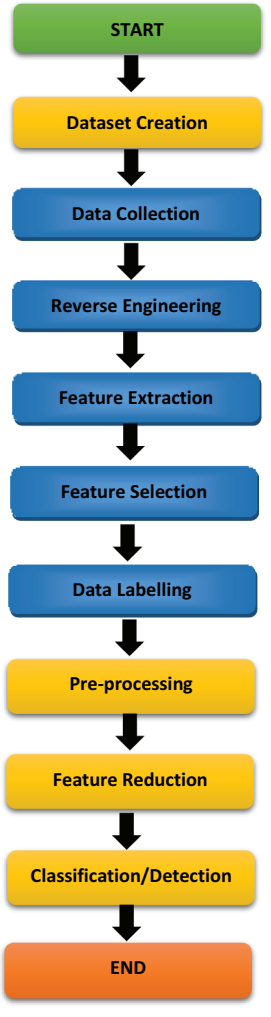

Figure 5. Overview of our process analysis.

6.1. Convolutional neural network (CNN)

A Convolutional Neural Network (CNN) is a type of feedforward neural network that allows information to flow only forward from input nodes, through hidden nodes, and to output nodes, with no loops or cycles. CNNs are primarily used for pattern recognition tasks. They are effective at detecting simple patterns in data and using those patterns to create more complex ones in deeper layers. Typically, CNNs are composed of convolutional and pooling layers. The convolutional layer detects local features from the previous layer, while the pooling layer combines similar features into a single one. CNNs use shared weights, local receptive fields, and spatial subsampling to solve high-dimensional non-convex problems with parallel and cascaded convolutional filters. These filters allow CNNs to be used for tasks such as regression, image classification, semantic

segmentation, and object detection. Compared to traditional neural networks, CNNs require fewer parameters and are easier to train due to weight sharing and processing limited dimensions.

A one-dimensional convolutional neural network (1D CNN) is useful for datasets with a one-dimensional structure, where shorter segments of the feature set can be analysed and the feature's location in the segment is irrelevant. It is particularly useful when vectorized data is used to represent the properties of the items whose state or category is being predicted, such as Android applications. 1D CNN can be used to extract more meaningful feature representations that describe patterns or relationships within vector segments. These features are then processed by a classifier, eliminating the need for separate feature ranking and selection outside of the deep learning model. In summary, 1D CNNs can be used as feature extraction layers for a given classifier, providing a more efficient and integrated deep-learning model.

6.2. Static analysis

Static analysis is an approach that does not require an application to be run and is considered passive. As the detection is done before the execution of the application, there is no impact on the system from any malicious behaviour. The manifest file, which is a component of the apk file, provides information for static analysis, including the hardware properties, permissions, themes, and activity properties for the application. The tags in the manifest file are used to define the application's permissions, such as internet access, camera access, and file reading and writing [21].

6.3. Preprocessing

The Android application permissions dataset is in CSV format and is used for training and testing purposes. Preprocessing of the dataset involves removing duplicates and NaN values. For binary classification studies, the family and category features were removed. The dataset assigns a value of '1' to malware samples and '0' to benign samples.

6.3.1. Feature reduction

Feature reduction is a common technique in machine learning and data analysis that aims to improve model performance and reduce computational complexity by reducing the number of input features. There are several reasons why we might want to perform feature reduction. Some of the common reasons are:

- To improve model performance: High-dimensional feature spaces can cause overfitting, meaning that a model is too complex and fits the training data too closely, resulting in poor generalisation performance on new data.

By reducing the number of input features, we can often improve a model's ability to generalize and make accurate predictions on new data.

- To reduce computational complexity: In many cases, large datasets with high-dimensional feature spaces can be computationally expensive to process and analyse. By reducing the number of input features, we can often reduce the computational complexity of our models and analyses, making them faster and more efficient.
- To remove irrelevant or redundant features: Some features in a dataset may be irrelevant or redundant, meaning that they do not provide any useful information for our analysis or may provide the same information as other features. By removing these features, we can simplify our analysis and improve our ability to interpret and understand our results.

The code we provided is dropping columns in our dataset where the mean of the values that are equal to 0 is greater than or equal to 0.85. This indicates that these columns contain a high percentage of zeros, which may not provide useful information for the analysis and may be effectively redundant. By dropping these columns, the code simplifies the dataset and potentially improves model performance and computational efficiency. Out of the original 440 features, the code dropped 418 columns and kept 22 columns, which are as follows:

- *INTERNET*
- *ACCESS_COARSE_LOCATION*
- *ACCESS_FINE_LOCATION*
- *GET_TASKS*
- *CHANGE_WIFI_STATE*
- *WRITE_EXTERNAL_STORAGE*
- *READ_PHONE_STATE*
- *SYSTEM_ALERT_WINDOW*
- *C2D_MESSAGE*
- *CAMERA*
- *ACCESS_NETWORK_STATE*
- *ACCESS_WIFI_STATE*
- *GET_ACCOUNTS*
- *READ_EXTERNAL_STORAGE*
- *RECEIVE_BOOT_COMPLETED*
- *VIBRATE*
- *WAKE_LOCK*
- *BILLING*
- *RECEIVE*
- *BIND_GET_INSTALL_REFERRER_SERVICE*
- *WRITE_SETTINGS*
- *INSTALL_SHORTCUT*

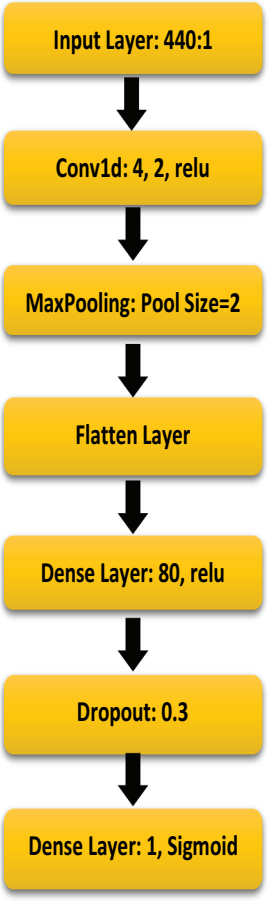

Figure 6. Proposed CNN model.

This means that 418 of the original features contained a high percentage of zeros and were considered redundant and less important, while the remaining 22 features were deemed to be more important and useful for the analysis. It's worth noting that the choice of 0.85 as the threshold for dropping columns is somewhat arbitrary and may depend on the specific dataset and analysis. In some cases, a different threshold may be more appropriate. Additionally, it's important to carefully consider the potential impact of feature reduction on the accuracy and interpretability of the analysis. In some cases, dropping too many features may result in the loss of important information and negatively impact the results.

6.4. Proposed Classifier

We used a deep-learning neural network to detect malicious Android adware in our dataset. Because of the immense flexibility of the math

performed in the overall function, a Convolutional Neural Network (CNN) is an excellent estimator in our case. It is a completely mathematical system that uses a large amount of data to gradually approximate complex input-output relationships. Figure 6 illustrates our proposed CNN model.

The number of input nodes must be the same as the number of permissions in the dataset, which is 441. Furthermore, even with so many input nodes, only one output node is required because the classification is a yes/no decision-maker. Initially, we did not use permission columns in the dataset where zeros represent 85% of the column or more. Then we identified that the best settings were a Convolutional 1D layer with a relu activation function, followed by a MaxPooling layer of size 2, and finally, a flatten layer. The following step includes a dense layer with 80 perceptrons, a dropout layer with 0.3, and the final dense layer for classification using sigmoid and Adam with a learning rate of 0.01. Finally, the batch size was set to 128 and the number of epochs was set to 20. The algorithm begins with convolution, as shown in equation 1. Where y is the output, n is the length of the convolution represented by x, and h is the kernel. S is the number of positions shifted by the kernel. In equation 2, the relu function used in the convolution layer is shown. Where a value ranging from 0 to infinite is returned for the output y and the input x. The pooling layer is applied in the following step to reduce the dimensionality to a size of 2, which aids in reducing any potential overfitting. The flatten layer then concatenates the output to form 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 an input summary. Following that, a dropout layer with 20% of the nodes is used, and the output uses a one-hot encoding with the output being either 0 or 1 for an input x based on the sigmoid function shown in equation 4.

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

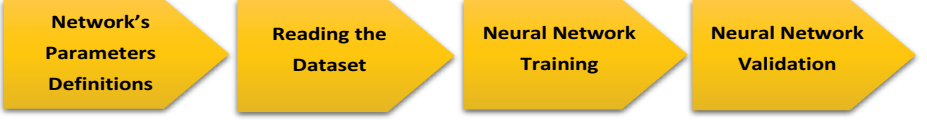

Figure 7. Demonstration of simulation stages.

$$y(x) = \max(0, x) \tag{2}$$

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

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

7. Experimental evaluation

We developed a tuned convolutional neural network (CNN) with Python and the Scikit-learn, Keras, and TensorFlow libraries as described in Section 6. Furthermore, 5-fold cross-validation was used in all experiments. We used the Python programming language to train and validate our neural network classifier on our malicious adware dataset. For array operations and reading data from files, the NumPy and Pandas libraries are required. The simulation is divided into the following stages: defining the network's parameters, such as node numbers and learning rate, reading the dataset, training the neural network, and finally validating the neural network with the remaining dataset. Figure 7 demonstrates the simulation stages.

7.1. Cross-Validation

We used a technique called 5-fold cross-validation to evaluate the model's ability to generalise to the reduced-feature dataset. This involved randomly dividing the dataset into five subsets and subjecting it to five cycles of training and testing. In each cycle, one subset was excluded from the training process and used for testing. The performance metrics of the classifier were collected for each cycle, and if the variance between these metrics was high, it indicated that the classifier was overfitted and did not generalise well. However, if the variance was low, the mean values of the performance metrics were considered reliable.

7.2. Evaluation metrics

In the experiments conducted, we utilised the Python programming language, alongside the scikit-learn, Keras, and TensorFlow libraries. To assess the performance of our approach, we employed evaluation metrics such as Accuracy, Precision, Recall, and F1, which are specified in equations 5, 6, 7, and 8, respectively. The acronyms TP, TN, FP, and FN correspond to true positive, true negative, false positive, and false negative, respectively. Accuracy, calculated via Equation 5, provides an overall indication of model performance. Precision, determined using Equation 6, describes the proportion of predicted Adware and is another vital metric. The Recall metric, as defined in Equation 7, represents the percentage of correctly classified

Adware. Additionally, we utilised the F1-score, which is a number between 0 and 1 that determines the harmonic mean of precision and recall, as expressed by Equation 8.

The basic four performance measures of a binary ML-based classifier are:

- True Positives (TP) is a performance metric that represents the number of positive samples that are correctly classified as positive by a binary machine learning classifier. It is calculated by dividing the number of test instances are true and predicted values are 1 (positive) by the total number of test instances whose true value is 1.
- False Positives (FP) refer to the number of instances in which a negative sample is predicted as positive by a binary machine learning classifier. It is calculated as the number of test instances whose true value is 0 and the predicted value is 1, divided by the number of test instances whose true value is 0.
- True Negatives (TN) refer to the number of negative samples that are correctly classified as negative by the binary classifier. Specifically, it is the number of test instances whose true value is negative (0) and the predicted value is also negative (0), which is then divided by the total number of test instances whose true value is negative (0).
- False Negatives (FN) are the number of positive samples that were incorrectly classified as negative. This is calculated by counting the number of test instances whose true value is 1 (positive) and the predicted value is 0 (negative), and then dividing by the total number of test instances whose true value is 1 (positive).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

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

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

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

8. Results

This section describes the experiments and compares the proposed method to other well-known classifiers as well as the most relevant previous research in this field. We used a self-made dataset to evaluate the proposed method and selected 500 Android malicious adware samples from 10 families. All the benign samples were scanned through

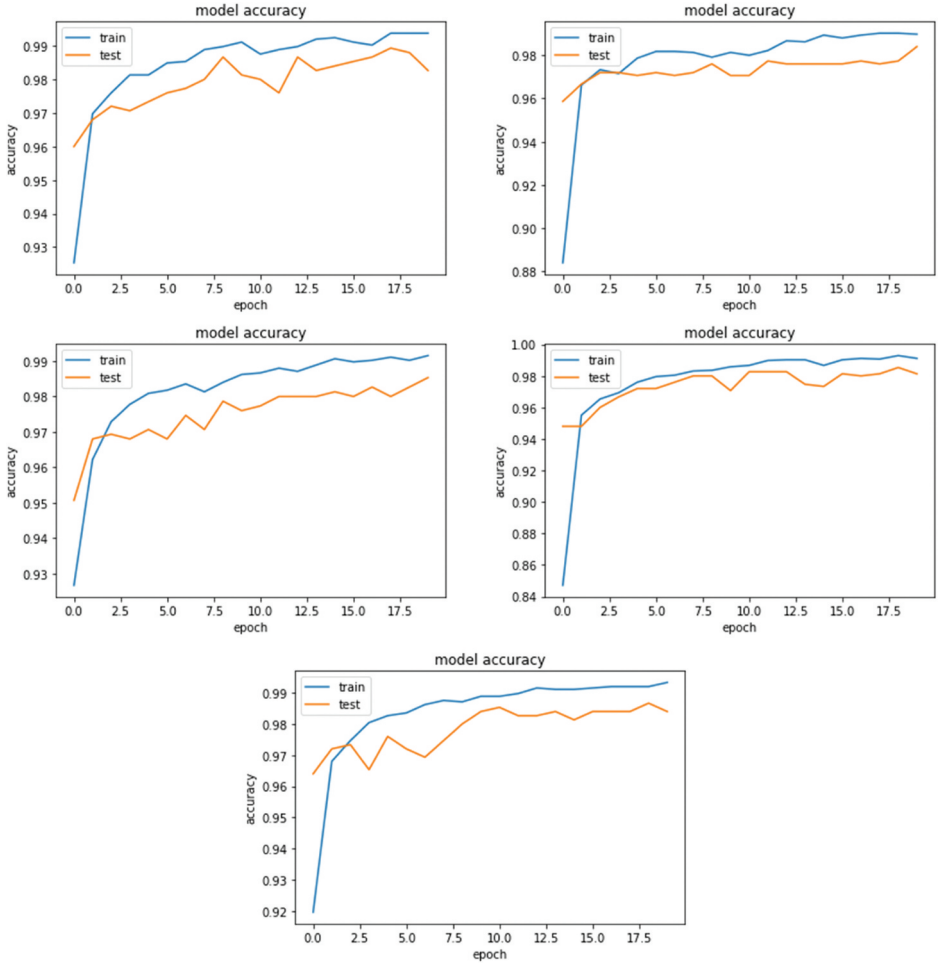

Figure 8. Training and test accuracy over epochs.

VirusTotal [6] to make sure that the benign class does not include any malware samples. The dataset contains 2000 samples, and the proposed method was evaluated using 5-fold stratified cross-validation on this dataset. In addition, all experiments were carried out on a 64-bit Microsoft Windows 11 Professional operating system with hardware including an Intel (R) Core (TM) i5-8365 U @ 1.60 GHz and 1.90 GHz CPU,

Table 4. MadDroid evaluation results.

Execution No	Accuracy%	Precision%	Recall%	F1%
Run 1	98.35	98.51	98.19	98.34
Run 2	98.53	98.41	98.67	98.53
Run 3	97.76	97.21	98.35	97.77
Run 4	98.91	98.83	98.99	98.90
Run 5	97.65	97.61	97.71	97.65
Average	98.24	98.11	98.38	98.24

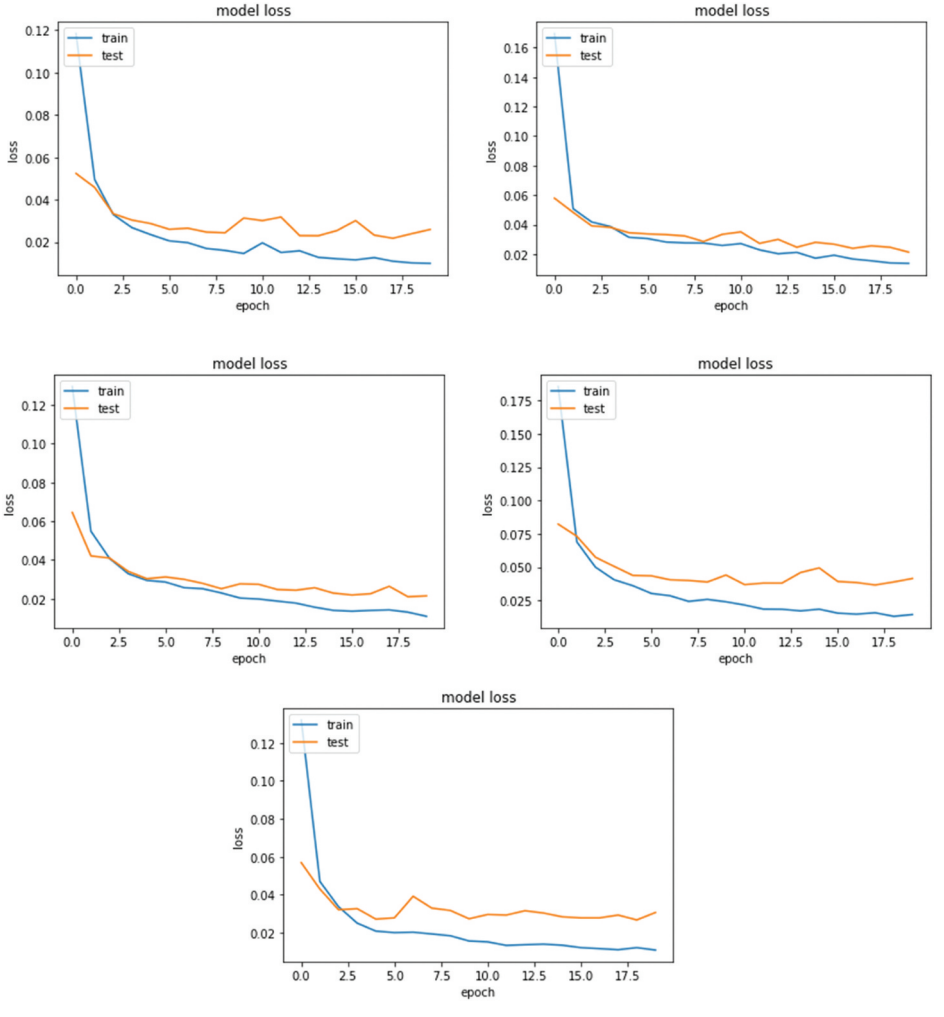

Figure 9. Model loss over epochs.

16.00GB of RAM, and an Intel UHD Graphics 620 GPU. Figures 8 and 9 show the training/test accuracy and loss over the course of 5 epochs for each of the five folds for the 5-fold cross-validation for the 1st run, the final evaluation results of which are presented in the top row of Table 4. Table 4 shows the accuracy, precision, recall, and F1 scores across 5 different cross-fold executions, and at the end is the average of these completed executions.

8.1. Comparisons with the other adware detection works

Table 5 presents the obtained results from the proposed method using CNN compared to other adware studies using the proposed dataset. The table indicates

Table 5. Comparisons with the other adware detection work.

Reference	Features	Method	Accuracy%	Precision%	Recall%	F1%
AdStop Ref [2] 2022	Network Traffic	MLP[1]	95.7	94.7	93.9	93.7
Ref [12] 2019	Adware Behaviour	LR[2]	96.0	94.5	94.9	94.6
		LDA[3]	96.2	94.3	95.7	95.0
		K-NN[4]	95.1	92.4	95.2	93.6
		DT[5]	94.2	92.2	91.8	92.5
		NB[6]	68.8	71.2	77.1	67.0
Ref [4] 2019	Malicious Codes	Dynamic Random Forest[7]	96.6	95.1	95.4	95.1
AdDetect, Ref [7] 2014	Module of apps	SVM[8]	95.4	93.7	94.1	93.8
MadDroid	Permissions	CNN	**98.24**	**98.11**	**98.38**	**98.24**

that the proposed method is more successful in classifying benign and adware applications.

The algorithms used in the comparisons are the following

[1] *Multilayer Perceptron: activation: 'relu', solver: 'adam', learning_rate_init: 0.001, 200 iterations*

[2] *Linear Regression: fit_intercept: True, normalize: False, copy_X: True, n_jobs: None, positive: False, precompute: False*

[3] *Linear Discriminant Analysis: solver: 'svd'*

[4] *K-Nearest Neighbor: n_neighbors: 5*

[5] *Decision Tree: criterion: 'gini', splitter: "best*

[6] *Naive Bayes: priors: None, var_smoothing: 1e-9*

[7] *Random Forest: n_estimators: 100*

[8] *Support Vector Machine: C: 1.0*

9. Conclusion and future works

In this paper, we present a novel method for detecting malicious Android adware using Android permissions and a tuned convolutional neural network. To the best of our knowledge, we are the first researchers to use permissions as features and apply the CNN model to detect malicious Android adware. To begin, we downloaded 1500 benign apk files from the Google Play Store from various categories and 500 apk files infected with malicious adware to create our own dataset based on permissions. The AndroidManifest.xml files that provided access to the permissions granted to each application were then extracted using reverse engineering from 1500 benign and 500 malicious adware apps from our self-made dataset. Finally, in a 5-fold cross-validation experiment, we trained and tested a proposed CNN model using the employed dataset. Our experiments show that the proposed method outperforms several conventional ML methods in this field, achieving 98.24% accuracy and 98.11% precision. These promising results suggest that the proposed method can detect

Android Adware using the permissions provided. In the future, we intend to investigate how permissions can be used to detect other types of malwares, such as Ransomware, scareware and SMS malware. Furthermore, we intend to use other types of features in conjunction with permissions to detect sophisticated Android malware.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Nikolaos Polatidis http://orcid.org/0000-0003-4249-4953

References

[1] Javaid A, Rashid I, Abbas H, et al. (2018, June). Ease or privacy? a comprehensive analysis of Android embedded adware. In *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Paris, France (pp. 254–260). IEEE.

[2] Alani MM, Awad AI. AdStop: efficient flow-based mobile adware detection using machine learning. Computers & Security. 2022;117:102718. doi: 10.1016/j.cose.2022.102718

[3] Ndagi JY, Alhassan JK (2019, December). Machine learning classification algorithms for adware in android devices: a comparative evaluation and analysis. In *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria (pp. 1–6). IEEE.

[4] Lee K, Park H (2019, July). Malicious adware detection on android platform using dynamic random forest. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (pp. 609–617). Springer, Cham.

[5] Bagui S, Benson D. Android adware detection using machine learning. Int J Cyber Soc Educ (IJCRE). 2021;3(2):1–19. doi: 10.4018/IJCRE.2021070101

[6] VirusTotal. Free online virus, malware and URL scanner, https://www.virustotal.com/.

[7] Narayanan A, Chen L, Chan CK (2014, April). Addetect: automated detection of android ad libraries using semantic analysis. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Singapore (pp. 1–6). IEEE.

[8] https://www.spiceworks.com/it-security/security-general/articles/what-is-adware/

[9] Hoffman C. What is malvertising and how do you protect yourself. HowTo Geek. 2015.

[10] Suresh S, Di Troia F, Potika K, et al. An analysis of Android adware. J Comput Virol Hack Tech. 2019;15(3):147–160. doi: 10.1007/s11416-018-0328-8

[11] Aboosh OSA, Aldabbagh OAI (2021, September). Android adware detection model based on machine learning techniques. In *2021 International Conference on Computing and Communications Applications and Technologies (I3CAT)* (pp. 98–104). IEEE.

[12] Dobhal DC, Purushottam D, Aswal K. Detection of Android adware by using machine learning algorithms (IJEAT) ISSN: 2249–8958. *Volume- Issue-4S*, *April*. India: BEIESP; 2019.

[13] Arul E, Punidha A (2021). Adware attack detection on IoT devices using deep Logistic Regression SVM (DL-SVM-IoT). In *Congress on Intelligent Systems: Proceedings of CIS 2020, Volume 1* (pp. 167–176). Springer Singapore.

[14] Lee K, Park H (2020). Malicious adware detection on android platform using dynamic random forest. In *Innovative Mobile and Internet Services in Ubiquitous Computing: Proceedings of the 13th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2019)*, Sydney, Australia (pp. 609–617). Springer International Publishing.

[15] Ideses I, Neuberger A (2014, December). Adware detection and privacy control in mobile devices. In *2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI)*, Eilat, Israel (pp. 1–5). IEEE.

[16] Soto-Valero C, González M. Empirical study of malware diversity in major android markets. J Cyber Secur Technol. 2018;2(2):51–74. doi: 10.1080/23742917.2018.1483876

[17] Dowling S, Schukat M, Barrett E. Improving adaptive honeypot functionality with efficient reinforcement learning parameters for automated malware. J Cyber Secur Technol. 2018;2(2):75–91. doi: 10.1080/23742917.2018.1495375

[18] Rauti S, Laurén S, Mäki P, et al. Internal interface diversification as a method against malware. J Cyber Secur Technol. 2021;5(1):15–40. doi: 10.1080/23742917.2020.1813397

[19] https://www.unb.ca/cic/datasets/

[20] Yerima SY, Alzaylaee MK, Shajan A. Deep learning techniques for android botnet detection. Electronics. 2021;10(4):519. doi: 10.3390/electronics10040519

[21] Bayazit EC, Sahingoz OK, Dogan B (2022, June). A deep learning based Android malware detection system with static analysis. In *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey (pp. 1–6). IEEE.