EVASION OF GOOGLE PLAY PROTECT SECURITY MECHANISMS THROUGH INCREMENTAL MALICIOUS UPDATES

By

Zia Muhammad

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

Feb 2021

<u>CERTIFICATE</u>

This is to certify that **Zia Muhammad,** Student of **MSIS-17** Course Reg.No: **00000276735,** has completed his MS Thesis title **"Evasion of Google Play Protect security mechanisms through incremental malicious updates"** under my supervision. I have reviewed his final thesis copy and am satisfied with his work.

Signature: ______________________________

Supervisor Name: <u>Dr. Muhammad Faisal Amjad</u>

Dated: ______________________________

Signature (HOD): ______________________________

Dated: ______________________________

Signature (Dean/Principal): ______________________________

Dated: ______________________________

ABSTRACT

Android is a leading mobile Operating System (OS), and its market share is increasing drastically. Every Android device has a built-in service called Play Store for application distribution and updates. A malicious application distributed through the Google play store may create a privacy breach. In order to protect end-users, an in-depth security mechanism, namely Google Play Protect, has been deployed in the Google Play Store to safeguard Android devices from malicious applications. In this work, we have investigated the malicious application detection capabilities of the Google Play Protect by employing a novel attack based on incremental malicious updates, which circumvents the security afforded by Play Protect. Therefore, a seemingly benign application called Voice Search is designed and deployed on Play Store. The Voice Search application exploits Google Play Store permissions and bypasses users' privacy through malicious updates. After malicious updates are installed, the application collects the required data such as device details, location, contact information and exfiltrates it to the attacker's server. Results show that Google Play Protect is vulnerable to malicious incremental update attacks.

ACKNOWLEDGMENTS

I am thankful to the Almighty God who gave me the power and zeal to carry out this research, and I am grateful to Him for His grace and benevolence.

I would like to convey my gratitude to my supervisor, Dr. Muhammad Faisal Amjad, and my committee members, Dr. Haider Abbas, and Col Syed Amer Ahsan Gilani, for their supervision and constant support. Throughout the experimental and thesis works, their invaluable support of insightful comments and feedback is a valuable contribution to the accomplishment of this research.

I'm really grateful to my parents. They always stood by my hopes and dreams and were a great source of inspiration for me. I would like to thank them for all their care, affection, and encouragement during my work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

OS	Operating System
APT	Advanced Persistent Threat
GPP	Google Play Protect
PII	Personally Identifiable Information
AMTs	Android Antimalware Tools
OTA	Over the Air
PHA	Potentially Harmful Applications
SPARTA	Static Program Analysis for Reliable Trusted Apps
VPN	Virtual Private Network
OEMs	Original Equipment Manufacturer
DOS	Denial of Service
SQL	Structured Query Language
XSS	Cross-Site Scripting
IRC	Internet-relay Chatbot
CFG	Control Flow Graphs
API	Application Programming Interface
HIN	Heterogeneous Information Network
CCSC	Comodo Cloud Security Center

Chapter 1

INTRODUCTION

1.1 Introduction

Android operating system (Android OS) is one of the most popular platforms to be used for mobile computing devices. More than 86.70% of the worldwide mobile device users rely on Android as their primary operating system [1]. The primary source of all software for the Android platform is Google's Play Store. Google's Play Store acts as the software distribution and update management system for all devices that are running the Android OS. The play store provides an install and update mechanism for its users. A malicious software distributed using the Android Play Store can affect millions of users [2] [3]. Therefore, Google has deployed an in-depth strategy for rooting out any malicious software that is uploaded to its distribution servers. Google uses Play protect to maintain Android security and perform in-depth analysis by thoroughly checking android applications. Play protect a multi-tiered protection system that is specifically designed to keep Android devices safe. Google Play Protect scans for malware, blocks malicious links, and performs heuristic analysis to maintain a safe user experience. Google Play protect also performs routine analysis of installed applications [4].

This research aims to perform an analysis of tools and techniques relevant to methodologies adopted by Play protect. This study aims to identify possible techniques that can circumvent the security afforded by Play protect.

1.2 Problem Statement

Being the popular platform that Android is, it is very important that the tools and techniques to secure the platform be thoroughly and critically examined. A huge number of android devices are targeted by malware in the wild. Most android users are not aware of these kinds of threats. This study will examine such techniques and provide possible breach on google play store that is considered secure by most Android users.

Google claims that Android users are secure if they install official updates from Google Play Store and Enable Google Play Protect services on their mobile devices. This claim needs to be evaluated with experimentation that either these official stores are really secure from malware attacks or not.

1.3 Research Objective

The main objectives of the thesis are:

1. Perform an analysis of tools and techniques relevant to methodologies adopted for maintaining Android security.
2. Propose a technique that can circumvent the security afforded by Play protect.
3. Demonstrating a possible breach of Play Protect security mechanisms using incremental malicious updates.

1.4 Scope of Research

The thesis's focus is to conduct a critical analysis of existing Android malware detection tools and techniques. We have evaluated their capabilities against this advanced malware propagated in the form of malicious periodic updates. A technique is introduced to bypass the android malware detection system deployed by Google Play Store.

The research applies to global Android users, researchers, and antivirus companies. This research can be used to enhance the security and performance of smart antivirus solutions to make them capable of catching advanced malware threats. Android users can know the Android Play Store's security flaws, and they will be careful while installing Android applications and their updates on their devices.

1.5 Significance of Research

Security and confidentiality is a basic need for every individual. The use of smartphones cannot be ceased due to communication dependency. Therefore, to maintain the secure use of Android phones, we need to educate people by creating awareness among all Android users to be careful while installing an Android application. Furthermore, Android users can maintain a check and balance for pre-installed Android applications maintain a secure credibility check before installing any application from the Google Play store. Periodic updates can be created in particularly launched to steal PII (Personally identifiable information). This work will express the bigger picture of Android that can establish security on the individual level, national level, and selected sensitive organizations. The key significance highlights are listed below:

- The security and performance of smart antivirus solutions can be increased by implementing security measures against these types of threats.
- Android users will get to know these security flaws and will be careful while installing Android applications on their devices.
- People worldwide would be able to know malicious updates, and they can prevent them through legal action.

1.6 Research Methodology

Our research methodology aims to develop a malicious application to circumvent Google Play Protect security checks. During our literature review, we have found that Google Play Protect is capable enough to detect/identify and block an application that tries to install Over the Air (OTA) updates. But, there is a possibility to evade and bypass Play Protect security mechanisms. This can be done if an application is spread and updated using a trusted application distribution platform like Google Play Store.

According to the Google Android security report released in 2018, applications installed from the Google Play Store are eight times more secure than applications installed from other application distribution platforms. It is due to the in-depth analysis of the application that Google performs. When an application is submitted to the Google Console, then Google starts the review process. This review procedure generally takes a period of 1 hour to 3 days to approve an application. When an application qualifies against the developer distribution agreement, and no policy violation is found, the application is made public on the Play Store.

1.7 Thesis Outline

The thesis is structured as follows:

- Chapter 2 focuses on a Literature review of Android permission, treats, and malware detection techniques. Furthermore, it covers past efforts and significant contributions.

- Chapter 3 outlines different malware detection tools that are publicly available and used to secure Android devices.

- Chapter 4 proposes our core methodology that is developed to bypass the Google Play Protect security mechanism using incremental malicious updates. It also covers the structure of our voice search application.

- Chapter 5 covers evaluation and demonstration of Play protect breach using incremental malicious updates followed by data breaches performed.

- Chapter 6 has conclusion and feature directions.

Chapter 2

LITERATURE REVIEW

We do not assume that readers have prior knowledge of Android fundamentals. The background section covers brief concepts of the Android platform for ease of their understanding like permissions, vulnerabilities, and security threats. Furthermore, we discussed past efforts, proposed models, and several attacks that are considered helpful in maintaining Android security. For a better understanding of the reader, we also added some attacks that were organized by large groups of the hacker community to steal user information.

2.1 Introduction

There is undoubtedly no end to the significance of cell phones in our daily lives and activities. In the past few years, the use of Android phones has risen exponentially. As per Google Play report, Android has more than 2 billion active devices all around the world [5]. It is the most popular platform among all kinds of cellphone devices. The global distribution of the Android OS makes it a superlative target for cybercriminals. Therefore, various types of malware are developed to target Android devices. They get installed in Android devices through different means and steal users' data like device details, contacts, messages, call logs, user location, images, and linked accounts [6].

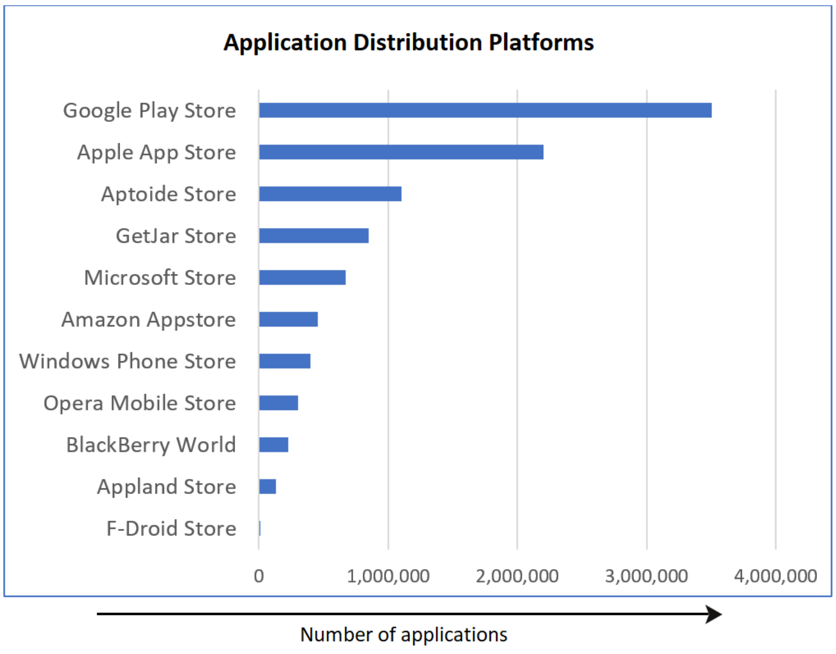


Figure 1: Application Distribution Platforms.

There are several well-known application distribution platforms through which applications can be installed on Android devices. Fig. 1 gives a brief overview of Android Application distribution platforms along with an active user base. The analytics show that Google Play Store and Amazon Application store are the most widespread. Google Play Store is on the top due to end-user saturation and the presence of hybrid applications designed at industrial standards [7].

Google introduced several security procedures to protect Android users from malicious applications. They announced Google Play Store as a default application distribution and update managing system for the Android-based devices [8]. Google has developed an exhaustive skill set to detect any malicious application uploaded on their distribution servers. Earlier, Google was using Bouncer to keep the Play Store secure from malicious applications. This was used to detect, classify, and block malicious applications. The Bouncer classified malware as spyware, trojan, adware, backdoor, and downloaders [9].

Google Bouncer was replaced with Google Play Protect services in 2017. It is a built-in mechanism that is used to identify potentially harmful applications (PHA) [10]. Play Protect is a multi-tiered malware detection system that performs routine scans for rooting out suspicious applications on an Android device. It is also responsible for performing heuristic malware analysis, including but not limited to monitoring network activity, malicious links, background services, and applications startups. In our paper, we have analyzed the security aspects of Play Protect by uploading our application to the Play Store.

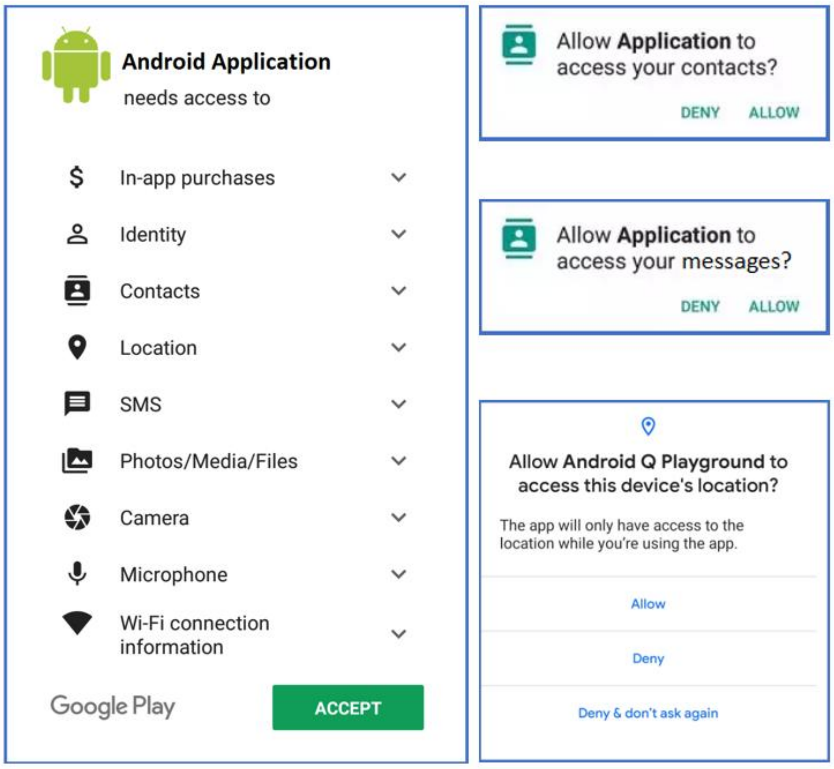
2.2 Android Background

We do not assume that readers have prior knowledge of Android fundamentals. For ease of their understanding, the background section covers brief concepts of the Android platform like permissions, vulnerabilities, and security threats. Furthermore, we discussed past efforts, proposed models, and several attacks that are considered helpful in maintaining Android security. For a better understanding of the reader, we also added some attacks that were organized by large groups of the hacker community to steal user information.

2.2.1 Android permissions

According to the latest release of Android OS (Android Marshmallow 6.0 - API 23), dated

October 03 2017, every installed application needs to declare all required permission in its manifest.xml file using the "uses-permission" tag. Hence, an application provider cannot use any permission without giving a custom dialog prompt [11]. An overview of permission dialogue can be seen in Fig. 2.


Android Application needs access to

Allow **Application** to access your contacts?

Allow **Application** to access your messages?

Allow Android Q Playground to access this device's location?

The app will only have access to the location while you're using the app.

Figure 2: Android permission dialogues.

There is still a need to define why various permissions are required for an application. Several algorithms are proposed to classify malware by analyzing the Android permissions model, such as Amandroid, and SPARTA (Static Program Analysis for Reliable Trusted Apps) [12]. According to these models, all Android permissions can be classified into normal, signature, dangerous, and special permissions. [11]

2.2.1.1 Android normal permissions

There is no need to create a run time prompt for normal permissions as they are automatically allowed to the application without interference. Android examines the list of permissions in the manifest.xml file and allows normal permissions on runtime automatically. Android normal permissions are categorized as access network state, Bluetooth, access location, install application shortcuts, and kill background processes/ activities.

2.2.1.2 Android signature permissions

These permissions require a runtime prompt to request permissions from the user. Sometimes, a developer needs to explain the purpose of asking for this permission. These permissions pop up when an application starts for the first time. These signature permissions are not allowed automatically. Android ensures that digital certificates are being used to access and define these permissions by an application. These permissions include various services such as bind VPN (Virtual Private Network) service, text services, voice interaction, telecom connections, manage documents, and request install packages.

2.2.1.3 Android dangerous permissions

This category includes a list of permissions that can potentially affect device operations. These permissions are related to the user's privacy and security. An Android application can not use these permissions until the user explicitly grants it. A runtime prompt is compulsory to ask for dangerous permissions. These permissions include read/ write contacts, messages, logs, calendar, external storage, record audio, and answer/ call phone.

2.2.1.4 Android special permissions

Special permissions are mostly defined by OEMs (Original Equipment Manufacturer) or mobile distribution platforms to restrict access for a, particularly commanding operation. All special permissions have their implementation mechanisms and need to create permission dialogue. These include write settings and generate a system alert window.

2.2.2 Android vulnerabilities

Android OS has evolved tremendously over a period of time. In the last few years, plenty of security improvement is introduced to Android devices. In this research, a survey has been conducted that covers publicly published Android vulnerabilities that were identified and patched in a timely manner. Fig. 3 gives a detailed graphical representation of the number of vulnerabilities reported every year [13]. The year 2017 is considered the most vulnerable year because Android was targeted with massive attacks. Upon close examination of the bar graph, it can be identified that a total of 843 vulnerabilities were identified in that year.

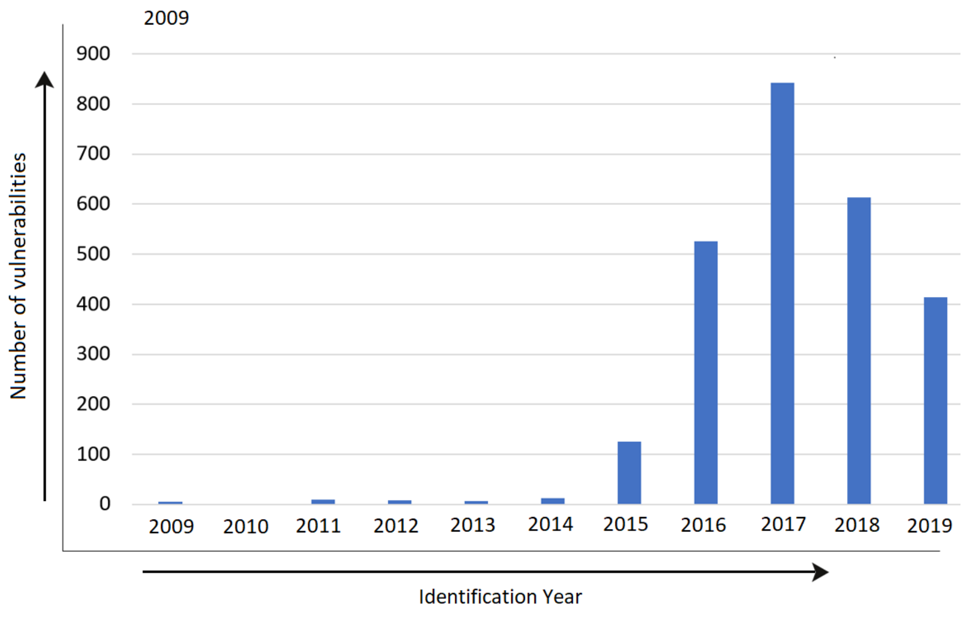


Figure 3: Android vulnerabilities identified by year.

Fig. 4 gives a detailed graphical representation of different types of vulnerabilities published across different Android versions like DOS (Denial-of-Service), code execution, the device bypasses, memory corruption, potential information gain, buffer overflow, SQL injection (Structured Query Language), directory traversal, and XSS (cross-site scripting). Upon close examination of the bar graph, it can be identified that the two most occurring vulnerabilities are code execution and buffer overflow.

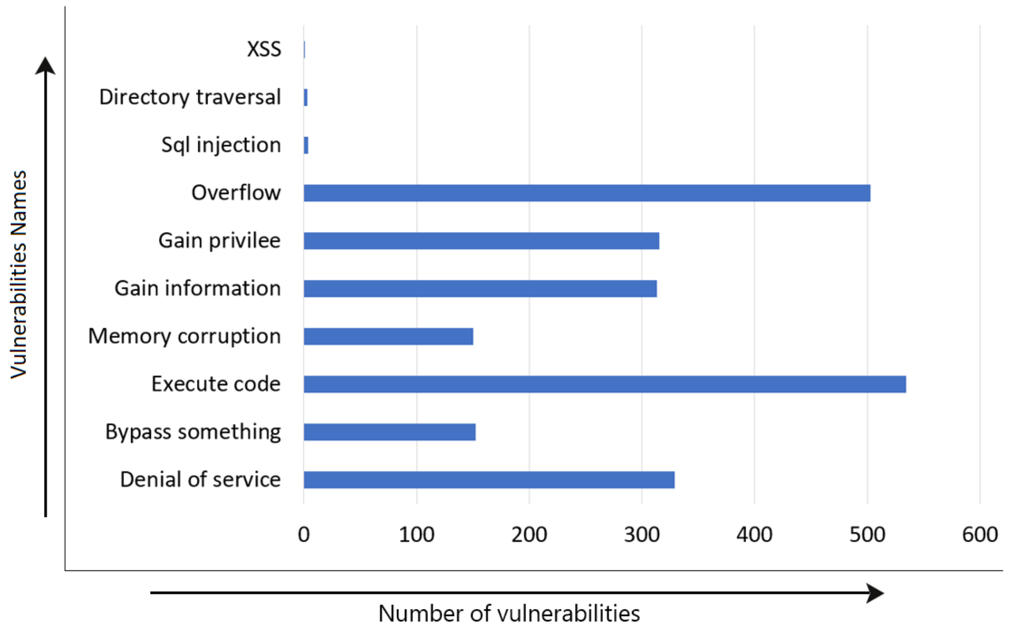


Figure 4: Android vulnerabilities identified by type.

2.2.3 Android Security Threats

Android is the most targeted platform for attackers due to its popularity. A vast number of Android devices are attacked every day.

During the research, it is noticed that Android is targeted with multiple security threats over time. These threats affected multiple Android devices and caused data loss for numerous users. These threats can be classified into the following categories. Fig. 5 gives a brief overview of android security threats followed by detailed definations. [14] [15]

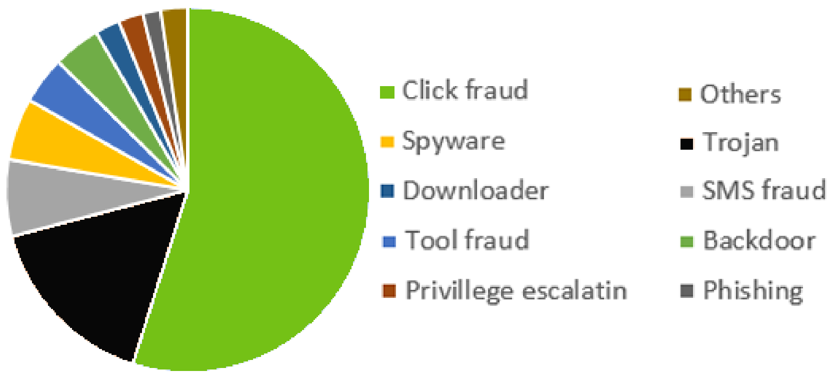


Figure 5: Classification of Android malware.

1. **Adware:** It is a piece of code that displays advertisements. Its service runs in the background and creates a popup window to display ads and sell fake products.
2. **Trojan:** It is a malicious application that hides its identity and misleads users by pretending something like legitimate software using a valid icon and title.
3. **Backdoor:** It is a covert way to bypass normal authentication in a device. Backdoor can be created by a compromised application or an unauthorized person to perform an illegal remote activity. Furthermore, a backdoor can be used to establish a communication channel. It allows attackers to perform malicious activities like sniffing, activity monitoring, call logs, and SMS backups. A backdoor can also be used to steal PII [9].
4. **Spyware:** It is an application that steals personally identifiable information, contacts, messages, call logs, and device details. Subsequently, these applications send the stolen information to their command and control servers.

5. **Click fraud:** Hackers are paid when a user clicks on an advertisement, and they create fake clicks by overlaying buttons, images, and test layouts over advertisements. Click Fraud leads to more clicks and increased revenue.
6. **Smishing/ SMS fraud:** Targeted victims receive messages that contain website links. Clicking on these links directs them to the desired page for malware installation.
7. **Logic bombs:** It is a piece of code that is intently added in a software solution to start malicious functionality when some specific condition meets. Some successful experiments are conducted to exploit Android permission models using logic bombs [16].
8. **IRC-bots:** Attacker infects multiple Android devices to create IRC (Internet-relay chatbot). In this attack, at first, a group of machines is infected to get their control. Subsequently, these machines are used remotely through the Internet Relay Chat channel to launch DOS-like attacks against desired targeted platforms.
9. **Money-mulling/ Financial fraud:** These are targeted attacks in which fake bank applications are propagated to gain banking details. Furthermore, valid dumps and inactive accounts are filtered and compromised. They used for money transfer and receiving for money laundering purposes and illegal payments.
10. **Pharming:** It is used to generate/ manipulate traffic to a specified website. An application is propagated that overwrites the host file, overrules the original DNS. This leads to a targeted website and generates fake traffic.
11. **Ransomware:** It is malicious software that infects device to encrypt data, displays a message to users, and demands money to restore data access. Ransomware gets installed by fake websites and misleading links.
12. **SMS worms:** These are malicious applications that propagate themselves. Worms can spread by sharing links with saved contacts on the device. Sometimes worms are used to create targeted attacks on the specific word's presence in the contact list.
13. **Rogue ware:** These are fake software solutions that pose as anti-malware applications. They generate a popup message that warns the user that their device is infected and that the virus can be removed after installing their software. This phenomenon leads to a rogue ware installation that leads to information stealing.
14. **Keyloggers:** Android keylogger applications are propagated to steal keystrokes. They record keys when the user types something and copy anything on the device.

2.2.4 Android Malware Detection Techniques

Android malware is hard to detect, and it is due to the inherited issues native to android operating (OS) systems and mobile devices. This includes permission issues in android OS and low computational resources of the mobile platform. This makes comprehensive malware analysis on mobile devices extremely difficult. We have to perform analysis on both mobile and dedicated servers to maintain accurate detection mechanisms for Android-based malware. All Android malware detection techniques use similar steps to perform malware analysis on an Android application, their entire method is the same. Still, the way of doing it and their implementation method can vary. On the most abstract level, we can categories Android malware detection in three major steps, as shown in Fig. 6.

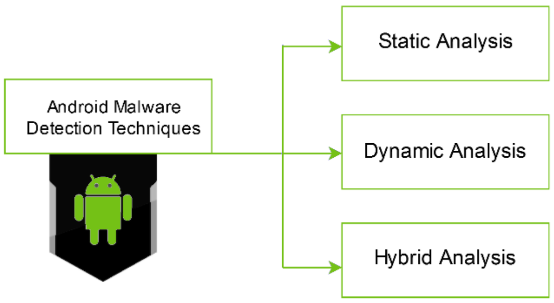


Figure 6: Android malware detection techniques

All malware detection techniques work on the same principles for malware detection. First of all, an Android Package Kit (APK) is identified and analyzed using any malware detection technique, and in the end, the report is generated. There exist multiple Android malware detection techniques depending on our requirements. There exist some predefined and well-known methodologies that we are going to discuss. We can broadly classify Android malware detection mechanisms into three major categories that are static analysis, dynamic analysis, and hybrid analysis. This is shown in Fig. 7.

2.2.4.1 Static Analysis

Static analysis technique detects the malicious applications without running a malicious file[17]. Static analysis can give various types of malware information, including function information, opcode sequence, Control Flow Graphs (CFG) [18], malware signatures, Android permissions, Dalvik bytecode, etc. All this information can be used to make a dataset, and therefore, different artificial intelligence (AI) techniques can be applied to that dataset. There are a lot of static analysis variants that exist. They perform malware analysis by selecting custom features like a combination of signature-base-detection and CFG's. There exist some other approaches that use deobfuscation and machine learning techniques to perform static malware analysis. The latest static malware analysis uses a combination of different techniques for malware classification and detection.

2.2.4.2 Dynamic Analysis

Dynamic analysis is performed on Android applications by running a malicious sample in virtually designed environments like virtual machines and mobile emulators. Dynamic analysis is used to detect malicious behavior through different detection mechanisms based on features like runtime behavior, system calls, device traces, registry changes, Application Programming Interface (API) calls, system calls, memory writes, instruction traces, monitoring network traffic, API call logs, etc. There exist different malware detection techniques based on selective features like fine-grained models that use system calls to analyze Android malware behavior. Machine learning algorithms are trained using custom datasets to perform dynamic malware analysis. API calls, runtime behavior, network traffic, and other key features are extracted to build a sequential model of required malware and classified based on training data set. The dynamic analysis technique is considered a more accurate detection technique than static analysis because it can detect runtime code execution and monitor real-time activity in parallel to network traffic monitoring. Table. 1 gives a quick overview of Android static and dynamic analysis.

Table 1: Static vs. dynamic analysis

Category	Static Analysis	Dynamic Analysis
Analysis	Perform analysis without running malware samples.	Perform analysis by running malware samples.
Methodology	Source code is extracted from APK file with the help of reverse engineering tools like Dex2Jar or APK tool etc.	Runtime analysis is performed by checking system calls, execution paths, dynamic privacy leaks, power consumptions, and network traffic.
Chosen parameters	The analysis is performed based on permissions, suspicious patterns, and system API calls, etc.	The analysis is performed based on Runtime analysis, behavior patterns, and system dynamic code loading, etc.

Nature of approaches	Most static approaches are signature-based approaches.	Most dynamic approaches are behavior-based approaches.
Pros	Simple and easy to implement, which require fewer resources and computational ost.	Analyzing runtime behavior creates network overhead and increases computational cost and requires high-end resources to implement.
Cons	This technique is not effective against obfuscated and today's state of the art malware.	This can counter obfuscated and today's malware due to runtime analysis techniques.

2.2.4.3 Hybrid Analysis

The hybrid malware analysis technique combines key aspects of both static and dynamic malware detection mechanisms. This uses a combined analysis technique that checks the malware sample based on selected features of static analysis and dynamic analysis. This includes but is not limited to signature-based-detection, intent base detection, API calls, system calls, runtime behavior, device traces, functional calls, classes names, services created, broadcast incited, opcode sequence, CFG's, malware signatures, Android permissions, Dalvik bytecode patterns, device traces, registry changes, memory writes, instruction traces, etc. The process of hybrid analysis is not limited to this. It is a customizable technique that provides flexibility to make a feature set of your own choice based on requirements and perform analysis with higher accuracy and less false positive rate. There are many standards-based on custom hybrid analysis techniques to counter state of the art Android malware. Fig. 7 gives a quick overview of hybrid analysis.

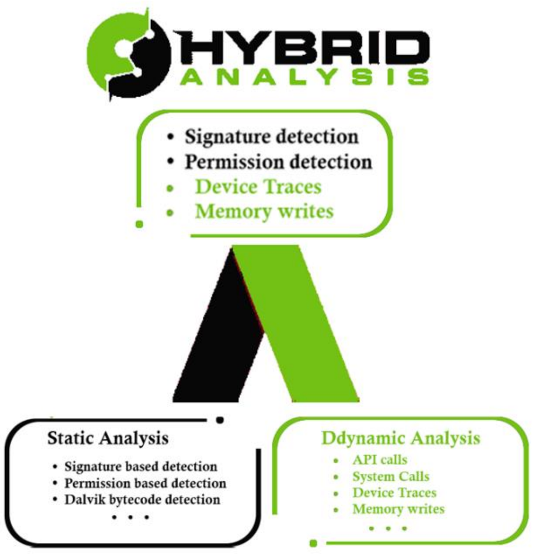


Figure 7: Classification of Android malware using multiple Android malware analysis techniques.

2.3 Related Work

Due to our work's novelty, we found a comparatively less quantity of associated literature in our domain. However, several authors have nominated different aspects of Android device security like creation, propagation, and malware detection among Android devices. Recent studies explain the state of the art techniques that are used to secure Android users from malware attacks. The related work is provided in the ensuing paragraphs.

Google Play Protect uses multiple types of time complexity algorithms for spam detection. It classifies applications into different categories based on their functionality and performs routine checkups to detect all installed applications' suspicious activities. Furthermore, a rule-based filtering mechanism is used to enhance the detection algorithm's throughput [19]. Consequently, the research is conducted to check the frequency of the updates of published applications [20]. In this research, the authors have proposed a bi-weekly Play Store application updates mechanism effective against application visibility and helps to attain more users on the Play Store. The proposed method helps create a new application and release its updates in a timely manner to get more visibility and installs in the Google Play Store.

Several papers are presented to detect and classify new and repackaged malware designed to infect Android devices. In this research [21], the author has described the working principle of Android malware. He used state of the art forensic tools to visualize the behavior and working model of Android Malware. Multiple aspects are envisioned, such as malware propagation, working principle, and activity flow. Similarly, broad research on repackaged malware is presented here [22]. In this research, the authors have described a functional model to detect repackaged malware. This research briefly explains how a code is injected into an existing application by exploiting the actual behavior and how they are made undetectable. These kinds of malware are hard to detect, and their proposed technique is a useful contribution. Similar to malware detection, malware classification is considered vital to see the efficiency of malware detection tools. In Springer publication [23], the authors have researched existing Android malware classification and categorization techniques. They proposed a model based on modus operandi and existing malware vendor reports to classify the state of the malware.

More and more Android scholars are working to ensure Android safety and assessing the strengths of existing detection tools for quality assurance. In this regard, the author has demonstrated how to bypass automated malware analysis systems in the book [24]. he bypasses antivirus solutions, Android sandboxes, and Google Bouncer by proposing a tool called Sand-Finger. This is a relatively old study, but it covers the essential aspects of bypassing techniques. The author explained a quick pathway to bypass Android malware detection tools. This study is not directly applicable to new malware detection tools as they are more vigorous and efficient. The baseline for massive Android threats is efficient malware propagation. Detailed research has been conducted on malware propagation mechanisms here [8]. In this article, the study has been conducted on possible malware propagation among Android devices. Researchers analyzed propagation techniques and explored the spread capabilities of multiple malware in a targeted environment. The author analyzed multiple malware behavior and introduced three different malware propagation states: susceptible, Latent, and breaking state. The proposed model calculates the malware propagation's threshold based on these three states.

Efficient malware detection is compulsory for the safety of Android users. Accordingly, different authors have proposed diverse detection models. In chapter [25], the researchers have designed a set of rules to evaluate an Android application. They perform a detailed study on the malware scanning process and briefly described malware detection mechanisms. They designed evaluation phases and a set of rules to consider while analyzing an Android application. Their primary focus is to protect Android users from attacks by using both static and dynamic analysis. They introduced a ranking system for applications based on trustworthiness, patterns, and the Android permission model.

In IEEE/ACM International Conference 2019 [26], the authors have proposed a malware detection approach based on network analysis. The analysis is performed by analyzing network activity. Authors collected Android applications from the Google Play Store and analyzed their internet usage and network connectivity. Their core detection model is based on Android permissions, intent actions, signatures, discriminative APIs, and pattern recognition. Similarly, a functional malware detection mechanism is presented in Elsevier journal [9]. The researchers have introduced a hierarchical embedding approach for the detection of callback-based APIs and are named as Callback2Vec. The offered solution is based on application behavior and information losses. Moreover, the proposed solution is helpful for the detection of downstream Android applications that use callbacks as a standard API that is used in some malware.

The authors have conducted a large-scale empirical case study on Google Play Store in the book [27]. They describe Google Play Store policy violations that lead to application termination. Google has developed a set of policies that apply to all Android application developers. It is a basic necessity for Play Store developers to comply with these policies if they want to publish an Android application to the Play Store. Furthermore, if a developer violates a policy, then its application is removed by the Play Store. Some severe violations lead to account termination as well.

All of these works are comprehensive efforts to ensure Android malware detection mechanism, propagation techniques, classification/ categorization based on their behavior, repackaging of existing malware, and an efficient malware scanning process. Although this work is a great motivation in research, however, none of these studies evaluate Google Play Protect services against custom-designed malware updates. In contrast, our research work analyses Google Play Protect detection mechanism and demonstrates a malicious bypass.

Chapter 3

ANDROID MALWARE DETECTION TOOLS

3.1 Introduction

This section contains a detailed analysis of already existing malware detection tools. It includes a comprehensive survey for preexisting tools freely available in the Android market. As with the passage of time, new detection tools have been introduced to counter the latest malware threats. In this paper, we have analyzed all the primary Android malware detection tools discovered in the period of 2014 to 2019. All the tools are thoroughly described along with their advantages and drawbacks. They are classified on the basis of factors like accuracy rate, false-positive rate, and training dataset used for algorithm training.

3.1.1 RoughDroid

RoughDroid [28] is a comprehensive hybrid malware analysis tool that was introduced in 2018. This tool is based on features extraction and machine learning algorithms. The author has classified all selection features in such a way that ten features set has been introduced. That feature sets are FS1, FS2, FS3, … FS10. These feature sets are classified into two major groups. One is a set of an XML file that contains 7 feature sets, and the second is based on DEX file that includes 3 feature sets. The algorithm is trained in such a manner to identify malicious applications based on these feature sets. This technique purely focuses on Hardware components, software components, Android permissions, application components, application activities, intent filters, application services, API calls, and behavior analysis in the DEX file. RoughDroid is trained under Deribit dataset and succeeds with a higher accuracy rate of 95.6% and false positive rate of 1% but, the underlying "Deribit dataset" is old and has been used 100's of times, and it does not include advanced malware families and the latest threats of smart device like Agent-Smith Android malware, Copycat malware, SpyDealer malware, GhostCtrl malware, Marcher malware, Dvmap malware. RoughDroid was evaluated using different datasets like the Drebin dataset and 179 distinct families of malware. After thorough analysis, the applications are categorized as malware applications, adware applications, and benign applications.

3.1.3 Drebin

Drebin [29] is a lightweight static analysis tool that was introduced in 2014. This tool is capable of identifying malicious Android applications on the Android phone directly. It uses a joint vector space to extract maximum features of an Android application for the purpose of malware detection with an accuracy rate of 94% and 1% false positive. The detection process takes an average of 10 seconds. The entire detection process consists of 8 vectors and performs analysis on vector bases. Drebin is capable of detecting malicious applications on the smartphone by analyzing its malicious activity and constructs a comprehensive vector space by analyzing different application features. First of all, static analysis is performed on an Android application in which its hardware components, application Components, filtered intents, requested permissions, restricted Application Program Interface (API) calls, permissions used, suspicious API calls, and network addresses are analyzed and mapped against a vector space. This vector space is further used for learning-based detection and categorizes malware as benign or malicious.

3.1.4 AppFA

A comprehensive tool [26] was introduced in the period of 2018. It is based on a novel dynamic approach for the detection of malicious Android applications on the network. It's lightweight and is a very efficient framework that uses an efficient algorithm to cluster the network traffic of the application. There is no need to install a specific program on an Android device or for system modification. This tool is capable of handling encrypted traffic while carrying our analysis online and uses a constrained clustering technique to classify network traffic of an Android application. This tool uses an efficient algorithm to get and check network traffic. This technique is originally implemented on a public dataset and Google play store applications with an accuracy rate of 90%, and a false-positive rate is less than 0.4%.

3.1.5 DroidDector

DroidDector [30] is a comprehensive hybrid analysis tool that was introduced in the period of 2016. It uses deep learning techniques and performs static and dynamic analysis in parallel based on 192 feature sets for higher accuracy. This is an online deep learning base detection engine that uses a feature set for detection containing required permissions and sensitive API calls. It classifies Android applications and categorizes them based on listed permissions. DroidDector can broadly be classified feature sets into three categories, which are required permissions, sensitive APIs, and dynamic behaviors of an Android application. DroidDector is trained under a dataset of 21760 applications collected from Android stores. This tool performs in-depth analysis with an accuracy rate of 96.76% and a false positive rate of 0.9%.

3.1.6 TinyDroid

TinyDroid [31] is a lightweight tool that was introduced in 2018. This is implemented using static analysis and use machine learning. First of all, an APK is decompiled, and the opcode sequence is extracted and classified depending upon application features. then that application is decompiled, and the opcode is extracted. Furthermore, these opcodes are used in combination with N-Gram to predict an application as malicious or benign. TinyDroid is efficient and fast as compared to other antivirus applications. Under the testing of 4000 application samples, it gives an accuracy rate of 98.6% and a false-positive rate of 1.4%.

3.1.7 DynaLog

DynaLog [32] is a dynamic analysis malware detection tool that was introduced in 2016. It uses a wide variety of dynamic features to classify any Android application into benign or malicious applications. This framework is built on some preexisting open-source tools like a mobile sandbox and its detection mechanism, including mass analysis, application characterization along with API calls, and performance of critical events in an application. This technique is analyzed in 1940 application samples containing both benign and malicious applications. DynaLog is designed in such a manner that it can analyze multiple malicious applications at the same time. The entire detection process is composed of five major components. First of all, an application is launched in an emulator, and its logs are extracted to perform further analysis. In the end, an emulator-based analysis sandbox is used to classify Android applications as benign or malicious. DynaLog gives an accuracy rate of 93.29% under the discussed dataset.

3.1.8 ICCDetector

ICCDetector [33] is a static analysis malware detection tool that was introduced in the period of 2016. This tool is trained with 17290 malicious and benign applications that are collected from different Android markets and include other open source applications too. ICCDetector is capable of analyzing malware on the basis of self-defined ICC characteristics and ICC patterns of applications. All these patterns are obtained by analyzing an Android application on the basis of app components, intents, intent filters, required permission, etc. This tool is capable of analyzing all types of malicious applications, and all these applications are roughly classified into five major categories, which are server connector, system monitor, advertiser, effective launcher, and telephony abuser. ICCDetector set a benchmark for an accuracy rate of 97.4% with a false positive rate of 0.67%.

3.1.9 CASSANDRA

CASSANDRA [17] is a static analysis tool that was published in 2017. This tool extracts features from an Android application and uses an online learning mechanism to flag an application as malicious or benign. The used methodology can be subdivided into four major modules as static analysis, features extraction & representations, online learning, machine learning-based malware detection. It uses contextual inter-procedural control flow graphs to gather contextual and structural information from a malicious application. This graph-based technique makes CASSANDRA more scalable and helps to achieve an accuracy rate of 99.23% over the analysis of 87000 applications. CASSANDRA is trained under 87000 applications collected from different application stores.

3.1.10 MADAM

MADAM [2] is a hybrid analysis tool that was published in 2019. This tool uses key features of both static and dynamic analysis and performs detection only on rooted devices. This is a host-based multilevel framework that classifies on the basis of system calls, user activities and develops a behavioral pattern for unauthorized kernel-level activities for identification. MADAM is trained with three different types of data sets, which are Genome, virus share database and Contagio-Mobile dataset. It uses behavior base Android malware detection and detects if an application misbehaves. Moreover, it has the ability to remove the malicious application in Android and stops its further prorogation. This tool uses a signature base and anomaly-based detection mechanism to give an accuracy rate of 96.9% that is tested over 9804 applications.

3.1.11 APK Auditor

A static analysis tool [34] was introduced in the period of 2015. This is capable of performing permission-based malware analysis to classify targeted applications as malicious or benign. APK Auditor is based on three components, which are the Android client module that is installed in an Android device, signatures database, and a server that is responsible for communication between the module and database. The tool analyzes applications on the basis of Android permissions, receivers, and services to classify them as benign or malicious. The model is tested with 8762 Android applications from different Android stores and has an accuracy rate of 88% with a false-positive of 0.925%.

3.1.12 MalDozer

MalDozer [35] is a dynamic analysis tool introduced in the period 2018. It performs dynamic analysis on the basis of API calls by using deep learning algorithms. This is a comprehensive framework having the capability to deploy on servers, mobile phones, and Internet Of Things IoT devices. First of all DEX file is extracted from the APK file, and further assembly is extracted by the DEX file. Furthermore, API method calls are extracted from assembly and used for the development of a Tensorflow artificial neural network. The framework is trained with 38000 benign and 33000 malware samples from these open source datasets: Malgenome, Drebin, Virusshare samples, and Contagio Minidump. This framework gives an accuracy rate of 96%-99% with a false positive rate of 0.06%-2% as the training dataset is of 71000 applications.

3.1.13 AndroDialysis

AndroDialysis [36] is a static analysis malware detection tool introduced in the period of 2017. It performs static analysis on the basis of implicit and explicit intents created in an Android application. The framework is trained with 7406 applications, among which 1846 is benign, and 5560 are malware from different datasets like ProfileDroid and Drebin dataset. This is not a perfect solution because it's very easy to evade and can be misled under trained dataset. Moreover, this approach is tested under the given dataset, which gives an accuracy rate of 91% and a false positive rate of 4.4%

3.1.14 Apposcopy

Apposcopy is a static analysis based tool that was introduced in the period of 2014. It performs static analysis on the basis of Android application signatures to identify specific malware families. Along with signature base detection, it uses "Inter-Component-Call-Graph" for efficient malware detection. This framework is purely based on signature-based detection that is inefficient and is easy to evade by the latest malware families. First of all, an APK is decompiled, and permissions are extracted from the Android manifest file. A total of 1027 malware samples were collected from the Malware Genome project, and the Accuracy rate for detection was 90%, and the false positive rate is 2%.

3.1.15 M0Droid

M0Droid is a static analysis tool introduced in the period of 2015. It performs static analysis based on Android signatures and application behavior. This approach is lightweight and is based on two components; a client agent, which is installed on an Android application, and a server analyzer. To perform analysis, they work in parallel to identify malicious applications. M0Droid is analyzed with a Genome dataset and achieved a detection rate of 60.16% and a false positive rate of 39.43%. This technique is not up-to-date and cannot be further used for malware detection. The accuracy rate of this tool is very low due to the poor implementation of signature-based techniques.

3.1.16 SeqDroid

SeqDroid [37] is a hybrid analysis tool that was introduced in the period of 2019. It uses Recurrent Neural Networks and Stacked Convolutional to detect obfuscated Android malware. This technique is robust, lightweight, and has the latest detection technologies for the detection of obfuscated and runtime dynamic creation of strings and package names. This dataset is tested under 2,000,000 malware samples from VirusTotal, along with benign applications are 888,620. Results were classified on the basis of 5 different tier based methodologies as Ngram, RNN, Ngram-PA, RNN_PA, CNN_RNN_PA, and SeqDroid attained up to 95% accuracy rate, and false positive is 0.001%.

3.1.17 DroidEvolver

DroidEvolver [38] is a static analysis tool that was introduced in the period of 2019. It evolves itself with machine learning without a user's continuous evolvement. The framework updates itself and its dataset from online learning techniques. This tool is truly helpful in the detection of code obfuscation, and the entire detection is based on API calls. DroidEvolver is evaluated under 33,294 benign and 34,722 malicious applications and attained an accuracy rate of 95.27%, along with a false positive rate of 0.48%.

3.1.18 HinDroid

HinDroid is a static analysis tool introduced in the period of 2017. It detects malicious Android applications using API calls along with structured Heterogeneous Information Network (HIN). They claim that this is a novel approach that has been used for the first time for malware detection in which analysis is performed by HIN on extracted API calls. This framework is trained under the malware dataset obtained from Comodo Cloud Security Center (CCSC). Dataset is trained under 32,334 applications, among which 16,118 are benign applications, and 16,216 are malicious applications. android has an accuracy rate of 98.60% and a false positive rate of 4%. \\\\ Android malware can further be classified on the basis of their approaches and the way of their classification in Android malware categories.

A comprehensive survey is performed on preexisting tools that are freely available in the Android market. All the tools are thoroughly described along with their advantages and drawbacks. Table .2 explains the classification on the basis of factors like accuracy rate, false-positive rate, and training dataset used for algorithm training.

- AR: Accuracy Rate (%)
- FP: False Positive Rate (%)

Table 2: Different malware detection tools.

Name	**Approach**	**year**	**Description**	**Advantages**	**Discussion**	**Data set**	**AC**	**FP**
Drebin [29]	Static analysis	2014	Use a joint vector space to extract maximum features like API calls, use permissions, network addressed, hardware and software components	The detection process takes an average of 10s	A static approach without Runtime code linking detection feature	129013 GooglePlay Store, Chinese Markets, Russian Markets, Genome Project	94	1
APK Auditor [34]	Static analysis	2015	Use permission-based malware analysis to classify targeted applications and has three components as the client, server, and signatures database	The tool analyzes application permissions, receivers, and services to classify them as benign or malicious, not just signatures.	Obfuscated malware with code words can bypass this detection mechanism. That's why it has a low accuracy rate	8762 Application stores	88	0.92
DroidDector [30]	Hybrid analysis	2016	Use deep learning on listed permissions in the manifest file and sensitive API calls and other 192 feature sets with	Dataset is trained using Deep learning on both API calls and permissions	Less effective and can be fooled by the latest malware families	21760 Google Play Store, Contagio Community, Genome Project	96.76	0.9

			deep learning algorithms					
ICCDetector [33]	Static analysis	2016	Analyze app component, Intents, Intent filters, and required permission	Malware is analyzed based on ICC characteristics and ICC patterns of applications.	Unable to detect malicious code loading on run time.	17290 Application store, opensource APK	97.4	0.67
CASSANDRA [6]	Static analysis	2017	use online learning based on contextual Inter-procedural control flow graphs to gather contextual and structural information	Graph-based technology makes it fast and efferent with a higher accuracy rate.	Advance malware with runtime and dynamic malicious code linking can easily evade this.	87000 Application store, opensource APK	99.23	0.86
RoughDroid [28]	Hybrid analysis	2018	Extract 10 Features extraction and apply machine learning	Perform Fast and comprehensive analysis based on ten features that make malware evading almost impossible.	Deribit dataset" is old, and it doesn't include advanced malware families	Deribit dataset	95.6	1
Tiny Droid [31]	Static analysis	2018	Use Opcodes in combination with N-Gram to predict as malware and benign	Efficient, lightweight, and fast due to the use of the machine learning technique	Small training dataset, this can be modified and retrained with new malware samples	4000 Application store, opensource APK	98.6	1.4
MADAM [2]	Hybrid analysis	2018	a host-based multilevel framework that classifies based on system calls, users activities and develops behavioral patterns for unauthorized kernel-level activities for identification	The model is trained under great datasets and achieved a higher accuracy rate that makes it worthy for the latest malware	Only rooted devices are eligible for analysis, and the latest malware can execute malicious payloads	Genome, Virus share database, Contagio-Mobile dataset	96.9	0.5 – 1.1
MalDozer [35]	Dynamic analysis	2018	A comprehensive framework based on API calls and methods along with deep learning algorithms and trained with multiple datasets to gain a higher accuracy rate	This is not limited to android phones, but the framework is capable of deploying on servers, mobiles phones, and IoT devices	Tensorflow artificial neural network needs more computational cost and expensive to deploy	71000 Malgenome, Drebin, Virusshare, Contagio Minidump	96 - 99	0.06 – 2
SeqDroid [37]	Hybrid analysis	2019	Use Recurrent Neural Networks and Using Stacked Convolutional to detect obfuscated Android malware and trained under great data set of up to 2 million	Its robust, lightweight and has the latest detection technologies for detection of obfuscated and runtime	Ngram, RNN, Ngram-PA, RNN PA, CNN RNN PA are hard to implement and increase computational cost	2888620 VirusTotal	95	0.001

			applications	dynamic creation of strings and package names.				
Droid Evolver [38]	Static Analysis	2019	Online learning-based detection of code obfuscation and detection is based on API calls.	Evolve itself with machine learning without user's continuous evolvement	Only static based API based approach is not so capable of countering today's threats	68016 Open source	95.27	0,48
SDAC [39]	Static Analysis	2020	The API call sequence is extracted, and a Neural Network is used to assign API to vector space and API clusters.	The feature set is adaptive and able to adapt new features in runtime that make it detect new and advanced malware.	The training data set is old and might not contain advanced malware samples that make it a week.	Open source app 2011 - 2016	98.1	0.1 – 1.9
DL-Droid [40]	Dynamic Analysis	2020	Works based on stateful input generation and use different stateless approaches for code coverage.	A good approach that gives a feature to combine static analysis with dynamic to get up to 99.6% accuracy rate.	Not able to counter today's obfuscated malware and very limited dataset used for training.	30,000 Open source apps	97.8	0.1 – 2.2

Chapter 4

PROPOSED TECHNIQUE TO BYPASS SECURITY AFFORDED BY GOOGLE PLAY PROTECT

4.1 Introduction

Our methodology aims to develop a malicious application to circumvent Google Play Protect security checks. During our literature review, we have found that Google Play Protect is capable enough to detect/identify and block an application that tries to install Over the Air (OTA) updates. But, there is a possibility to evade and bypass Play Protect security mechanisms. This can be done if an application is spread and updated using a trusted application distribution platform like Google Play Store.

According to the Google Android security report released in 2018, applications installed from the Google Play Store are eight times more secure than applications installed from other application distribution platforms. It is due to the in-depth analysis of the application that is performed by Google. When an application is submitted to the Google Console, then Google starts the review process. This review procedure generally takes a period of 1 hour to 3 days to approve an application. When an application qualifies against the developer distribution agreement, and no policy violation is found, then the application is made public on the Play Store.

4.2 Voice Search Application - Design and Architecture

The high-level architecture of the Voice Search application is shown in Fig.~\ref{fig5}, detailing the connectivity of the application developer, Play Store, and the end-user. It is a sequence diagram that provides an overview of the proposed methodology. The sequence is as follow:

1. An application named "Voice Search" is developed and uploaded on the Google Play Store. It allows users to perform various actions through voice commands. This application allows the following type of voice commands to a user:
    i. Call someone by speaking their name.
    ii. What is the current weather?

iii. Latest news articles.

iv. The shortest way to California.

Due to its attractive functionality, in a very short period of time, it is sported 12,782 types of Android devices and is made available in 151 countries. Fig.~\ref{fig6} shows our application published on the Google Play Store.

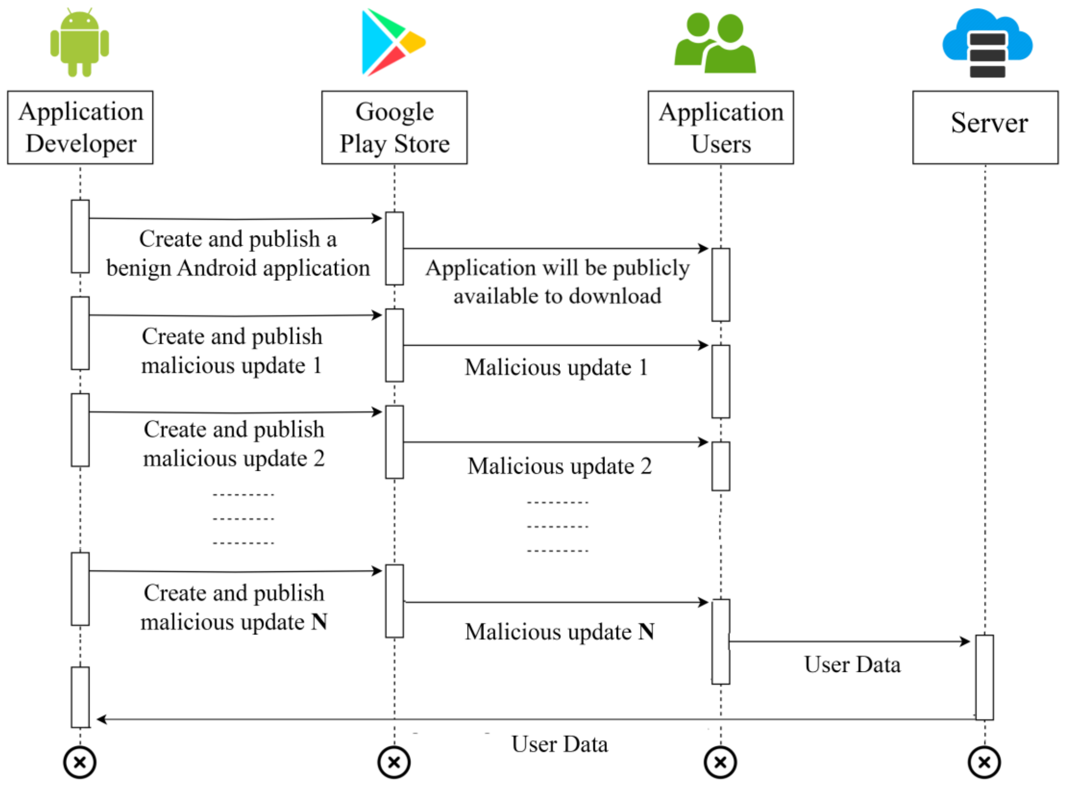

Application Developer
Google Play Store
Application Users
Server
Create and publish a benign Android application
Application will be publicly available to download
Create and publish malicious update 1
Malicious update 1
Create and publish malicious update 2
Malicious update 2
Create and publish malicious update N
Malicious update N
User Data
User Data

Figure 8: Proposed methodology to bypass Google Play Protect.

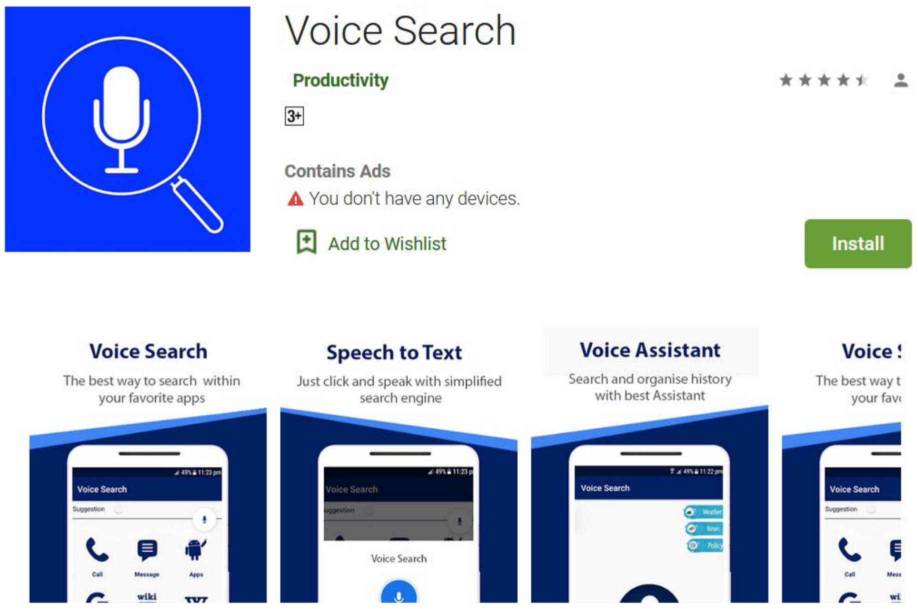
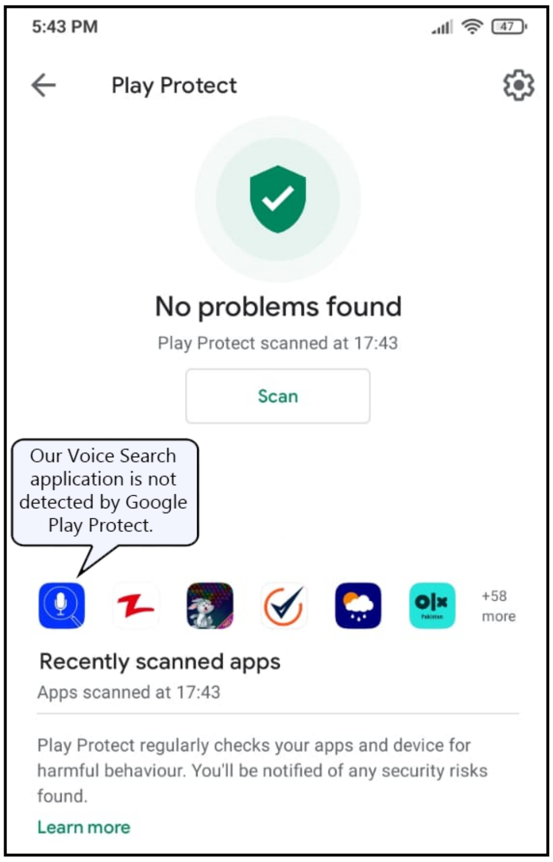

Voice Search
Productivity
3+
Contains Ads
You don't have any devices.
Add to Wishlist
Install
Voice Search
The best way to search within your favorite apps
Speech to Text
Just click and speak with simplified search engine
Voice Assistant
Search and organise history with best Assistant

Figure 9: Malicious application published on Google Play Store.

2. An early version called V1.1 is a legitimate application that performs publicly announced functionality. This version was submitted to Google Console on October 31, 2020. Google took seven days to review and accepted the application as legitimate on November 6, 2020. Since then, the Google Play Store has made the application publicly available.
3. After some days, a second version for Voice Search was developed and named V1.2. In this version, we started or malicious functionality by adding analytics, event logs, activity tracking, demographics, and user location. This update was submitted to Google Console on November 16, 2020, and was accepted on November 17, 2020. Google took just one day to review our update and made it public on Google Play Store.
4. Subsequently, a third version, V1.3, was developed. This was an entirely malicious update that was capable of exploiting android permissions. It was collecting device contacts, version, API level, manufacture, and model details. This version was capable of creating a reverse connection on Firebase storage to store data against each user's entity. All this data was collected when the user was performing its first voice action. This version was submitted to Google Console on December 26, 2020, and was accepted on December 27, 2020. Google took just one day to review our update thoroughly, and this malicious version was made public on Google Play Store.


5:43 PM
Play Protect
No problems found
Scan
Our Voice Search application is not detected by Google Play Protect.
+58 more
Learn more

Play Protect scanned at 17:43

Recently scanned apps

Apps scanned at 17:43

Play Protect regularly checks your apps and device for harmful behaviour. You'll be notified of any security risks found.

Figure 10: Google Play Protect scanning process.

The experiments conducted show that the Google Play Store performs an in-depth analysis that takes more days when an application is initially published on the Play Store, while it takes less time to review when an update is created to an existing application. We analyzed with different experiments that it is difficult to upload a fully malicious application, but the same could be done in multiple updates.

We downloaded this application on multiple Android devices to test against Google Play Protect, but Play Protect was unable to detect our malicious application. Although Play Protect performs a routine analysis on installed applications in the device and checks against the potentially harmful activity but still, it was not able to detect this kind of attack. Fig.~\ref{fig11} shows a device screenshot in which Google Play Protect scanned all applications, including our malicious one, but it was not able to detect our application.

Chapter 5

EVALUATION AND DEMONSTRATION OF PLAY PROTECT BREACH USING INCREMENTAL MALICIOUS UPDATES

5.1 Introduction

Our evaluation is based on the accuracy and effectiveness of the presented work. The proposed methodology provided an effective way to propagate a malicious application using a most trusted platform like Google Play Store. The effectiveness of the proposed methodology is measured and evaluated by launching and performing a malicious attack through the Google Play Store. This means that any kind of benign application can be created and published, and an application developer can exploit and make it malicious at later stages. This proves that, If this kind of attack is launched in multiple applications, then it will cause massive distribution in global users and will be hard to stop it.

5.2 Effectiveness

We created three versions for the Android application and published them on Google Play Store in a timely manner. Table. 3 contains application publication details against their version number. The table has three columns:

1. Version: This column contains the application version that contains major increments.
2. Processing time: This column explains the number of days that Google took for evaluation before making a version live on Google Play Store.
3. Functionality: This column contains application functionality that was introduced in a specific version.

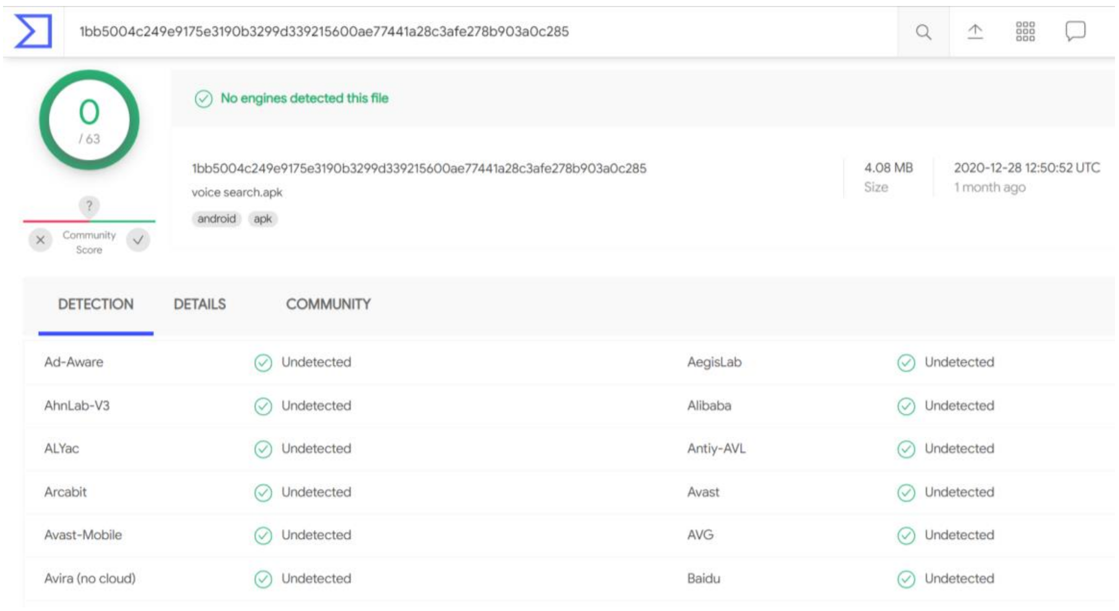
Table 3: Application publication workflow.

Version	Processing Time	Functionality
V 1.1	7 Days Submission: 31-10-2020 Acceptance: 6-11-2020	- Benign App. - App permissions. - 12,782 devices.

		- 151 countries. - Support API 16 – 30.
V1.2	1 Days Submission: 16-11-2020 Acceptance: 17-11-2020	- Added analytics. - Event logs. - Activity tracking. - Demographics. - Userbase location. - Affinity audience.
V 1.3	1 Days Submission: 26-12-2020 Acceptance: 27-12-2020	- Permission exploit. - Firebase data backup. - Contacts backup. - Device details. - Data is backup on voice actions.

5.2.1 Application detection across scanners

After a successful demonstration to bypass the Google Play Store detection mechanism, we test the "Voice Search" application across different anti-malware solutions. We analyzed the APK file of our application across multiple antiviruses solutions using Virustotal. These are application details collected after its thorough analysis across different antivirus and anti-malware platforms. A complete scan report is available here [41] and a quick overview of the detailed scanning report is visible in Fig. 11.


1bb5004c249e9175e3190b3299d339215600ae77441a28c3afe278b903a0c285

0 / 63

No engines detected this file

1bb5004c249e9175e3190b3299d339215600ae77441a28c3afe278b903a0c285
voice search.apk
android apk

4.08 MB Size

2020-12-28 12:50:52 UTC 1 month ago

Community Score

DETECTION DETAILS COMMUNITY

Ad-Aware	Undetected	AegisLab	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
Avast-Mobile	Undetected	AVG	Undetected
Avira (no cloud)	Undetected	Baidu	Undetected

Figure 11: Application detection across virustotal.

Moreover, the "Voice Search" application was tested against malware solutions such as the Cuckoo sandbox and Anyrun malware sandbox. The application was undetectable; application details of the designed "Voice Search" is included below for further analysis:

- **MD5:** 96a4a3fd70f627155133bec58b3f1d23
- **Vhash:** 6b3340a01f08d92e7d05c44f2ac42a77
- **SHA:** 092f13d3c2915d3efc6139b9bf2bba1dbab5ca26
- **SHA-256:** 1bb5004c249e9175e3190b3299d33921s5600ae77441a28c3afe278b903a0c285
- **Common Name:** Android APK
- **The organization hosted:** Google Inc, Mountain View (California), US
- **Certificate Issuer:** C:US, CN:Android, L:Mountain View, O:Google Inc., ST:California, OU:Android
- **Virus Total Submission:** 2020-12-28 12:50:52

5.3 Accuracy

This section explains the user's data that has been collected throughout our experiment. We organized the collected in a particular order for better understanding. All the collected data is stored in Firebase servers. Firebase is a store platform launched by Google [42]. It is considered the most trusted platform for analytics and back-end services.

The data collection phase was simple when a new user installs our application first time from the Google Play Store and opens it, and then this functionality was executed:

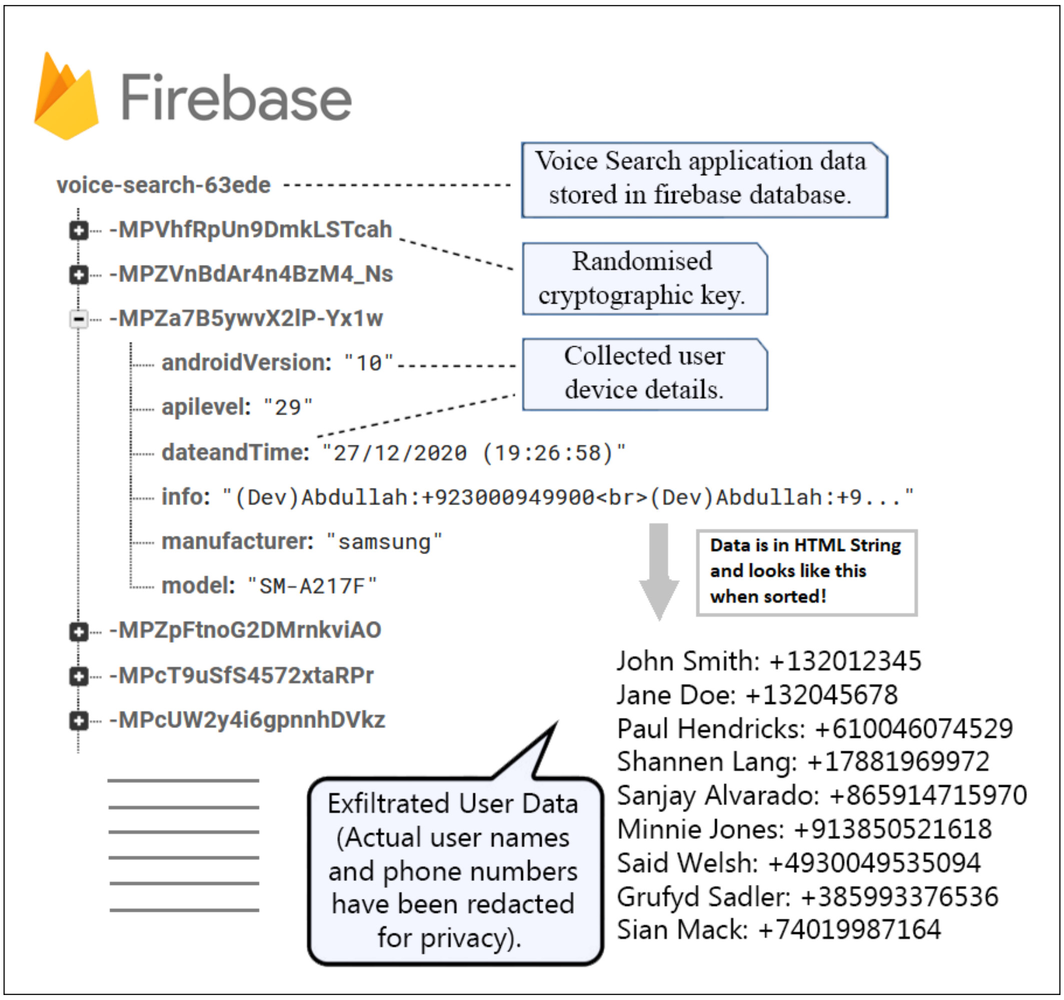
1. User entry was created in Firebase with a unique key.
2. User contacts, device details, location, and demographics were uploaded to the selected destination.
3. The device's current date-time was logged, and the device was marked successful in avoiding feature repetitions.

Afterward, this data was safely stored and in a particular order. The collected data was organized into two sections, i.e., Firebase Bulks and Analytics data.

5.3.1 Firebase Bulks

This contains data bulk details stored in a Firebase server. Fig. 12 shows data stored in Firebase against every user's random key.

1. **MPZa7B5ywvX2lP-Yx1w:** This entity contains a Firebase unique key for individual devices.
2. **Manufacturer:** It contains the name of the mobile company that is the manufacturer of devices, e.g., Samsung, Nokia, Huawei.
3. **Model:** This entity contains device model details, e.g., Galaxy S3, Nokia 2.2.
4. **Android version:** This entity contains Android version numbers like Android 10, 9, 8.
5. **API level:** It contains an integer value for a specific API.
6. **Date and time:** This entity contains a local timestamp at the time of data uploading.
7. **Info:** This entity contains a string that has device contacts locally stored in the system.
8. **Demographics:** This shows the overall statistical view for the gender and age group of the audience that installed this application.
9. **Location:** These are statistics for countries/ regions in which the application is installed.
10. Activity: This is a graphical representation for users who installed the application every day.
11. **Affinity Audience:** These statistics give a detailed report about application users based on their interests, lifestyle, habits, passion, and online activities.



```
voice-search-63ede
  -MPVhfRpUn9DmkLSTcah
  -MPZVnBdAr4n4BzM4_Ns
  -MPZa7B5ywvX2lP-Yx1w
      androidVersion: "10"
      apilevel: "29"
      dateandTime: "27/12/2020 (19:26:58)"
      info: "(Dev)Abdullah:+923000949900<br>(Dev)Abdullah:+9..."
      manufacturer: "samsung"
      model: "SM-A217F"
  -MPZpFtnoG2DMrnkviAO
  -MPcT9uSfS4572xtaRPr
  -MPcUW2y4i6gpnnhDVkz
```

Voice Search application data stored in firebase database.

Randomised cryptographic key.

Collected user device details.

Data is in HTML String and looks like this when sorted!

```
John Smith: +132012345
Jane Doe: +132045678
Paul Hendricks: +610046074529
Shannen Lang: +17881969972
Sanjay Alvarado: +865914715970
Minnie Jones: +913850521618
Said Welsh: +4930049535094
Grufyd Sadler: +385993376536
Sian Mack: +74019987164
```

Exfiltrated User Data (Actual user names and phone numbers have been redacted for privacy).

Figure 12: Firebase data storage.

5.3.2 Analytics Data

The Analytics data contains a graphical representation of Android users. Fig.~\ref{fig8} graphically displays the geolocation of users that has installed the application. As the image shows, most of the audience is coming from India, Russia, and the United States. These graphs change in a timely manner when new users are acquired from different locations.

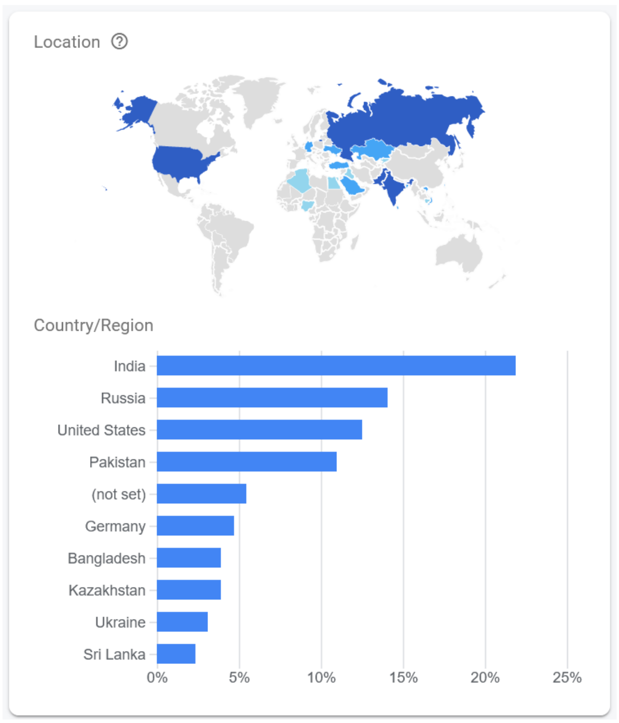


Figure 13: Users' location data collected by voice search application.

Fig.~\ref{fig9} contains a graphical representation of the targeted audience based on their ages. All users are classified into age groups. The minimum age group we targeted in our application is 18 years. We do not allow under 18 users to install our application. Application users are largely classified into two major age groups, i.e., the 18-24 years group and the 25-34 years group. We also have broadly classified our total audience into groups based on their gender, and we got 55.2% males and 44.8% females.

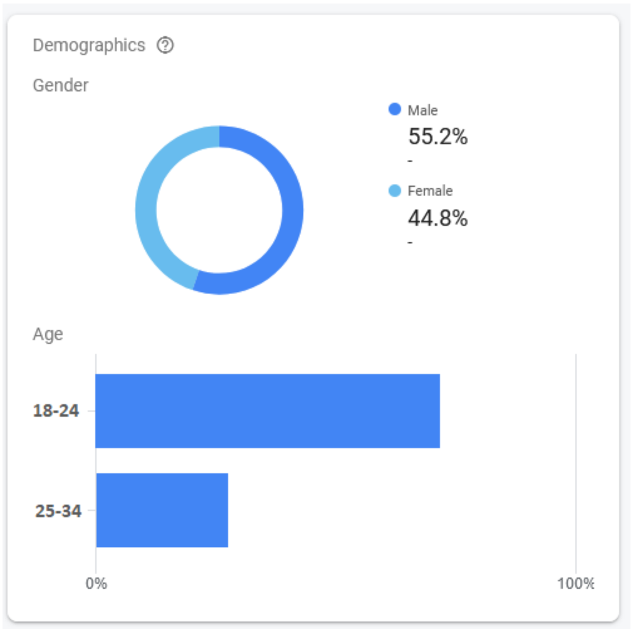
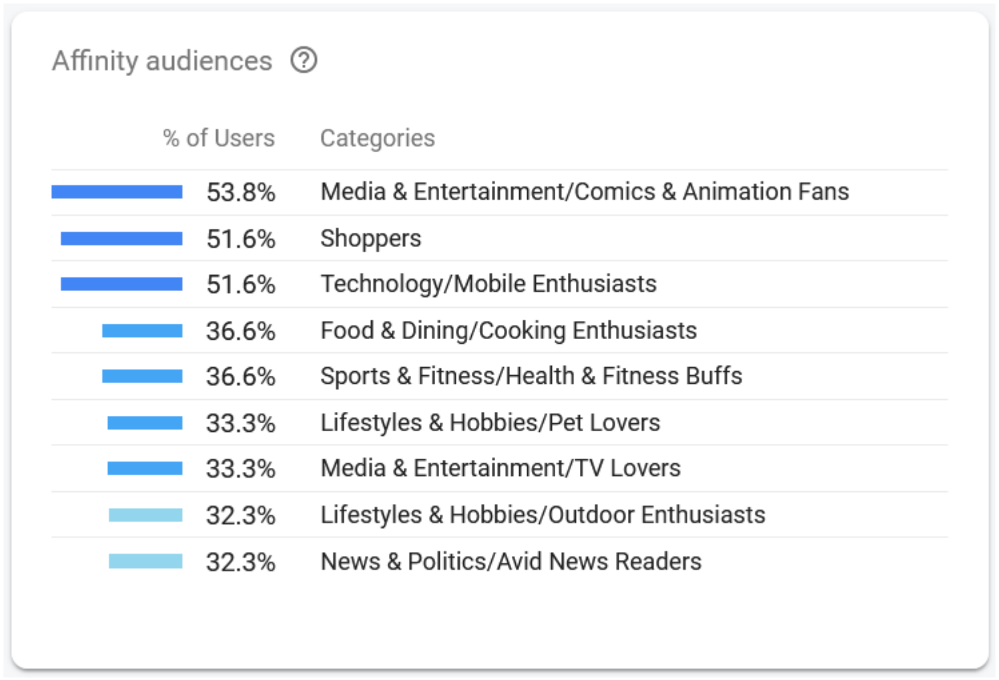


Figure 14: Users' age group and gender collected by voice search application.

Fig.~\ref{fig10} gives a brief overview of our affinity audiences. It categorizes users into their respected categories base on past activities, interests, and hobbies. The Firebase analytics algorithm keeps track of their daily activities and assigns an affinity group based on their recent behavior. The major categories are news, politics, media, entertainment, shoppers, sports, fitness, technology, traveling, vehicles and transportation. These group categorization leads to custom advertisements and can be used to generate more sales.


Affinity audiences

% of Users	Categories
53.8%	Media & Entertainment/Comics & Animation Fans
51.6%	Shoppers
51.6%	Technology/Mobile Enthusiasts
36.6%	Food & Dining/Cooking Enthusiasts
36.6%	Sports & Fitness/Health & Fitness Buffs
33.3%	Lifestyles & Hobbies/Pet Lovers
33.3%	Media & Entertainment/TV Lovers
32.3%	Lifestyles & Hobbies/Outdoor Enthusiasts
32.3%	News & Politics/Avid News Readers

Figure 15: Users' data relevant to their interests, lifestyle, habits, passion, and online activities collected by voice search application.

After installing the application and collecting desired data from devices without getting noticed by Play Protect, it can be concluded that such attacks are hard to detect. Attackers can attach this malicious piece of code to the popular Android application. Then he/ she can further propagate these applications to a targeted audience using a messaging service.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Google promises its users to provide secure and authentic applications if they install from the Google Play Store and enable Google Play Protect services on their devices. These services are available on all the latest Android devices by default. In our research, We performed auditing/evaluation of Google Play protect. We experimented and designed a multi-step model and successfully bypassed Google malicious application detection mechanisms by incremental malicious updates. In this regard, two policies of the Google policy center have been violated and breached during our experiment, i.e. (1). Privacy, deception, and system abuse. (2) Malware policy. These policies ensure that Android users are provided with safe and secure applications through Play Store [43].

This experiment has demonstrated a possible breach that can circumvent the security afforded by Google. Although Play Protect was introduced back in 2017 and it is still inefficient for the detection of malicious applications, particularly when periodic updates are released to exploit the Android permission model. If this kind of malware is released in public, then data bulks can be generated in a couple of days and can be used to exploit individual users and entire organizations. To block such attacks, a thorough analysis of updates is recommended. As seen in Table: 3 that Google Play Store took seven days to scan the application, which was published for the first time but less than one day to scan its update. To block such kinds of attacks, it is compulsory to perform an in-depth analysis of every update, not just the initial application submission.

6.2 Future work

This research invites Android researchers and developers to investigate and counter modern security breaches. Users can use this research for a better understanding of available free applications in online markets against different vulnerabilities. This research can further be used to investigate other application distribution platforms and operating systems like the IOS Apple store and Windows app stores.

BIBLIOGRAPHY

[1] R. Riasat, M. Sakeena, C. Wang, A. H. Sadiq, and Y. Wang, “A Survey on Android Malware Detection Techniques,” *DEStech Trans. Comput. Sci. Eng.*, no. wcne, Jan. 2017, doi: 10.12783/dtcse/wcne2016/5088.

[2] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, “MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention,” *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 1, pp. 83–97, Jan. 2018, doi: 10.1109/TDSC.2016.2536605.

[3] “CRAZY Android vs iOS Market Share Discoveries in 2020,” *Tech Jobs*, Nov. 15, 2019. https://leftronic.com/android-vs-ios-market-share/ (accessed Dec. 01, 2020).

[4] N. Viennot, E. Garcia, and J. Nieh, “A measurement study of google play,” in *The 2014 ACM international conference on Measurement and modeling of computer systems - SIGMETRICS ’14*, Austin, Texas, USA, 2014, pp. 221–233, doi: 10.1145/2591971.2592003.

[5] “App Stores List (2020) - Business of Apps.” https://www.businessofapps.com/guide/app-stores-list/ (accessed Feb. 11, 2021).

[6] A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, “Context-aware, Adaptive and Scalable Android Malware Detection through Online Learning (extended version),” *ArXiv170600947 Cs*, Jul. 2017, Accessed: Sep. 14, 2020. [Online]. Available: http://arxiv.org/abs/1706.00947.

[7] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, “End Users’ Perception of Hybrid Mobile Apps in the Google Play Store,” in *2015 IEEE International Conference on Mobile Services*, New York City, NY, USA, Jun. 2015, pp. 25–32, doi: 10.1109/MobServ.2015.14.

[8] I. Almomani and M. Alenezi, “Android Application Security Scanning Process,” in *Telecommunication Systems - Principles and Applications of Wireless-Optical Technologies*, I. A. Alimi, P. P. Monteiro, and A. L. Teixeira, Eds. IntechOpen, 2019.

[9] C. Guo, J. Zhu, X. Yan, and Y. Li, “Security Threats Caused by Public Event Callback in Android Application,” *J. Phys. Conf. Ser.*, vol. 1453, p. 012127, Jan. 2020, doi: 10.1088/1742-6596/1453/1/012127.

[10] “Potentially Harmful Applications (PHAs) | Play Protect,” *Google Developers*. https://developers.google.com/android/play-protect/potentially-harmful-applications (accessed Dec. 01, 2020).

[11] “Permissions on Android,” *Android Developers*. https://developer.android.com/guide/topics/permissions/overview (accessed Nov. 27, 2020).

[12] A. Mahindru and P. Singh, “Dynamic Permissions based Android Malware Detection using Machine Learning Techniques,” in *Proceedings of the 10th Innovations in Software Engineering Conference*, Jaipur India, Feb. 2017, pp. 202–210, doi: 10.1145/3021460.3021485.

[13] “Google Android : CVE security vulnerabilities, versions and detailed reports.” https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224 (accessed Dec. 01, 2020).

[14] “Vulnerabilities and threats in mobile applications, 2019.” https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/ (accessed Dec. 01, 2020).

[15] “Top 7 Mobile Security Threats in 2020 | Kaspersky.” https://www.kaspersky.com/resource-center/threats/top-seven-mobile-security-

threats-smart-phones-tablets-and-mobile-internet-devices-what-the-future-has-in-store (accessed Dec. 01, 2020).

[16] R. P. Medina, E. B. Neundorfer, R. Chouchane, and A. Perez, “PRAST: Using Logic Bombs to Exploit the Android Permission Model and a Module Based Solution,” in *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*, Nantucket, MA, USA, Oct. 2018, pp. 1–8, doi: 10.1109/MALWARE.2018.8659369.

[17] H. Kang, J. Jang, A. Mohaisen, and H. K. Kim, “Detecting and Classifying Android Malware Using Static Analysis along with Creator Information,” *Int. J. Distrib. Sens. Netw.*, vol. 11, no. 6, p. 479174, Jun. 2015, doi: 10.1155/2015/479174.

[18] T. Li, M. Xu, X. Deng, and L. Shen, “Accelerate CTU Partition to Real Time for HEVC Encoding With Complexity Control,” *IEEE Trans. Image Process.*, vol. 29, pp. 7482–7496, 2020, doi: 10.1109/TIP.2020.3003730.

[19] T. Xia, “A Constant Time Complexity Spam Detection Algorithm for Boosting Throughput on Rule-Based Filtering Systems,” *IEEE Access*, vol. 8, pp. 82653–82661, 2020, doi: 10.1109/ACCESS.2020.2991328.

[20] S. McIlroy, N. Ali, and A. E. Hassan, “Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store,” *Empir. Softw. Eng.*, vol. 21, no. 3, pp. 1346–1370, Jun. 2016, doi: 10.1007/s10664-015-9388-2.

[21] K. Allix, Q. Jerome, T. F. Bissyande, J. Klein, R. State, and Y. L. Traon, “A Forensic Analysis of Android Malware -- How is Malware Written and How it Could Be Detected?,” in *2014 IEEE 38th Annual Computer Software and Applications Conference*, Vasteras, Sweden, Jul. 2014, pp. 384–393, doi: 10.1109/COMPSAC.2014.61.

[22] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, “‘Andromaly’: a behavioral malware detection framework for android devices,” *J. Intell. Inf. Syst.*, vol. 38, no. 1, pp. 161–190, Feb. 2012, doi: 10.1007/s10844-010-0148-x.

[23] D. Maier, T. Muller, and M. Protsenko, “Divide-and-Conquer: Why Android Malware Cannot Be Stopped,” in *2014 Ninth International Conference on Availability, Reliability and Security*, Fribourg, Switzerland, Sep. 2014, pp. 30–39, doi: 10.1109/ARES.2014.12.

[24] A. Scurtu, “Learning to predict whether an app will be kept or removed from the Play Store by Google,” bachelor, 2020.

[25] I. Almomani and M. Alenezi, “Android Application Security Scanning Process,” in *Telecommunication Systems - Principles and Applications of Wireless-Optical Technologies*, I. A. Alimi, P. P. Monteiro, and A. L. Teixeira, Eds. IntechOpen, 2019.

[26] G. He, B. Xu, and H. Zhu, “AppFA: A Novel Approach to Detect Malicious Android Applications on the Network,” *Secur. Commun. Netw.*, vol. 2018, pp. 1–15, Apr. 2018, doi: 10.1155/2018/2854728.

[27] H. Wang, H. Li, L. Li, Y. Guo, and G. Xu, “Why are Android apps removed from Google Play?: a large-scale empirical study,” in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR ’18*, Gothenburg, Sweden, 2018, pp. 231–242, doi: 10.1145/3196398.3196412.

[28] K. Riad and L. Ke, “RoughDroid: Operative Scheme for Functional Android Malware Detection,” *Secur. Commun. Netw.*, vol. 2018, pp. 1–10, Sep. 2018, doi: 10.1155/2018/8087303.

[29] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,”

presented at the Network and Distributed System Security Symposium, San Diego, CA, 2014, doi: 10.14722/ndss.2014.23247.

[30] Z. Yuan, Y. Lu, and Y. Xue, “Droiddetector: android malware characterization and detection using deep learning,” *Tsinghua Sci. Technol.*, vol. 21, no. 1, pp. 114–123, Feb. 2016, doi: 10.1109/TST.2016.7399288.

[31] T. Chen, Q. Mao, Y. Yang, M. Lv, and J. Zhu, “TinyDroid: A Lightweight and Efficient Model for Android Malware Detection and Classification,” *Mob. Inf. Syst.*, vol. 2018, pp. 1–9, Oct. 2018, doi: 10.1155/2018/4157156.

[32] “Dynalog: an automated dynamic analysis framework for characterizing android applications,” in *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, London, United Kingdom, Jun. 2016, pp. 1–8, doi: 10.1109/CyberSecPODS.2016.7502337.

[33] K. Xu, Y. Li, and R. H. Deng, “ICCDetector: ICC-Based Malware Detection on Android,” *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1252–1264, Jun. 2016, doi: 10.1109/TIFS.2016.2523912.

[34] K. A. Talha, D. I. Alper, and C. Aydin, “APK Auditor: Permission-based Android malware detection system,” *Digit. Investig.*, vol. 13, pp. 1–14, Jun. 2015, doi: 10.1016/j.diin.2015.01.001.

[35] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, “MalDozer: Automatic framework for android malware detection using deep learning,” *Digit. Investig.*, vol. 24, pp. S48–S59, Mar. 2018, doi: 10.1016/j.diin.2018.01.007.

[36] S. Hassan, C. Tantithamthavorn, C.-P. Bezemer, and A. E. Hassan, “Studying the dialogue between users and developers of free apps in the google play store,” in *Proceedings of the 40th International Conference on Software Engineering*, Gothenburg Sweden, May 2018, pp. 164–164, doi: 10.1145/3180155.3182523.

[37] W. Y. Lee, J. Saxe, and R. Harang, “SeqDroid: Obfuscated Android Malware Detection Using Stacked Convolutional and Recurrent Neural Networks,” in *Deep Learning Applications for Cyber Security*, M. Alazab and M. Tang, Eds. Cham: Springer International Publishing, 2019, pp. 197–210.

[38] K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, “DroidEvolver: Self-Evolving Android Malware Detection System,” in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, Stockholm, Sweden, Jun. 2019, pp. 47–62, doi: 10.1109/EuroSP.2019.00014.

[39] J. Xu, Y. Li, R. Deng, and K. Xu, “SDAC: A Slow-Aging Solution for Android Malware Detection Using Semantic Distance Based API Clustering,” *IEEE Trans. Dependable Secure Comput.*, pp. 1–1, 2020, doi: 10.1109/TDSC.2020.3005088.

[40] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, “DL-Droid: Deep learning based android malware detection using real devices,” *Comput. Secur.*, vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.

[41] “VirusTotal.” https://www.virustotal.com/gui/file/1bb5004c249e9175e3190b3299d339215600ae77441a28c3afe278b903a0c285/detection (accessed Dec. 28, 2020).

[42] “Firebase.” https://firebase.google.com/ (accessed Feb. 11, 2021).

[43] “Developer Policy Center.” https://play.google.com/about/developer-content-policy/ (accessed Dec. 28, 2020).