Computer Science Review 40 (2021) 100372

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review Article

Comparative analysis of Android and iOS from security viewpoint

Shivi Garg [a,b], Niyati Baliyan [b,*]

[a] Faculty of Computer Engineering, J.C. Bose University of Science and Technology, YMCA, Faridabad, India
[b] Information Technology Department, Indira Gandhi Delhi Technical University for Women, Delhi, India

ARTICLE INFO

Article history:
Received 5 May 2020
Accepted 23 January 2021
Available online 13 February 2021

Keywords:
Android
iOS
Malware
Security
Smartphones
Vulnerabilities

ABSTRACT

Smartphone usage has increased exponentially in the recent years. Android and iOS are the most popular smartphone platforms, while the ease of use along with the computational power to handle a wide array of applications attracts millions of users worldwide, also raises the security concerns on these platforms. This paper presents a comparative analysis between Android and iOS on a wide range of security aspects. It analyzes data for the period 2015-2019 and gives a detailed snapshot of not only the quantum of vulnerabilities, but also their impact. In addition, the paper leverages the well-established security triad i.e. CIA (Confidentiality, Integrity, Availability) to compare both the operating systems. The comprehensive and pragmatic approach taken in the paper makes it easier to infer that Android is more susceptible to security breaches and malware attacks as compared to iOS. Hence, researchers should divert their efforts and focus on finding solutions to problems pertaining to Android. The paper concludes by laying down future research directions and scope of work, which can be leveraged not only by application developers, but also by researchers. This will help make Android safer for users and will further increase its demand as a mobile operating system.

Contents

1. Introduction

Technological advancements in smartphones are at par with personal computers. With increased computing power, smartphones are becoming ubiquitous part of daily life. Hence, number of smartphone users has exponentially risen in the last five years. The fact is established with the help of Statista [1], where the

* Corresponding author.
E-mail addresses: shivi002phd16@igdtuw.ac.in (S. Garg), niyatibaliyan@igdtuw.ac.in (N. Baliyan).

https://doi.org/10.1016/j.cosrev.2021.100372
1574-0137/

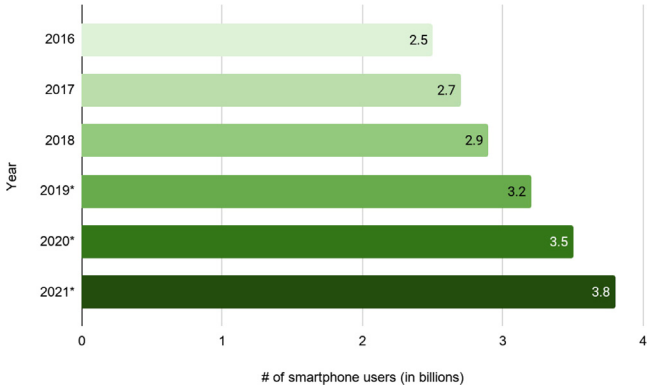

Fig. 1. Number of Smartphone users by year.

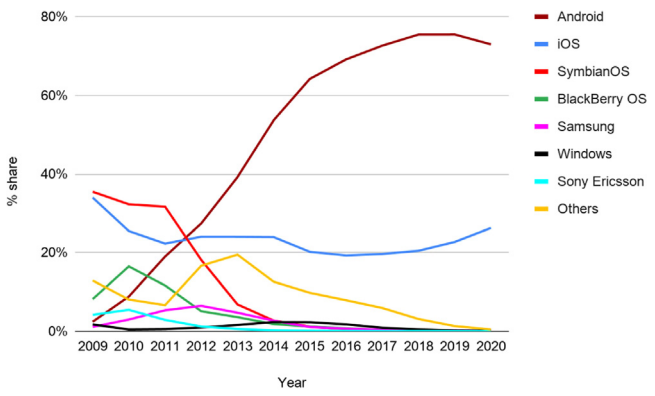

Fig. 2. Market share of different Mobile OS from 2009–2020.

world smartphone users has exceeded three billion and is predicted to further increase by several hundred million by 2021 as shown in Fig. 1.

There are multiple mobile platforms for smartphones, e.g. Symbian, Blackberry, Windows, Android, iOS, etc. Fig. 2 shows the market share of different mobile operating system (OS) from 2009–2020. Android (72.95%) and iOS (26.27%) are prominent due to their enhanced capabilities and popularity among users. Even though there are other smartphone platforms available, the combined market share of Google's Android and Apple's iOS forms 99% by the year 2020 [2], therefore this study focuses on Android and iOS.

It is also seen that Android is more popular than iOS. According to ZDNet [3], Google revealed that it has ∼2.5 billion active Android devices in 2019, making the largest mobile OS by user numbers. On the other hand, Apple has 1.4 billion users across all products, including macOS and iOS devices.

Fig. 3 depicts the statistics for the mobile application downloads from 2018 to 2024 worldwide, sorted by app store. In 2024, it is predicted that mobile users will download 139 billion mobile applications from the Google Play Store, up from 102 billion applications downloaded from Google Play in 2019 [4]. Aforementioned reasons account for Android's prominence in the smartphone marketplace.

The key contributions of this paper are:

1. Comparative analysis of Android and iOS based on architecture, security model, isolation mechanism, encryption mechanism, app permissions, and auto-erase mechanism
2. Listing of common vulnerabilities prevalent in both Android and iOS, their distribution pattern over the recent years, mean severity score and vulnerability assessment based on confidentiality, integrity and availability triad
3. Thorough investigation of strengths and weaknesses of related works are also discussed
4. Discussion on various research challenges along with future work presented for academicians and application developers in order to cope with increasing vulnerabilities

The paper is structured as follows: Section 2 discusses the approaches for the smartphone security from the past with their strengths and weaknesses. Section 3 presents the comparative analysis of Android and iOS based on architecture, security model, isolation mechanism, encryption mechanism, app permissions, and auto-erase mechanism. Section 4 describes common vulnerabilities prevalent in Android and iOS. Section 5 discusses the data collection methodology along with the preprocessing. Section 6 presents the vulnerability trends of Android and iOS over the years, mean severity score and vulnerability assessment on confidentiality, integrity and availability triad. In Section 7, malware attacks on Android and iOS are discussed in the time period of 2015–2019. Furthermore, Section 8 explains future directions that should be adopted by the researchers and app developers to avoid growing hazards. Finally, the conclusion is provided in Section 9 of the paper.

2. Related work

Research work carried out in the domain of smartphone security deals with issues like vulnerabilities in smartphones, malware impact, security procedures in mobile platforms, and different policies used for safety and privacy of users.

Hidhaya and Geetha [5] detected web view vulnerabilities, which can cause malicious attacks on smartphones. They used static analysis techniques to analyze the apps. They achieved a detection accuracy of 85%, however, run time/dynamic behavior of the apps was not analyzed. Zhuang [6] studied smartphone security vulnerabilities based on the apps using SSL protocol for secure communication, WebView technology, and HTML5-based. Author was able to demonstrate SSL validation, WebView code injection and HTML5-based application attacks in an accurate and explicit way. However, the work was not very comprehensive since they used only function to analyze SSL vulnerabilities. Ahvanooey et al. [7] surveyed on smartphone security and studied the attacks on mobile applications from 2011–2017. The survey was not very comprehensive and does not provide a clear idea of the mobile platform vulnerabilities. Talal et al. [8] work comprehensively illustrated the smartphone security in four different categories — mobile security aspects, solutions related to smartphone security, studies related to threats posed by smartphone malware and malware classification into families. The first category reviewed survey articles related to smartphone security. The second category discussed articles on solutions related to smartphone security. Malware related studies and threats posed by malware are included in the third category, and lastly the ranking and classification of malware into families is discussed in the fourth category.

There are different concerns related to Android and iOS security. In [9] authors introduced a Tap-Wave-Rub, which is a lightweight permission enforcement approach for smartphone malware prevention. The approach was based on two mechanisms — acceleration-based phone tapping detection; and proximity-based finger tapping, rubbing or hand waving detection. This approach proved to be effective with low false positive rate for malware detection. Li and Ju [10] assessed malicious applications in Android using permission sets. They estimated the required permissions of the applications and performed Maximum Severity Rating (MSR) classification of the applications.

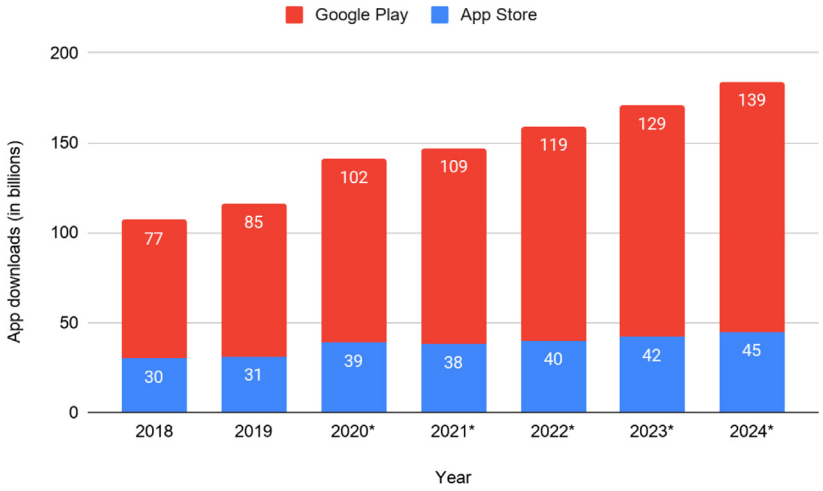

Fig. 3. Mobile application downloads from 2018 to 2024 by store worldwide.

Then, MSR classification was analyzed based on the user experiences. These steps were then combined into an application installation procedure. The discussed approaches fail to enforce access control mechanisms to access a resource or data in the permission sets, thereby bypassing the permissions once the root privilege is obtained. The customized Linux kernel of Android can make it vulnerable when the root privileges are obtained. Code-signing method is used by Android, which contains the self-signed certificates of developers without verifying the certification authority [11]. Cho and Seo [12] demonstrated a method to mount a malware by exploiting the code-signing process used in Android. They also proposed a countermeasure to prevent this attack. Due to this, repackaged malware can be easily distributed.

The security mechanisms in iOS are stronger as compared to Android. However, researches in the past have shown how iOS is vulnerable to malware attacks. The studies [13–15] revealed that when SSH server is actively running, iOS can be easily jailbroken. Teufl et al. [14] also analyzed that backup stored on the iTunes can be easily attacked by using different techniques. Brute-force attack can be performed on the encrypted backup. If the backup is not encrypted, the attacker can gain access and manipulate with the backup files. Zdziarski [16] also detected various attack points, back doors, and surveillance mechanisms in iOS devices. He identified that iOS can be easily accessed via Universal Serial Bus (USB), Wi-fi or when paired with the compromised system. An attacker can target the iOS device on a Wi-Fi network by scanning TCP:62078 and can authenticate this pairing record. Moreover, the attacker can steal sensitive user information when a trusted relationship between a system and iOS device is established.

The discussed approaches are not comprehensive enough to present all the security aspects of Android and iOS. Most of the research articles lack the holistic view of smartphone vulnerabilities and impact in terms of confidentiality, integrity and availability. This study is first of its kind, which covers all the security aspects of mobile platforms.

3. Comparison between Android and iOS

Android and iOS are the most popular mobile platforms available in the market. The on-going Android vs. iOS battle seems to be never ending. Section 3 elaborates this in terms of different parameters, with more focus on security aspects.

3.1. System architecture

Android is a Linux-based mobile OS with different layers, namely, kernel, hardware abstraction layer, Android runtime, libraries, application framework and applications as shown in Fig. 4. The application architecture of Android is designed in such a way that it simplifies the reuse of components. Modular system components and services allow the applications to use each other's capabilities. It also allows components to be replaced by the user [17]. Functionality of these layers is described as follows:

Kernel — It supports and manages core system services like process, memory, security, network, etc.

Hardware Abstraction Layer (HAL) — It acts as an interface for communicating the Android application/framework with hardware-specific device drivers such as camera, Bluetooth, etc. HAL is hardware-specific and implementation varies from vendor to vendor.

Android Runtime (ART) — ART is introduced as a new runtime environment in newer Android versions (version 5.0 onwards). During app installation, it uses ahead-of-time (AOT) and just-in-time (JIT) compilation, which compiles the Dalvik bytecode into native binaries (ELF format). This optimizes garbage collection and power assumption and achieves high runtime performance.

Native Libraries — Core system services and different components of Android like ART and HAL are built from the native libraries, which are written in C/C++. There are different libraries, which provide support in building user interface application framework, drawing graphics and accessing database.

Application framework — Android SDK provides tool and API libraries to develop applications on Android java. This framework is known as Android Application Framework. Important features are database for storing data, support for audio, video and image formats, debugging tools, etc.

System applications — Applications are located at the top most layer of the Android stack. These consist of both native and third-party applications such as web browser, email, SMS messenger, etc., which are installed by the user.

The iOS is OS X based mobile OS, which is a variant of a BSD UNIX kernel running on top of a micro kernel called Mach. iOS has a layered architecture. There is no direct communication since it has an intermediate layer between the applications and the hardware. Basics services in iOS are handled by the lower layers, namely, Core Services layer and Core OS layer, whereas upper layers like Media layer and Cocoa Touch layer provide the user interface and sophisticated graphics [18]. The layered architecture

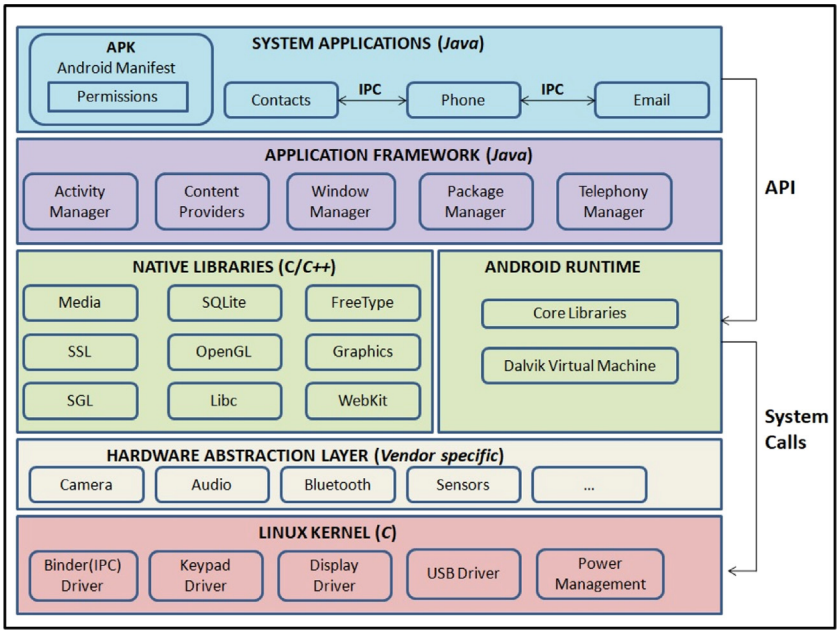

Fig. 4. Android architecture.

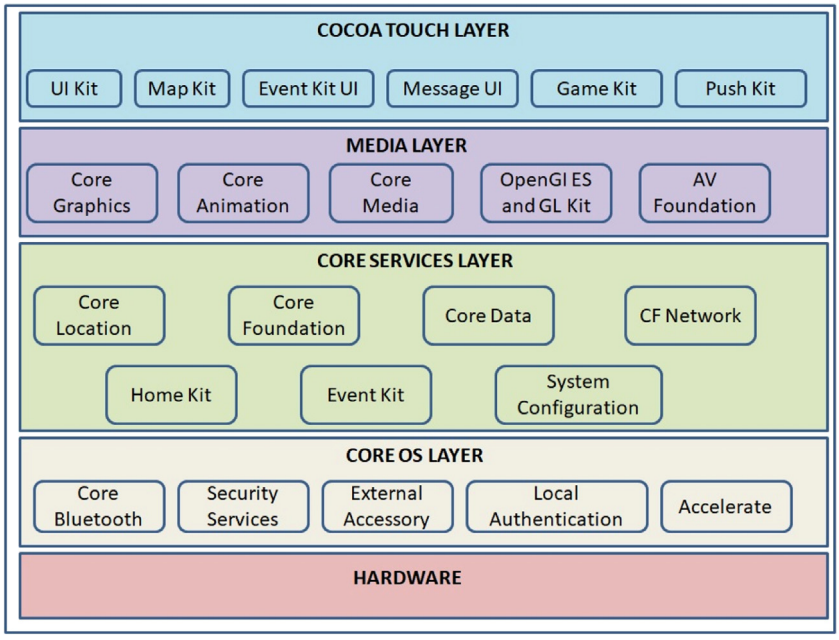

Fig. 5. iOS architecture.

of iOS is shown in Fig. 5. The different layers of iOS architecture are described as follows:

Hardware — This layer contains the physical chips, which are soldered to the iOS circuitry.

Core OS Layer — It is the bottommost layer, which interacts directly with the hardware. It has the operating system above which the other layers reside. This layer takes care of memory management (allocation and de-allocation once the application has finished using it), file management, network management, etc.

Core Services Layer — This forms the foundation layer on which above layers are built. It provides several features like data protection, iCloud storage, file sharing support, XML Support features, SQLite database, In-App purchases, etc.

Media Layer — This layer is responsible for graphics, audio and video capabilities. Media layer consists of three different frameworks — Graphic framework, Audio framework and video framework. These frameworks help in accessing photos and videos stored on the device, to manipulate the images through filters and provide support for 2D drawings.

Cocoa Touch Layer — It provides key frameworks for building iOS apps and defines their appearance. This layer is responsible for fundamental technologies like multitasking, touch-based input, push notifications, and many high-level system services.

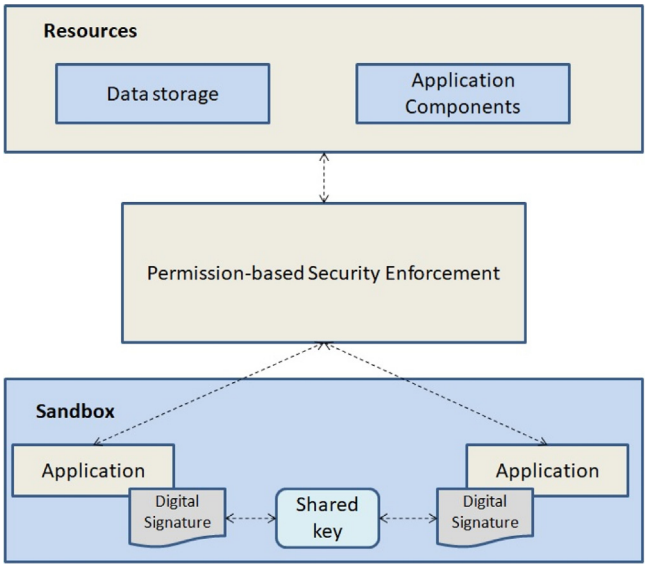

Fig. 6. Android security model.

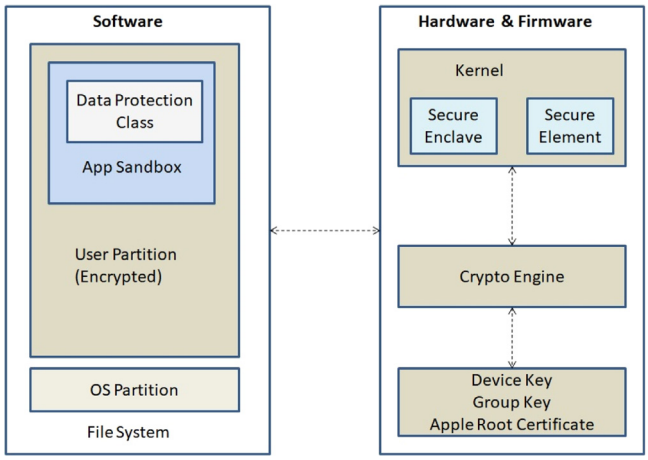

Fig. 7. iOS security model.

3.2. Security

Android is an open-source mobile OS, i.e., its code is available to all and can be used by anyone. Linux kernel in Android provides user protection, where it restricts the users to access system resources and prevents them from exhausting. This mechanism of user protection helps in creating Application Sandbox, where each Android application is assigned with unique user ID [19]. The processes are run in a separate space, thereby having user permission access control as shown in Fig. 6.

iOS security model is more restrictive as compared to Android. iOS is a closed system, where developers can develop their own apps but its source code is not released, as in case of Android [20]. Device level security includes a device-level locking mechanisms like passcode or Personal Identification Number (PIN) and remote wipe using Mobile Device Management (MDM). System-level security implements a secure boot chain process, Secure Enclave, Touch ID and authorizes system software updates. Data level security encompasses file encryption methodology through data-protection classes and using hardware and software components. Another level of comparison is that iOS is not easy to jailbreak. This means both the hardware and software is controlled by Apple. Data encryption in iOS is not configurable; therefore it is not disabled by the users. Fig. 7 illustrates the security model of iOS.

3.3. Isolation mechanism

Sandbox environment is provided for each and every application executing in the device, i.e., every application executes in its own environment. In this way they cannot modify other application.

In Android, applications are separated from each other and from the system's kernel; however, permissions are given to the applications to access the system resources. Isolation mechanism does not allow to access resources beyond the approved permissions. Isolation from kernel prevents gaining access to the administrative control. As a result, attacker is not able to compromise with other applications running on the Android. However, an application can neither launch other applications on the system using certain permissions nor they can check the programming logic of other applications [21]. An app can launch various types of attacks and can access some of the device's subsystems, depending on the set of permissions given.

iOS isolation mechanism is similar to Android. Applications running in iOS do not need user's permissions to access system resources. Application sandboxing enforced in iOS architecture isolates all applications from each other and restrains the third party applications from accessing iOS kernel [22]. As a result, applications can neither interact with each other nor they know about the kernel [23]. This makes the iOS systems more secured as compared to Android.

3.4. Encryption mechanism

Data on the mobile devices can be easily lost or stolen. Therefore, data stored on the mobile devices should be encrypted to ensure safety and security.

Previous versions of Android provide Full Disk Encryption (FDE). Recent Android versions support hardware encryption, also known as Trusted Execution Environment (TEE) and File-based Encryption (FBE). In FBE, different files are encrypted with different keys so that they can be unlocked independently [24], as shown in Fig. 8.

In iOS, combinations of hardware and software keys are iterated to encrypt every file to ensure only offline brute-force attack can carry out. Data protection class is associated with every file, which allows developers to define under what circumstances the data is accessible [25], as shown in Fig. 9.

3.5. App permissions

Permissions control and regulate the access to device resources, once an app is downloaded on the device. In most of the cases users are not aware about the permissions which applications are accessing, thereby causing serious security concerns.

In Android, list of all permissions which an app needs are shown to the users while installing the app. User can then decide whether that app needs that particular permission or not. In some cases, if the user denies certain permissions to an app, then it stops installing. Attackers can misuse the permissions in Android. They can force the users to allow certain permissions, which can steal their sensitive information. Different types of attacks can be performed using permissions, namely, data loss attacks, data integrity attacks, Denial of Service (DoS) and Distributed DoS (DDoS) attacks. The permission system in Android is not effective since security decisions are handled by the device users. This may compromise with the security aspect of Android as majority

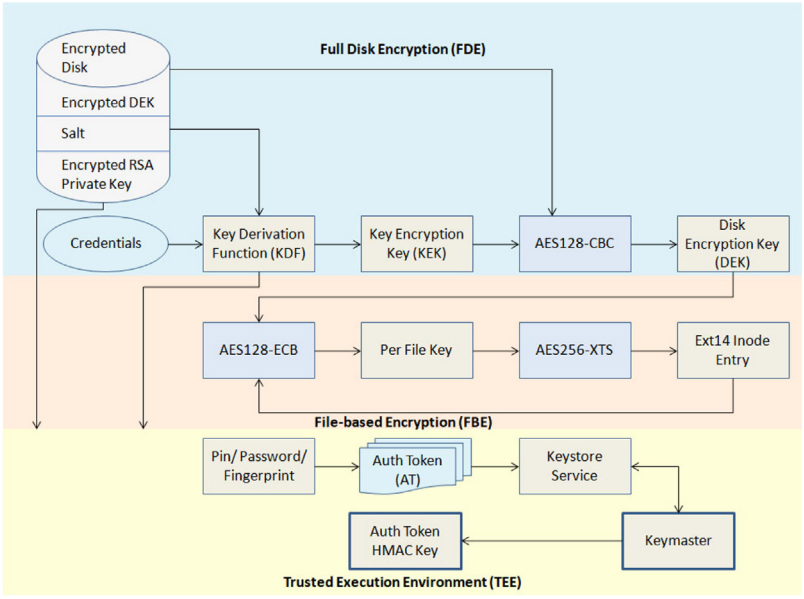

Fig. 8. Encryption model used in Android.

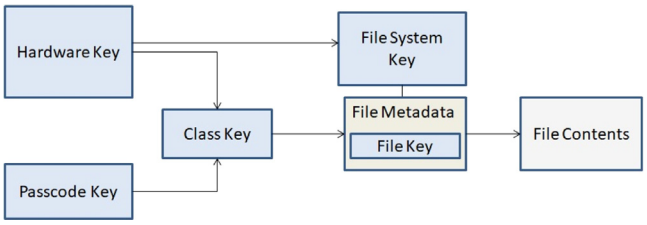

Fig. 9. Encryption model used in iOS.

of the users are not capable enough to take correct security decisions [26].

In iOS, permissions that an app needs to access are not shown to the users. All the apps are isolated from each other in Apple, thereby blocking access to most of the device's sensitive subsystems. Users are not responsible for making any kind of security decisions. This isolation policy used by iOS grants the permissions, which an app requires without users' consent [27]. However, in some cases users can grant permissions to access the device's resources like initiating an outgoing phone call, sending an email message or an outgoing SMS, accessing location using GPS, receiving push notifications from the Internet, etc.

3.6. Auto erase mechanism

Smartphone can carry huge sensitive and private information. Attackers are capable of causing fraud and identity thefts. Auto-erase [28] feature is helpful when the smartphone is lost or stolen. Personal and sensitive information can be wiped off from the smartphone using auto erase feature.

Android native apps do not provide auto erase mechanism, however third-party applications can be used for auto erase.

In iOS, if an attacker tries to attempt failed passcode attempts ten times, all the data on the device is automatically erased using this feature.

3.7. Application provenance

Application provenance [29] is defined as a process in which third-party applications are analyzed for their intended functionality and security. Every mobile platform vendor has to validate the authenticity and provenance before uploading the app into the app store. Once the app is verified, author stamps it with the identity, which is then signed digitally with a vendor-issued digital certificate to make it tamper resistant.

In Android, app developers are not required to register with the Google play store. They do not require Google-issued signing certificates. Android App developers can easily upload their apps on the play store without being monitored by Google. Google asks the app developers to pay a fee of $25 using credit card to get their apps certified. This process is used by Google to link an app with developer's digital certificate. There is no assurance of this linking since the attackers can use stolen credit card to pay the fee. Another security issue is that Google play store is not the single place to distribute the apps designed by the developers. Android app developers can distribute their app to third party, where these apps are not verified by the Google, thereby making the system more vulnerable.

Unlike Android, iOS developers can distribute their apps only on official app market place call App Store. iOS developers are first required to register with Apple to distribute their apps via app store. Apple follows a vetting process. It is a licensing agreement where each app submitted by the developer, is verified for its security and privacy. The app is digitally signed and then published on the app store, if it does not violate the licensing agreement. The digital signing process guarantees that both the app and its developers are not tampered by a third-party distributor or download site. The vetting process prevents the hackers from uploading malicious apps in the App Store or attacking published app. App developers are required to provide their identities while registering with Apple and obtaining a signing certificate. This acts as another defensive measure against any malicious activity. Table 1 summarizes the differences between Android and iOS.

Apart from aforementioned parameters there are other security concerns. Android lacks the control over device manufacturers. Android has high device manufactures count; therefore, it is difficult for Google to lay common set of protocols for its partnered OEMs. Apple has much more concentrated control over device manufactures. Android has high device fragmentation, meaning both the number of device variants and OS versions are very high, which provides an ecosystem for security breaches.

Table 1
Summary of differences between Android and iOS.

Feature	Sub feature	Android	iOS
Source model		Open-source	Closed, but iOS components are open source
Architecture	Kernel	Linux	OS X, UNIX
	Language	Dalvik (Java)	Objective C
	Layers	Kernel — management of core system services — process, memory, security, network HAL— interface for communicating the Android application/ framework with hardware-specific device drivers such as camera, Bluetooth, etc. Libraries — helps in building user interface, graphics drawing and database access Application framework — features are database for storing data, support for audio, video and image formats, debugging tools Applications — native and third-party applications such as web browser, email, SMS messenger, etc., which are installed by the user.	Hardware — contains the physical chips Core OS — layer takes care of memory management (allocation and de-allocation once the application has finished using it), file management, network management, etc Core services — provides several features like data protection, iCloud storage, file sharing support, XML Support features, SQLite database, In-App purchases, etc. Media — responsible for graphics, audio and video capabilities Cocoa touch — provides key frameworks for building iOS apps and defining their appearance.
Security	Application isolation	Individual sandbox for each app with user's permission to access system resources	Shared sandbox for all apps; no permission required from the users
	Encryption	Previous versions support FDE, later versions support TEE and FBE	Hardware encryption + Data protection class
	App permissions	Shown to the users	Not shown to the users
	Auto erase	No	Yes
Application provenance	App distribution Vetting process Digital signature	Google play store + third party app markets Partial Yes	Official App store Yes Yes

Table 2
Common vulnerabilities in Android and iOS.

Vulnerability	Description
Gain information	This vulnerability exposes sensitive information to the unauthorized attackers. Attackers can gain information using malicious scripts in the applications.
Gain privileges	It can occur when an attacker gains root or administrative rights, as a result of which normal security checks by OS are disabled.
Bypass something	This vulnerability occurs when attackers can evade authentication mechanisms. Attackers can access unprotected file and can attack protected applications by evading the authentication system.
Overflow	This vulnerability occurs when the buffer is overwritten by extra data, which is inserted by some malicious script. It can lead to serious crashes in the system, which can damage files and information.
Memory corruption	Memory corruption vulnerability occurs when software tries to read/ write to memory location, which is outside the bounded buffer. As a result of this, attacker can access sensitive and private information and can alter the control flow.
Denial of Service (DoS)	Attackers can exploit this vulnerability by making the resources unavailable to the legitimate users. Improper handling of the resources like memory, file, and database storage can result in denial of service.
Code execution	Malicious code can be implanted in the user's input, which can execute arbitrary code. Arbitrary code can then alter the control flow of software, thereby changing or deleting the important data.
SQL Injection	Attacker inserts controlled data in the SQL query, which can alter the database, access the sensitive information or can bypass the security checks in the system. SQL injection is commonly seen in database driven websites.
Cross site scripting (XSS)	XSS vulnerability occurs when a malicious data is inserted in the web application via a web request. The malicious script can change the HTML content of a web page, access tokens of the sessions, cookies, or any other sensitive information used by the browser.
Directory traversal	Directory traversal or path traversal vulnerability occurs when the attacker constructs a pathname using a controlled input to access directory or file located outside the restricted directory. As a result, the attacker is able to read arbitrary files on the target system.
HTTP response splitting	This vulnerability arises when the data from the HTTP request enters a web application. HTTP requests may contain CRLF (carriage return (\r) and line feed (\n)) characters, which are inserted in the HTTP response header and sent to the web user without validating for malicious characters.

4. Software vulnerabilities common in Android and iOS

Vulnerability can be defined as a weakness, which an attacker can exploit potentially to carry out unauthorized actions in a network or a system [30]. Vulnerabilities in mobile devices can arise due to lack of security practices by users and from poor technical controls. Software vulnerabilities are analyzed based on different technical parameters like causes, techniques, severity levels and software systems [31]. Common vulnerabilities seen in Android and iOS are shown in Table 2.

5. Data collection

The main source of data for this study is CVE details [32]. Web-based scraper tool Web Scraper 0.4.0 is used to collect this data. The web scraper looks for the CVE IDs for vulnerabilities and extracts corresponding details. Web-based scraper was able to extract all vulnerabilities in the iOS and Android category from CVE details. We collected 1655 CVE IDs of iOS and 2563 CVE IDs of Android, with a total of 4218 CVE IDs. Fig. 10 explains the process of web-based scraping. After scraping the data, data is preprocessed, where the unnecessary data elements (number of exploits, update date, etc.) are removed.

6. Vulnerability trends in Android vs. iOS

Smartphones are prone to cyber-attacks and can compromise the confidentiality, integrity and availability of the data saved on the mobile devices. Risk levels are indicated by the number of vulnerabilities and number of reported malware. The number of vulnerabilities determines the flaws found in the platform that

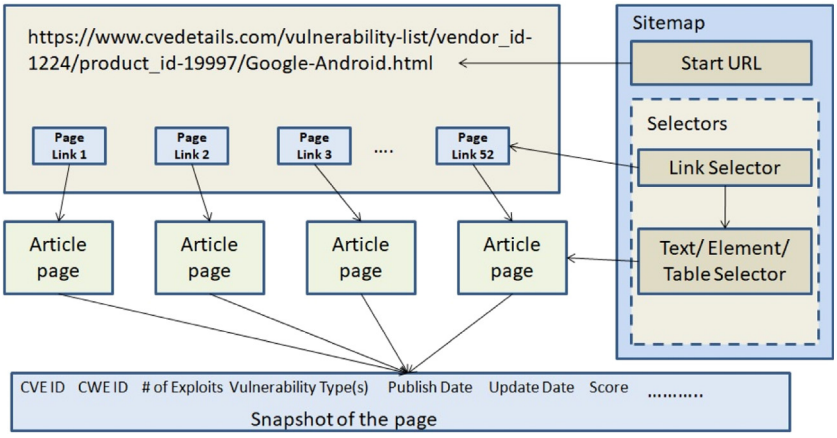

Fig. 10. Snapshot of a web-based scraping process.

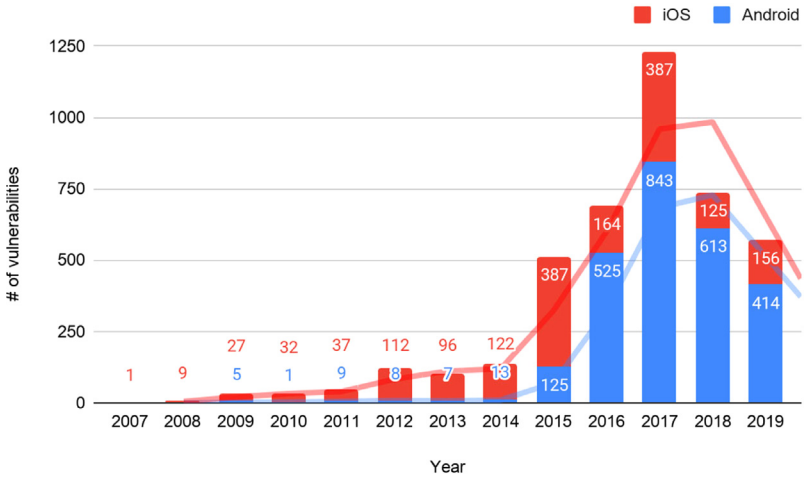

Fig. 11. Trend of Android & iOS vulnerabilities for 2007–2019.

can potentially compromise it, whereas the number of malware indicates the number of actual threats that are detected.

According to CVE details, the number of vulnerabilities reported for Android and iOS from 2000 to 2019 are 2563 and 1655 respectively. It is seen in Fig. 11 that there is a continuous increase in the number of vulnerabilities till 2017 and later on there is a steep decrease in the vulnerabilities in the year 2018 and 2019 for both Android and iOS.

It is also seen that percentage distribution of vulnerabilities for both Android and iOS is varying significantly as shown in Fig. 12. The percentage distribution of iOS vulnerabilities has decreased (from 100 to 27%) over the duration 20117–2019, whereas the trend is opposite for Android. Percentage distribution of Android vulnerabilities has considerably increased from 2014 to 2016 and then dipped in 2017 and again risen in 2018–2019. Before 2014, iOS was popular among the smartphone users but Android gained its popularity 2014 onwards, thereby increasing the number of vulnerabilities. So, we would be comparing Android and iOS 2015 onwards when Android came into the picture.

This decrease in the number of vulnerabilities is due to the better detection rates using ML and DL algorithms. Table 3 shows detection rates of ML/DL algorithms 2016 onwards.

It is seen that Android is dominated by 61% of vulnerabilities while in iOS it is 39%. Most of the major vulnerabilities like gain information, code execution, DoS, overflow and gain privilege are common in Android, while in iOS bypass and overflow memory corruption constitute large share. Table 3 shows the vulnerability types distribution between Android and iOS.

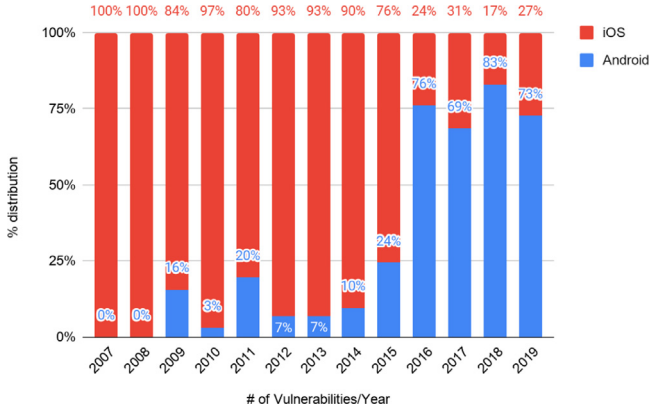

Fig. 12. Distribution of Android & iOS vulnerabilities (in %) from 2007–2019.

On analyzing the year wise vulnerability distribution of Android and iOS (2015–2019), it is evident that there is rise in Android vulnerabilities like unidentified vulnerabilities in Android have increased from 4% to 55% and DoS (3% to 7%), whereas most of the iOS vulnerabilities have decreased. Stacked area graph is plotted for both Android and iOS as shown in Figs. 13 and 14 respectively. A 100% stacked area graph shows how the single vulnerability of all the vulnerabilities has changed over the years. The *y*-axis scale is always 100%. Each colored area of represents a

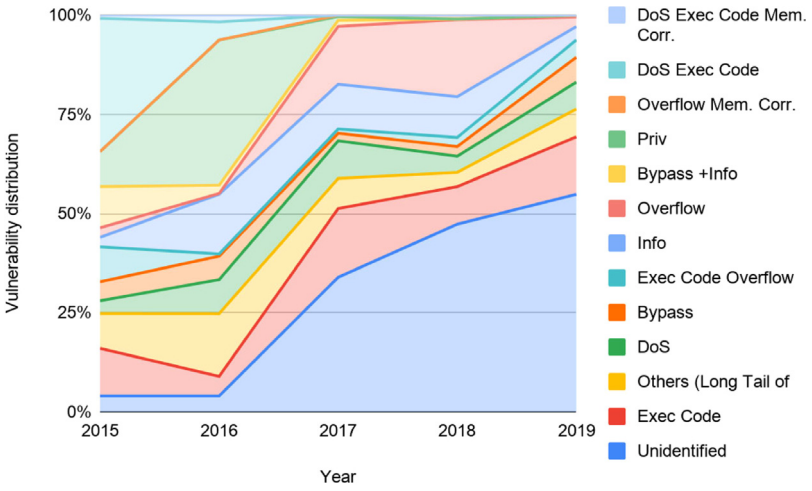

Fig. 13. Year wise vulnerability distribution for Android.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

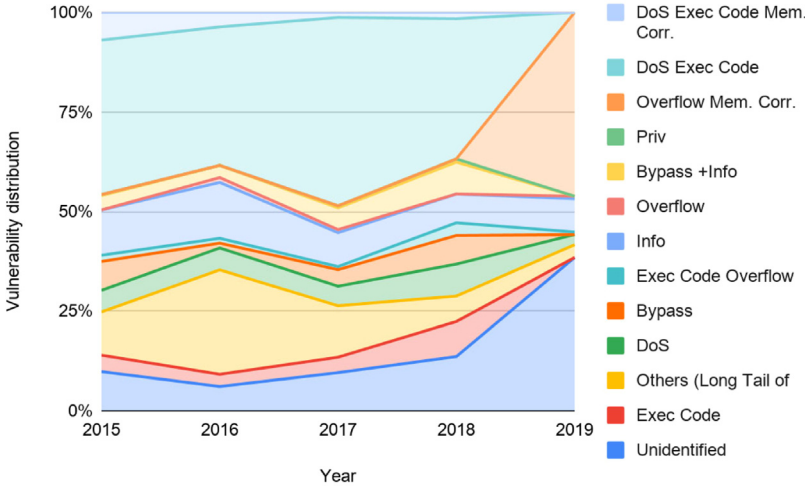

Fig. 14. Year wise vulnerability distribution for iOS.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Distribution of Android & iOS vulnerability types.

Vulnerability type(s)	Android	iOS	Total	Android %	iOS %
Unidentified	836	212	1048	80%	20%
DoS exec code overflow Mem. Corr.	66	561	627	11%	89%
Info	259	181	440	59%	41%
Exec code	310	56	366	85%	15%
DoS	191	95	286	67%	33%
Overflow	256	7	263	97%	3%
Priv	215	9	224	96%	4%
Bypass	97	105	202	48%	52%
Bypass +Info	40	55	95	42%	58%
Exec code overflow	56	26	82	68%	32%
Overflow Mem. Corr.	9	72	81	11%	89%
DoS exec code Mem. Corr.	10	66	76	13%	87%
Others	218	210	428	51%	49%
Total	2563	1655	4218	61%	39%

unique vulnerability. The parts are stacked up, usually vertically. The height of each colored stack represents the percentage proportion of that vulnerability at a given point in year. We have sorted vulnerabilities according to the starting year for Android; however, this is not applicable for iOS stack area graph so as to maintain same color coding and order. Table 4 shows the increase and decrease for both Android and iOS.

On analyzing the mean severity scores of Android and iOS, it is evident that Android vulnerabilities are more severe (mean score 6.9) as compared to iOS vulnerabilities (mean score 6.2). Fig. 15 shows the mean severity score of both Android and iOS from 2015–2019.

Another level of comparison is the access level. Access level is defined as how the vulnerability is exploited. Access levels can be local, remote or local network. Table 5 shows the percentage distribution of different access levels of the vulnerabilities in both Android and iOS. It is evident that 25% vulnerabilities in Android are exploited locally, while iOS vulnerabilities are 85% exploited remotely.

Access complexity is another parameter for comparing Android and iOS vulnerabilities. Access complexity measures the complexity of the attack, which is required to exploit the vulnerability once an attacker has gained access to the target system. Access complexity can be low, medium and high. Low value means that specialized access conditions or extenuating circumstances do not exist, medium specifies the access conditions are somewhat specialized and high means specialized access conditions exist. Table 6 shows the access complexity levels for both

Table 4
Distribution of Android & iOS vulnerability types.

Vulnerability type(s)	Android	iOS	Remarks
Unidentified	4% to 55% ↑	10% to 38% ↑	• Open source nature of Android • Jailbreaking in iOS
Exec code	12% to 14% ↑	4% to 0% ↓	• Buffer overflow in OpenJPEG 2.1.1 in Android • Use-after-free issue addressed with improved memory management in iOS
Others	9% to 7% ↓	11% to 3% ↓	• Regular software upgrades in Android and iOS
DoS	3% to 7% ↑	5% to 3% ↓	• Unhandled Java-level Null Pointer Exceptions (NPEs) in Android • Frequent software upgrades in iOS
Bypass	5% to 6% ↑	7% to 0% ↓	• Weak security policies in Android • Improved checks in iOS
Exec code overflow	9% to 4% ↓	2% to 1% ↓	• Isolation mechanism improved in Android • Improved input validation in iOS
Info	2% to 3% ↑	11% to 8% ↓	• Crafted libstagefright file in Media server applications in Android • Stack overflow addressed with improved input validation in iOS
Overflow	↔	0% to 1% ↑	• Small allocated stacks in iOS
Bypass +Info	10% to 0% ↓	4% to 0% ↓	• Better encryption mechanisms in Android • Improved checks in iOS
Priv	9% to 0% ↓	↔	• Better encryption mechanisms in Android
Overflow Mem. Corr.	↔	0% to 46% ↑	• Jailbreaking, small allocated stacks iOS
DoS exec code overflow Mem. Corr.	34% to 0% ↓	39% to 0% ↓	• Improved validation, improved memory management, improved checks in Android and iOS
DoS exec code Mem. Corr.	1% to 0% ↓	7% to 0% ↓	• Improved input validation, improved memory management in Android and iOS

↑ Increase, ↓ Decrease, ↔ Constant.

Table 5
Access levels of Android and iOS vulnerabilities.

OS	Local (%)	Local network (%)	Remote (%)
Android	25	2	73
iOS	14	1	85

Table 6
Access complexity of Android and iOS vulnerabilities.

OS	High (%)	Low (%)	Medium (%)
Android	5	47	49
iOS	2	26	71

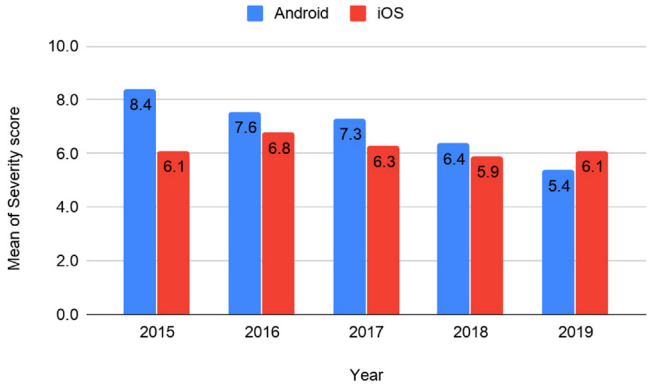

Fig. 15. Mean severity scores of Android & iOS vulnerabilities.

Android and iOS. It is evident that 71% of iOS vulnerabilities are medium-level complex while in Android, access complexity lies in the extremes-high (5%) and low (47%).

The confidentiality, integrity and availability (CIA) triad [33] is the core foundation of information security. Security vulnerabilities can have serious impact on CIA triad. For a comprehensive and complete security program, CIA triad must be addressed adequately.

Confidentiality — It ensures that information should not be accessible to unauthorized users, applications or processes. Cryptography, passwords, PIN numbers, etc. can be used to keep the information confidential. Password attacks, ping sweeps, phishing, key-logging, packet sniffing, etc. are some of the common attacks on confidentiality.

Integrity — It ensures protection of information from any unauthorized alteration. Only the authorized parties have the ability to modify the information once. Cyclic Redundancy Check (CRC) provides integrity protection against accidental or non-malicious errors. Man-in-the-middle attack, session hijacking, salami attack, trust relationship attack, etc. are included in the integrity attacks.

Availability — It guarantees that information should be accessible to authorized users only. DoS and DDoS attacks, SYN flood attack, etc. are most common availability attacks.

CIA triad is the backbone of information security system; therefore, it is important to understand the impact of vulnerabilities on CIA. Table 7 shows the impact of vulnerability scores on confidentiality. It is seen that 48% vulnerabilities in Android have complete impact on confidentiality, whereas in iOS it is only 22%. iOS vulnerabilities either have partial (60%) or none (18%) impact on confidentiality levels.

Table 8 shows the impact of vulnerability scores on integrity. It is seen that 46% vulnerabilities in Android have a complete impact on integrity, whereas 51% vulnerabilities in iOS have partial impact on integrity.

Table 9 shows the impact of vulnerability scores on availability. It is seen that 52% vulnerabilities in Android have a complete impact on availability, whereas 45%vulnerabilities in iOS account for partial impact on availability.

Table 7
Impact score on confidentiality.

Complexity	Complete (%)	None (%)	Partial (%)
Android	48	11	40
iOS	22	18	60

Table 8
Impact score on integrity.

Complexity	Complete (%)	None (%)	Partial (%)
Android	46	31	22
iOS	22	27	51

Table 9
Impact score on availability.

Complexity	Complete (%)	None (%)	Partial (%)
Android	52	24	24
iOS	24	31	45

7. Malware attacks in Android and iOS

Malware is malicious software that is designed to cause unintentional harm to the system [34]. Malware can evade security mechanisms, collect sensitive user information, display unnecessary advertisements, or can interrupt with the normal functioning of the mobile device. Different types of mobile malware are the Trojans, backdoors, ransomware, spyware, adware, etc. These malware are briefly described as follows:

Trojan — It is a type of malware, which does not self-replicate. To avoid detection, Trojans are generally masked as legitimate software. Trojan can enter into the system via numerous vectors, such as navigating untrusted websites (drive-by-download), by clicking attachments in phishing emails, other forms of social engineering, etc [35]. Trojan can steal sensitive user information, or can alter the data.

Ransomware — It is type of malicious software, which restricts the access to the system resources and extorts money from the victims [36]. Most popular ransomware in mobile platform is crypto-ransomware, which encrypts files of a system thereby restricting the users to access the files. Attackers then demand ransom for the key used to decrypt the files so as to resume the access.

Backdoor — It is a malware that evades the authentication mechanism of the system. As a result, it can remotely access the database and file systems. Backdoor installation requires administrator privileges by rooting Android devices and jailbreaking Apple devices [37].

Spyware — Spyware is unwanted software that infiltrates your computing device, stealing your internet usage data and sensitive information. Spyware is classified as a type of malware — malicious software designed to gain access to or damage your computer, often without your knowledge. Spyware gathers your personal information and relays it to advertisers, data firms, or external users. Spyware infection in the mobile device can be viewed differently in Android and iOS. In Android, social engineering attack can used to trick the users to download an app from the Google play store or from a third-party app store. Man-in-the-Middle (MitM) attack is used to perform remote infection, where the attacker intercepts the user's mobile communications to inject the malware. In iOS, spyware needs a physical access to the device or through exploitation of JailbreakME exploit [38].

Adware — Adware displays unwanted advertisements, which can alter the browser settings, steal personal information or can execute an arbitrary code [39].

Fig. 16 shows the timeline of Android and iOS malware from 2015–2019. The description of these malware is listed in Table 10.

Fig. 17 depicts the growth of Android malware worldwide as of May 2019 [40]. It is seen that the total number of Android malware detections amounted to over 10.5 million programs by May, 2019. There is a steep decline in the growth of malware from 2018 to 2019 due to the deployment of effective machine learning and deep learning malware detection techniques.

8. Research directions and future scope

Android's market share will continue to grow and it is evident that there would be no cross over between Android and iOS in the near future. Increasing market share and open-source nature of Android owes to its increasing vulnerabilities as compared to the closed nature of iOS. Therefore, we have chosen Android as our research area, where we will address the major problems encountered in Android. With the growing vulnerabilities, Android malware is continuously emerging at an alarming rate. This calls for more efficient malware detection techniques with less human intervention. We have proposed a novel parallel classifier scheme for malware detection in Android [41], which achieved an accuracy of 98.27%. Data related to this study can be found in [42]. Detected malware are then classified into 71 malware families using ensemble classifiers [43]. Last part of the research includes

Table 10
Malware description.

Year	Targeted OS	Malware	Malware type	Description
2015	Android	Acecard	Trojan	A type of Trojan Banker, which steals banking information of the user
2015	Android	AdDown	Adware	Shows ads to infected users, collects personal data on its victims, and secretly installs apps without the user's knowledge
2015	iOS	XcodeGhost	Trojan	Collects information on the device and upload it to the C2 servers
2015	iOS	YiSpecter	Adware	Attacks both jailbroken and non-jailbroken iOS devices through unique and harmful malicious behaviors
2016	Android	Triada	Backdoor	Evades anti-virus by executing DroidPlugin open-source sandbox to hide malicious Android application package (APK) plugins in its asset directory
2016	Android	Switcher	Trojan	It hacks the WiFi router, when the device is connected to a WiFi network
2016	iOS	AceDeceiver	Trojan	AceDeceiver manages to install itself without any enterprise certificate at all. It does so by exploiting design flaws in Apple's DRM mechanism
2016	iOS	Fusob	Ransomware	Demands ransom for illegal actions
2017	Android	ToastAmigo	Backdoor	Installs additional malware on an affected device using the toast overlay attack
2017	Android	ViperRAT	Trojan	Allows the attacker to access device data, SMS, WhatsApp database and encryption keys, browsing and search histories, documents and archives found in storage, and photos taken.
2017	iOS	Safari JavaScript pop-up scareware	Ransomware	Blocks the Safari browser on iOS until the victim pays the attacker money in the form of an iTunes Gift Card
2018	Android	BianLian	Trojan	Requests permissions that allows it to read, send, and receive text messages; monitor and make calls; insert overlays on banking applications; lock the device screen
2018	Android	Emotet	Trojan	Persuades users to click the malicious files by using tempting language about "Your Invoice", "Payment Details", or possibly an upcoming shipment from well-known parcel companies.
2018	iOS	Pegasus	Spyware	Pegasus has multiple spying modules such as taking screenshots, recording calls, accessing messenger applications, keylogging and exfiltrating browser history
2018	iOS	Roaming Mantis	Trojan	Targets iOS devices with phishing attacks, and desktops and laptops with the Coinhive cryptomining script
2019	Android	Cerberus	Trojan	Intercepts the calls, forwards the SMSs, locks the device and key logger
2019	Android	XHelper	Trojan	Displays popup ads, which redirects users to the Google play store and prompts users to install other apps, indicating that the actors behind the Trojan make profits off pay-per-install commissions

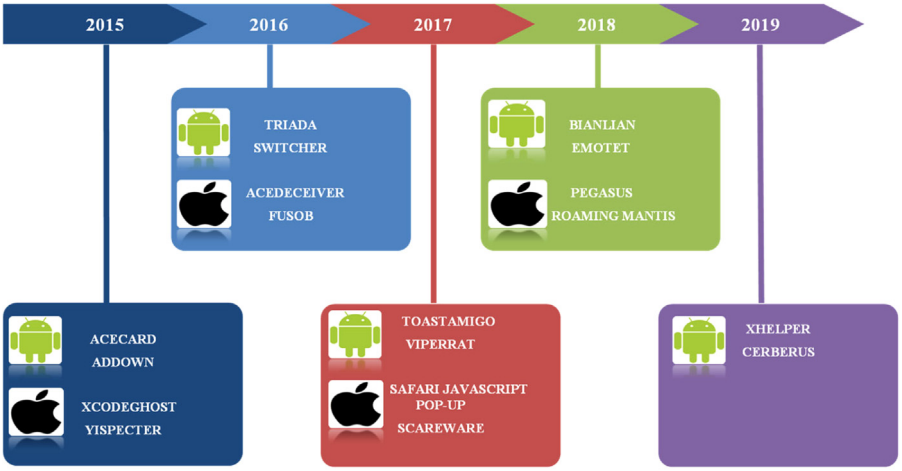

Fig. 16. Timeline of Android and iOS malware from 2015–2019.

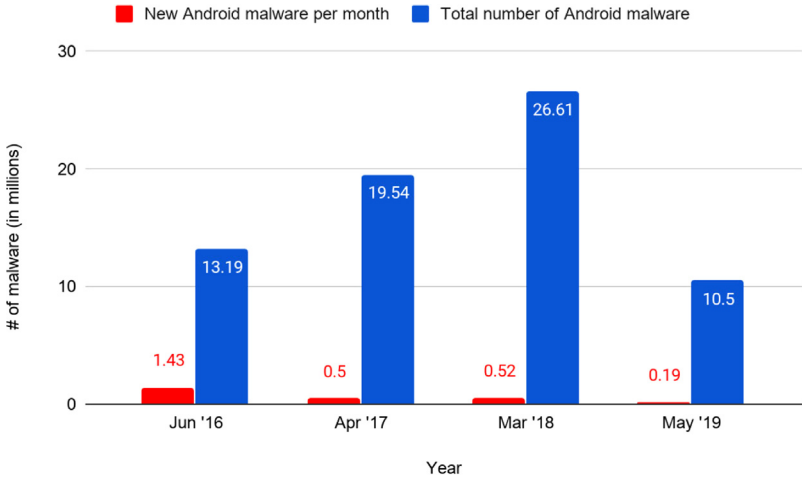

Fig. 17. Growth of Android malware from 2016 to 2019..

the mapping of the malware with the known vulnerabilities automatically. The aim is that in near future if a new Android malware variant appears, developers can know at prior which vulnerability it will exploit and appropriate measures can be taken timely and in a cost effective manner.

Major challenges in this domain and forthcoming research directions are divided into main buckets — device integrity, authentication mechanism, application privacy, and source code and hardware security along with the severity level of each problem, as discussed in Table 11.

Apart from the aforementioned challenges, other problems which prevail in Android security domain is the unavailability of comprehensive malware dataset. There is a need of standard, structured and updated datasets since the Android malware are increasing at a striking rate. Existing malware datasets are not comprehensive, thereby lacking up to date information. Such datasets can be helpful to perform malware detection analysis in an efficient manner. Also, large data volume poses difficulty to develop robust and efficient detection techniques. ML approaches have proved to be fruitful as compare to conventional malware detection techniques, however, Machine learning (ML) algorithms fail while dealing with large data volume. A shift from ML to Deep Learning (DL) is required to handle issues like large data volume and high false positive rate.

Table 11
Future research directions.

Issues with Android (vs. iOS)	Category	Severity
Provides infrequent updates for OS	Device integrity	High
Triggers auto erase under illegal action	Device integrity	Medium
Provides low level face recognition; gets tricked by photo	Authentication	High
Allows third party apps	Application privacy	High
Takes permissions from user; iOS itself does thorough check	Application privacy	Medium
Allows app Interaction in sandbox environment	Application privacy	Medium
Exempts application provenance	Application privacy	Low
Allows distribution of source code	Source code	High
Lacks control over hardware security	Hardware security	High
Provides number of device variants and OS versions	Hardware security	Low
Leverages multiple device manufacturers; affects standardization and control	Hardware security	Medium
Prevailing code coverage problem	Source code	Medium
Evades zero-day attack detection	Application privacy	High
Tampers reverse engineering techniques	Application privacy	Medium

9. Conclusion

Android and iOS being the most popular among all the smartphone platforms, also attracts large number of attackers with

malicious intentions. The pragmatic approach used for comparing Android and iOS helps to understand that Android is more susceptible to security breaches and malware attacks. Therefore, in the research domain Android becomes the first choice. The paper also presents forthcoming challenges and research directions, which are bucketed in the different categories along with the severity of each problem. These problems do not seem to be solvable in next coming years and continue to stay for long. The major buckets are device integrity, application privacy, hardware security, source code and authentication mechanisms, which should be focused by the researchers and academicians. These research directions can prove to be fruitful in smoothening the Android security domain, where application developers will be able to develop a more accurate, efficient, robust and scalable mechanisms.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. O'Dea, Number of mobile phone users worldwide from 2016 to 2021 (in billions), 2020, Available from: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[2] Statcounter GlobalStats, Mobile operating system market share worldwide, 2020, Available from: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[3] Liam Tung, Bigger than Windows, bigger than iOS: Google now has 2.5 billion active Android devices, 2020, Available from: https://www.zdnet.com/article/bigger-than-windows-bigger-than-ios-google-now-has-2-5-billion-active-android-devices-after-10-years/.

[4] J. Clement, Mobile app downloads worldwide from 2018 to 2024, by store, 2020, Available from: https://www.statista.com/statistics/1010716/apple-app-store-google-play-app-downloads-forecast/.

[5] S.F. Hidhaya, A. Geetha, Detection of vulnerabilities caused by webview exploitation in smartphone, in: Ninth International Conference on Advanced Computing (ICoAC), 2017, pp. 357–364.

[6] L. Zhang, Smartphone App Security: Vulnerabilities and Implementations Doctoral dissertation, 2018.

[7] M.T. Ahvanooey, Q. Li, M. Rabbani, A.R. Rajput, A survey on smartphones security: Software vulnerabilities, malware, and attacks, 2020, arXiv preprint arXiv:2001.09406.

[8] M. Talal, A.A. Zaidan, B.B. Zaidan, O.S. Albahri, M.A. Alsalem, A.S. Albahri, M. Alaa, Comprehensive review and analysis of anti-malware apps for smartphones, Telecommun. Syst. 72 (2) (2019) 285–337.

[9] B. Shrestha, D. Ma, Y. Zhu, H. Li, N. Saxena, Tap-wave-rub: Lightweight human interaction approach to curb emerging smartphone malware, IEEE Trans. Inf. Forensics Secur. 10 (2015) 2270–2283.

[10] S. Lee, D.Y. Ju, Assessment of malicious applications using permissions and enhanced user interfaces on android, in: IEEE International Conference on Intelligence and Security Informatics (ISI), 2013, p. 270.

[11] B. Rashidi, C.J. Fung, A survey of Android security threats and defenses, JoWUA 6 (3) (2015) 3–35.

[12] T. Cho, S.H. Seo, A strengthened Android signature management method, KSII Trans. Internet Inform. Syst. 9 (3) (2015).

[13] C. Szongott, B. Henne, M. Smith, Evaluating the threat of epidemic mobile malware, in: IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2012, pp. 443–450.

[14] P. Teufl, T. Zefferer, C. Stromberger, Mobile device encryption systems, in: IFIP International Information Security Conference, 2013, pp. 203–216.

[15] S. Salerno, A. Sanzgiri, S. Upadhyaya, Exploration of attacks on current generation smartphones, Procedia Comput. Sci. 5 (2011) 546–553.

[16] J. Zdziarski, Identifying back doors attack points and surveillance mechanisms in iOS devices, Digit. Invest. 11 (2014) 3–19.

[17] J. Wang, Introduction to Computing Applications in Forestry and Natural Resource Management, CRC Press, 2017.

[18] T.M. Grønli, J. Hansen, G. Ghinea, M. Younas, Mobile application platform heterogeneity: Android vs Windows phone vs iOS vs Firefox OS, in: IEEE 28th International Conference on Advanced Information Networking and Applications, 2014, pp. 635–641.

[19] R. Loftus, M. Baumann, R. van Galen, R. de Vries, Android 7 file based encryption and the attacks against it, 2017.

[20] S. Diaz, IOS security - iOS 11, 2020, Available from: https://www.readkong.com/page/ios-security-ios-11-january-2018-8964944.

[21] L. Davi, A. Dmitrienko, A.R. Sadeghi, M. Winandy, Privilege escalation attacks on android, in: Information Security, Springer, 2011, pp. 346–360.

[22] A.J. Bhatt, C. Gupta, Comparison of static and dynamic analyzer tools for iOS applications, Wirel. Pers. Commun. 96 (3) (2017) 4013–4046.

[23] T. Werthmann, R. Hund, L. Davi, A.R. Sadeghi, T. Holz, PSiOS: bring your own privacy & security to iOS devices, in: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, ACM, 2013, pp. 13–24.

[24] T. Groß, M. Ahmadova, T. Müller, Analyzing Android's file-based encryption: Information leakage through unencrypted metadata, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019, pp. 1–7.

[25] Y. Shen, H. Wang, Enhancing data security of iOS client by encryption algorithm, in: IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017, pp. 366–370.

[26] J.K. MacDuffie, P.A. Morreale, Comparing android app permissions, in: International Conference of Design, User Experience, and Usability, 2016, pp. 57–64.

[27] M. Lutaaya, Rethinking app permissions on iOS, in: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, 2018, pp. 1–6.

[28] H. Pieterse, M. Olivier, R. van Heerden, Detecting manipulated smartphone data on Android and iOS devices, in: International Information Security Conference, 2018, pp. 89–103.

[29] M. Opoku, J.G. Davis, P. Nimbe, Security evaluation of the smartphone platforms: A case study with Android, iOS and Windows phones, Asian J. Math. Comput. Res. (2016) 234–259.

[30] R. Kumar, R.R. Goyal, On cloud security requirements, threats, vulnerabilities and countermeasures: A survey, Comp. Sci. Rev. 33 (2019) 1–48.

[31] S. Garg, R.K. Singh, A.K. Mohapatra, Analysis of software vulnerability classification based on different technical parameters, Inform. Secur. J.: Glob. Perspect. 28 (2019) 1–19.

[32] CVE details, 2020, Available from: https://www.cvedetails.com/.

[33] I. Popescu, The influence of vulnerabilities on the information systems and methods of prevention, Int. J. Inform. Secur. Cybercrime (IJISC) 7 (2) (2018) 25–32.

[34] S.S. Chakkaravarthy, D. Sangeetha, V. Vaidehi, A survey on malware analysis and mitigation techniques, Comp. Sci. Rev. 32 (2019) 1–23.

[35] P. Yan, Z. Yan, A survey on dynamic mobile malware detection, Softw. Qual. J. 26 (3) (2018) 891–919.

[36] A. Cimitile, F. Mercaldo, V. Nardone, A. Santone, C.A. Visaggio, Talos: no more ransomware victims with formal methods, Int. J. Inform. Secur., 17(6) pp. 719–738.

[37] A. Qamar, A. Karim, V. Chang, Mobile malware attacks: Review, taxonomy & future directions, Future Gener. Comput. Syst. 97 (2019) 887–909.

[38] F. Pierazzi, G. Mezzour, Q. Han, M. Colajanni, V.S. Subrahmanian, A data-driven characterization of modern Android Spyware, ACM Trans. Manage. Inform. Syst. (TMIS) 11 (1) (2020) 1–38.

[39] J. Lim, J.H. Yi, Structural analysis of packing schemes for extracting hidden codes in mobile malware, EURASIP J. Wireless Commun. Networking 1 (2016) 221.

[40] J. Clement, Development of new Android malware worldwide from 2016 to 2019 (in millions), 2020, Available from: https://www.statista.com/statistics/680705/global-android-malware-volume/.

[41] S. Garg, N. Baliyan, A novel parallel classifier scheme for vulnerability detection in android, Comput. Electr. Eng. 77 (2019) 12–26.

[42] S. Garg, N. Baliyan, Data on vulnerability detection in android, Data Brief 22 (2019) 1081–1087.

[43] S. Garg, N. Baliyan, Android malware classification using ensemble classifiers, in: Cloud Security: Concepts, Applications and Perspectives, Chapter 13, CRC Press, 2021, In Press.