A Formal Overview of Application Sandbox in Android and iOS With the Need to Secure Sandbox Against Increasing Number of Malware Attacks

** Jithu Philip*
*** Merin Raju*

Abstract

Android and iOS are the most popular and commonly used mobile operating systems in the world. The increase in the usage share of these along with the extensive use of internet in smartphones and tablets lead to security threats being spread more frequently than expected for both. This document describes one of the security mechanisms used from earlier days of operating systems called Sandbox or Application Sandbox. The document also specifies details regarding how Application Sandbox is implemented in Android and iOS along with the common causes for malware attacks on both and an analysis about the common vulnerabilities and exposures reported on both the platforms. The major issue that the document focuses on is to make the users aware of the need to make use of protection mechanisms like Sandbox effectively to reduce the number of malware attacks.

Keywords: **Application Sandbox, Android, iOS, Malwares, Operating Systems, Sandbox, System Security.**

I. INTRODUCTION

A. What is Sandbox?

Sandbox (commonly called Application Sandbox) is a computer security mechanism which is used specifically for securing running programs from creating system failures and spreading software vulnerabilities, thereby reducing the harm caused by attackers.

B. Uses and needs of a Sandbox

The major use of a sandbox is for security researchers and normal users for executing untested or untrusted programs or codes from third parties that are unverified. A program which is running in sandboxed mode is only allowed a controlled set of resources, thereby making sure that it does not affect the host machine or the core parts of the operating system.

C. Existing implementations of Sandbox in Operating Systems

1) Jail : It is a sandbox implementation applied with prohibiting access to network resources and filesystem namespace (a kind of approach used in iOS).

2) Secure Computing Mode (seccomp): It is a sandbox used in the Linux kernel. If seccomp is applied properly in strict mode, it only allows the running process to execute basic system calls like read(), write(), exit(), and sigreturn().

3) User Account Control (UAC): Microsoft's Windows Operating Systems makes use of a kind of sandbox by restricting user's access controls with a low process mode. This was implemented from the start of Microsoft's Windows Vista operating systems. The low process mode called the User Account Control (UAC), works in a non-privileged mode and only allows the user to write to a specific directory and with specific registry keys.

4) Rule-based execution: It controls the system by assigning access levels for users or programs by a set of specified rules. Access level permissions of users for performing process level activity (what processes are started and spawned etc.), and also controlling the file/registry security (which programs can read/write to the file system/registry) can be achieved by rule based

Manuscript received March 26, 2019; revised April 28, 2019; accepted May 5, 2019. Date of publication June 6, 2019.
* J. Philip is Multimedia Specialist with Philco Media, Kerala.
**M. Raju is Lecturer with Department of Commerce, Bishop Kurialacherry College for Women, Kottayam, Kerala, India-686 561.
(email: merin.raju12s@gmail.com)
DOI: 10.17010/ijcs/2019/v4/i3/146164

execution. Linux Operating System makes use of SElinux and Apparmor security frameworks for this to achieve.

5) Virtual machines: It works like an actual operating system. It is a guest operating system which boots and works on top of the actual host machine. The guest operating system is sandboxed and is permitted to access the host machines resources only through an emulator.

II. HOW APPLICATION SANDBOX WORKS

Application sandbox is an access control technology which is mostly enforced at kernel level. Sandboxing of applications is designed in such a way that users have the provision to choose what they share with an application. This allows users with the option that their critical data and access to the system itself is protected in its major share, even if an application that is running in the system is compromised and is vulnerable to attack. It needs to be accepted in some way or the other as no technology is perfect and so, the main aim here would be to minimize the chances of potential risks, so that even if a successful attack happens, the damage is minimized. So in this sense, application sandbox does not prevent attacks against the app, but it is useful in minimizing the harm a successful attack can cause.

III. BENEFITS AND DESIGN OF APPLICATION SANDBOX

The main benefit of a sandboxed app than a non-sandboxed app is that a non-sandboxed app has the full rights and permissions of a user who is running the app. So, it has access to all the resources that the user can access and is more vulnerable to attacks. A sandboxed application environment restricts this by its design, thereby securing the system. Both of these design strategies are discussed next.

A. Case Without Application Sandbox

The representation of an application working without any sandbox is shown in Fig.1. [2]. In this case, the application, app1 has unrestricted access to all the user data and to all the system resources. So here, an attacker who gains access to the application can get track of the entire files in the system and can also tamper with the system resources. This type of design strategy always places security holes, that an attacker can potentially make use of, and take control of the entire system through the application itself.

B. Case With Application Sandbox

The representation of an application working within

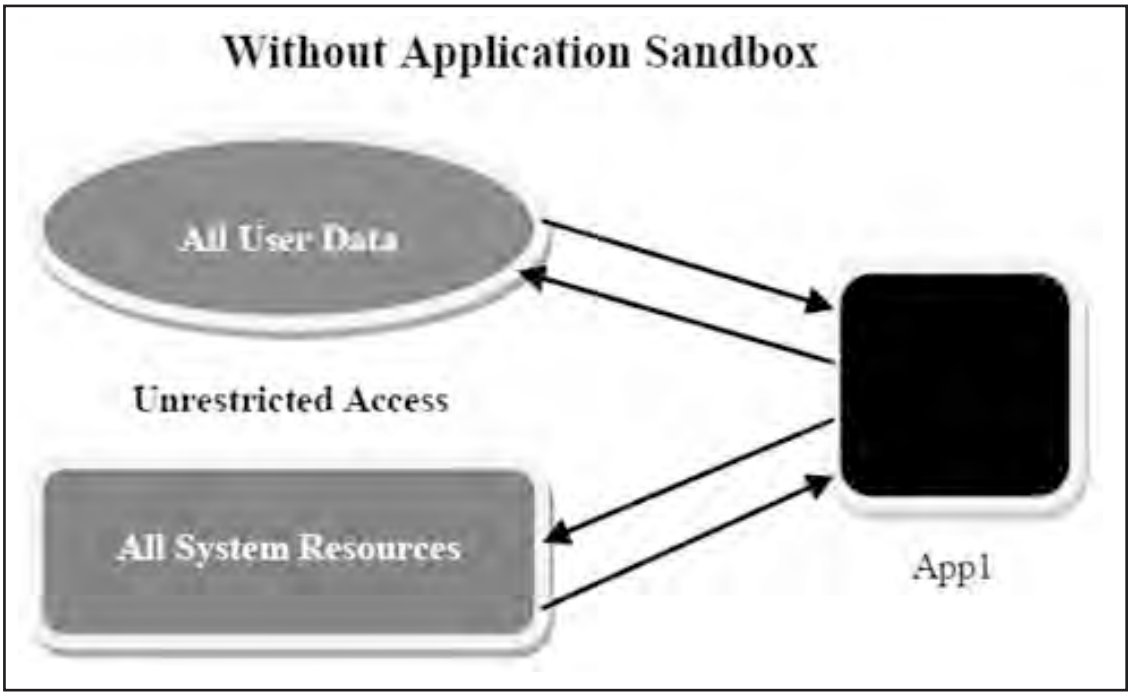

Fig. 1. The design of an application working without any sandbox applied

a sandboxed environment is shown in Fig. 2. [2]. In this case, the application, app1, has only restricted access to its own data and resources. Here, users have the provision to choose what data and resources they share with the application. Since all the access permissions are explicitly given by the user, on the basis of his/her interactions with the application, app1, the user's data and resources other than what app1 is granted access is isolated from app1. This design strategy is somewhat more secure than what is done in case A. So the user is permitted to assign specific rights to an application.

C. Principles to Follow While Designing an Application in a Sandboxed Environment

As explained in the previous section, a sandboxed environment only allows the application to have a restricted set of data and resources based on what permission the user has given to it. By doing so, it protects the system from most of the attacks like corruption or deletion of user data, hijacking system hardware etc.

So an application working in sandbox must explicitly state its intent for acquiring higher level privileges for operations like:

(i) Accessing hardware like camera, microphone etc.
(ii) Access for network operations (both inbound and outbound operations).
(iii) Accessing user data like calendar, contacts etc.
(iv) Users storage space (read / write access to internal, external storage etc.).

Applications not following this rule (explicitly requesting for higher level privileges, if needed) needs to be rejected its access at runtime.

IV. EXISTING IMPLEMENTATIONS OF APPLICATION SANDBOX IN MOBILE OPERATING SYSTEMS

The most widely used mobile operating systems in the world, namely Android and iOS, both support and

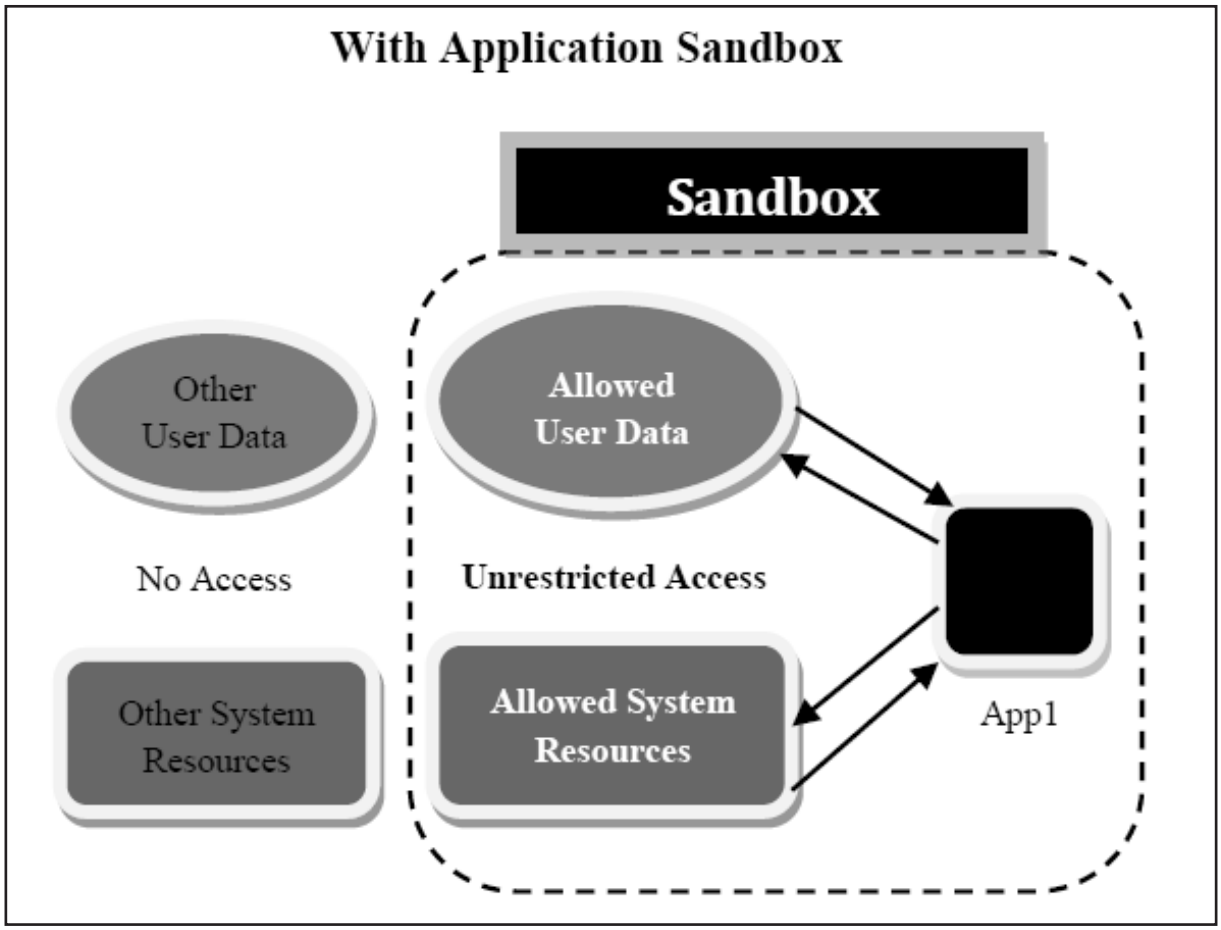

Fig. 2. The design of an application working with sandbox applied

make use of application sandbox in their own ways. The application sandbox implementation of both of these is discussed next.

A. Application Sandbox in iOS

For security purposes, the file system of iOS is treated in such a way that users do not have direct access to the file system. The implementation of iOS follows a sandbox (commonly referred to as *jail*), where access to the file system namespace and to unneeded network operations are prohibited. This design strategy gives an iOS application access only to the directories which reside inside its own sandbox directory. For this to be achieved, the installer creates different container directories inside an application's sandbox directory during the installation of an app. The container directories that were created are meant for different purposes. For example, the bundle container directory holds the app's bundle, and the data container directory holds the data for both the app and the user. The data container directory can create further sub directories within itself so that data can be organized. A representation of an iOS application operating within its own sandbox directory [1], [3] is shown in Fig. 3.

In general, an application installed in iOS is restricted from accessing or creating files outside its container directory. An exception to this kind of access restriction can be achieved by doing some specific tasks with the help of public system interfaces (for example, an application is allowed access to the users contacts) [1]. Even though the security architecture of iOS [1] is done in a better way, it became a practice that users jail breaking devices running iOS operating system became more to get rid of the access restrictions put by this sandbox strategy.

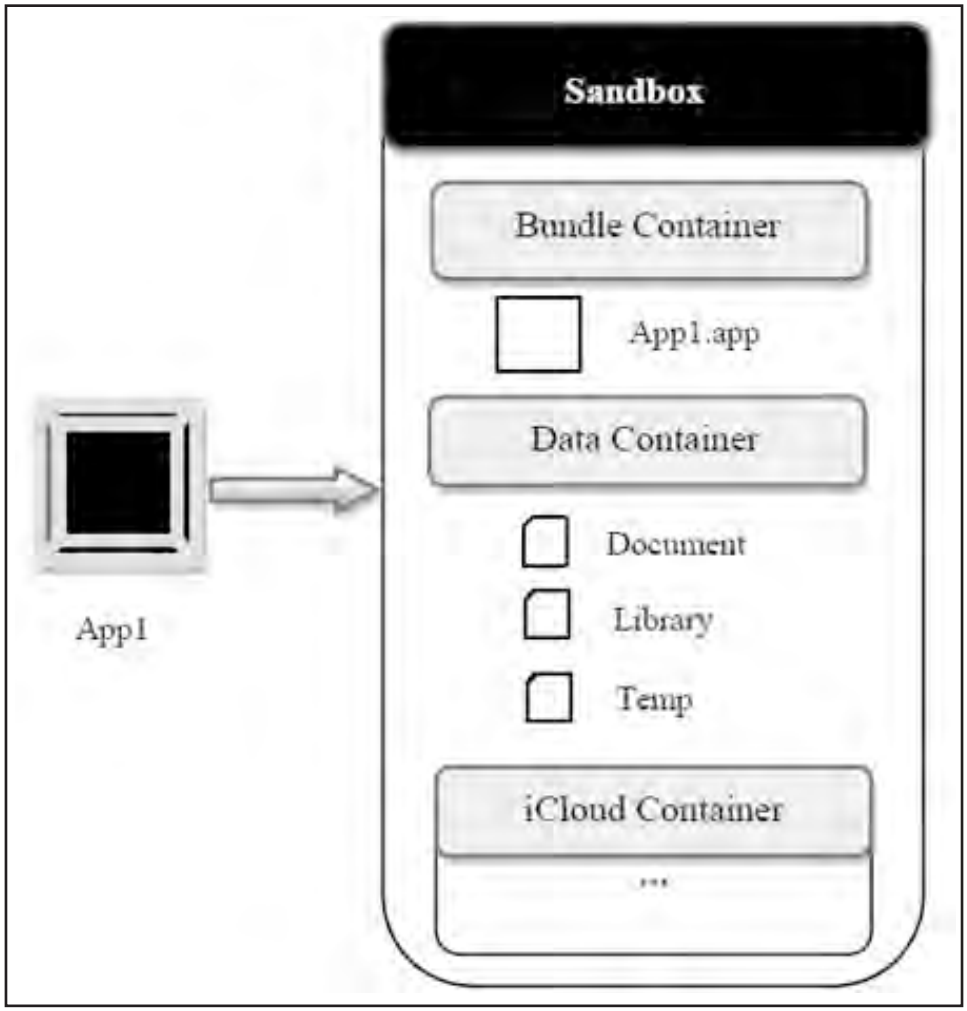

Fig. 3. An iOS application operating within its own sandbox directory

B. Application Sandbox in Android

As Android is an operating system which makes use of the Linux based kernel. It follows the Linux user-based protection for applications, and isolates each application. The sandbox is based on decades-old UNIX-style user separation of processes and file permissions.

For this to work, Android assigns a unique user ID (UID) to each application. This UID is used to set up a kernel-level Application Sandbox. The kernel does the security between apps and secures the system by assigning user ID (UID) and group ID. By default, apps can't interact with each other and have only restricted access to the operating system.

For example, if application A tries to access application B's data without necessary permission, the operating system protects against this behavior, because application A does not have appropriate user privileges to do this. This helps the system in securing against the malicious behavior of applications.

As Application Sandbox is in the kernel, all of the software above the kernel, such as operating system libraries, application framework, application runtime, and all application run within the Sandbox. There are no restrictions for application developers on Android, regarding how an application can be written to enforce security; in this respect, native code is as sandboxed as interpreted code.

1) Protections in Android

Android uses a number of protection mechanisms to enforce application sandbox. The enforcements applied are implemented and introduced over time in various Android version upgrades. This implementation has strengthened the original UID based discretionary access control (DAC) sandbox. Various protection mechanisms [4] introduced on different Android versions are:

a) From Android 5.0, SELinux provided mandatory access control (MAC) separation between the system and apps. However, third-party apps ran within the same SELinux context, so inter-app isolation was primarily enforced by UID DAC (Discretionary Access Control).

b) From Android 6.0, the SELinux sandbox was extended to isolate apps across the per-physical-user boundary. This provided safer default for private app data (however, apps can override these defaults).

c) From Android 8.0, all apps were set to run with a seccomp-bpf filter (an extension of the seccomp security facility used in the Linux Kernel) that limited the system calls that apps were allowed to use, thus strengthening the app/kernel boundary.

d) In Android 9 all non-privileged apps (with targetSdk-Version >= 28 (which specifies the API Level that the application is designed to run)) must run in individual SELinux sandboxes, providing MAC on a per-app basis. This protection improves app separation, prevents overriding safe defaults, and (most significantly) prevents apps from making their data world accessible.

V. MOBILE OPERATING SYSTEMS AND MALWARES

A common question that normally arises in the mind of every smartphone or tablet (either running Android or iOS) user is that "can their devices be affected by viruses?" The answer to this question is that it won't happen in the traditional sense like a computer virus. The most common threat to mobile operating systems is in the form of malwares.

Malwares are not self replicating threats like a virus. But if proper security measures are not taken, these can cause more damages than a virus. Malwares come in many forms, like, a spyware which gathers and steals personal data for third parties, a Ransomware which holds personal files as hostage and demands a ransom in order to release them.

A. How users can trace whether or not their device is affected by malwares?

Malware, like viruses, is all about the stealth attack. It steals users' documents and sensitive information without their prior knowledge and utilizes them. So users always need to be aware about this fact, and check for vulnerabilities frequently to stay secure. Some of the facts that the user should look forward to know whether the device is secured or not are stated below:

- Data usage keeps increasing without user's knowledge
- Excessive crashing of applications
- Adware pop-ups more than normal
- Unfamiliar apps installed without users knowledge
- Faster battery drains than normal
- Device keeps overheating in idle conditions

B. The major cause of malwares on both platforms with the differences in Application Signing process, File System Namespace, and Software Updates

Both of the operating systems, Android and iOS are designed in a better way to make the system secure.

However, attackers make use of minor flaws in the design to check for vulnerabilities and take control of the system.

Android makes use of its Play Store for users to download and install applications on their system. Applications can also be installed from outside. This is what hackers utilize to install malicious apps on devices. iOS on the other hand permits only signed applications from its App Store to be installed on its devices, thereby securing it against fake and malicious applications (an exception to this is where attacker jailbreaks an iOS devices to get rid of these limitations).

Another area where both these operating systems differ is in the frequency of software updates being released and the number of devices being supported overtime. It can be surely said that iOS devices gain advantage in this factor as for iOS devices, both the hardware and software are designed by the same company, Apple Inc. So, security patches and newer versions of the operating systems are released more frequently, and even supported for outdated devices. For Android, it too gets security patches and updates for the operating system, but the frequency depends on how the manufacturers of different hardwares using Android are providing it. Also, for outdated devices having old hardware, latest releases of software or security patches were not confirmed in most cases. As Android is an operating system developed by Google, latest software updates and security patches are promised for Google's own devices (like Google Pixel) and devices running under Android One programme (for atleast two years from the release of the device as promised by Google).

As described in the Section IV of this document, both these platforms, Android and iOS, employ Application Sandbox in their own ways. Android is a Linux based operating system, it does follows the security measures derived from it. Android version 8.0 makes use of an extension (seccomp-bpf) of the Linux Kernel based sandboxing technique, seccomp (Secure Computing). It also employs SELinux to enforce mandatory access control (MAC) over all processes [5], which use a Linux Kernel feature called LSM (Linux Security Modules, which works as a security framework to provide services to security modules). iOS on the other hand is somewhat more restrictive and provides a more secure approach in the case of Application Sandbox. Here, an application which is installed has only restricted access to data which resides inside its own sandbox directory. It makes uses of a kind of restricted Sandbox (jail), so that the file system namespace is restricted to the user. Also network services that are unnecessary are prohibited.

The lack of security patches rolled out on most of the outdated devices running Android and also its open source nature makes it vulnerable to attacks. On the other hand, jail breaking an iOS device leads to the existence of more malwares on its platform.

C. Analysis of Reported Vulnerabilities on Both Platforms

As part of the analysis process various information were collected from different sources [6], [7] to know how fast the reported number of vulnerabilities are growing year by year on both platforms, and also by the type of malicious behavior. Based on that the vulnerabilities reported on Android platforms year by year, for the years from 2009 to 2018, is shown in Fig. 4. Also, the reported vulnerabilities on these years based on the type of malicious behavior are shown in Fig. 5. The vulnerabilities reported on iOS platform year by year, for the years from 2009 to 2018, is shown in Fig. 6. The reported vulnerabilities on these years based on the type of malicious behavior are shown in Fig. 7.

VI. CONCLUSION

As the studies specified in this document, Application Sandbox is one security mechanism which is used widely in most of the operating systems in their own ways of implementations. So what a user needs to do keep in mind is that security threats are increasing day by day, and every user who is using devices having mobile operating systems, needs to think before they share their sensitive documents with an application. Users also needs to keep in mind the fact that giving excessive permissions to an application can also lead to vulnerabilities, as the use of fake applications, and hackers taking extensive control of it to mislead the control of the device itself is increasing. So the intention of this document is to give users the awareness so that they need to use the inbuilt Sandbox technology in a proper way to achieve better levels of security in their systems.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable and insightful comments on the details of this document.

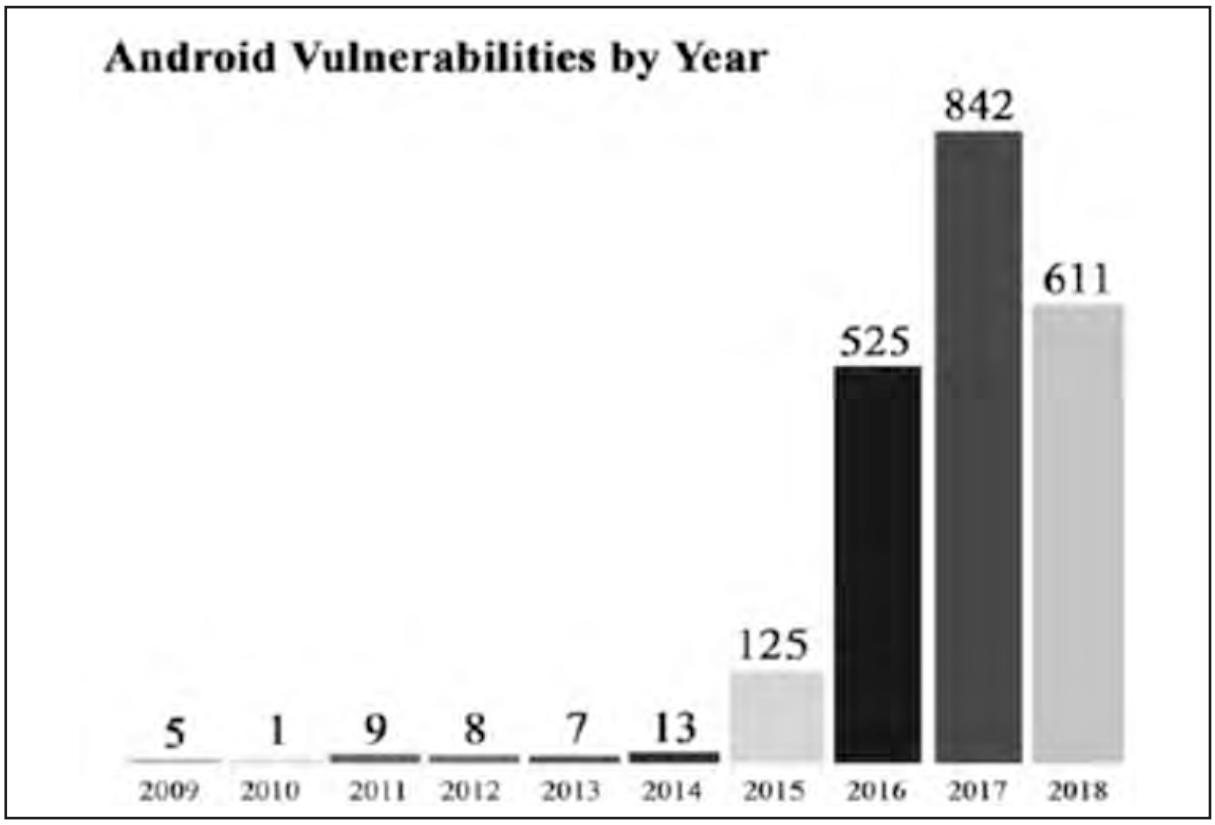

Fig. 4. The number of reported vulnerabilities on Android year by year, for the years from 2009 to 2018 [6]

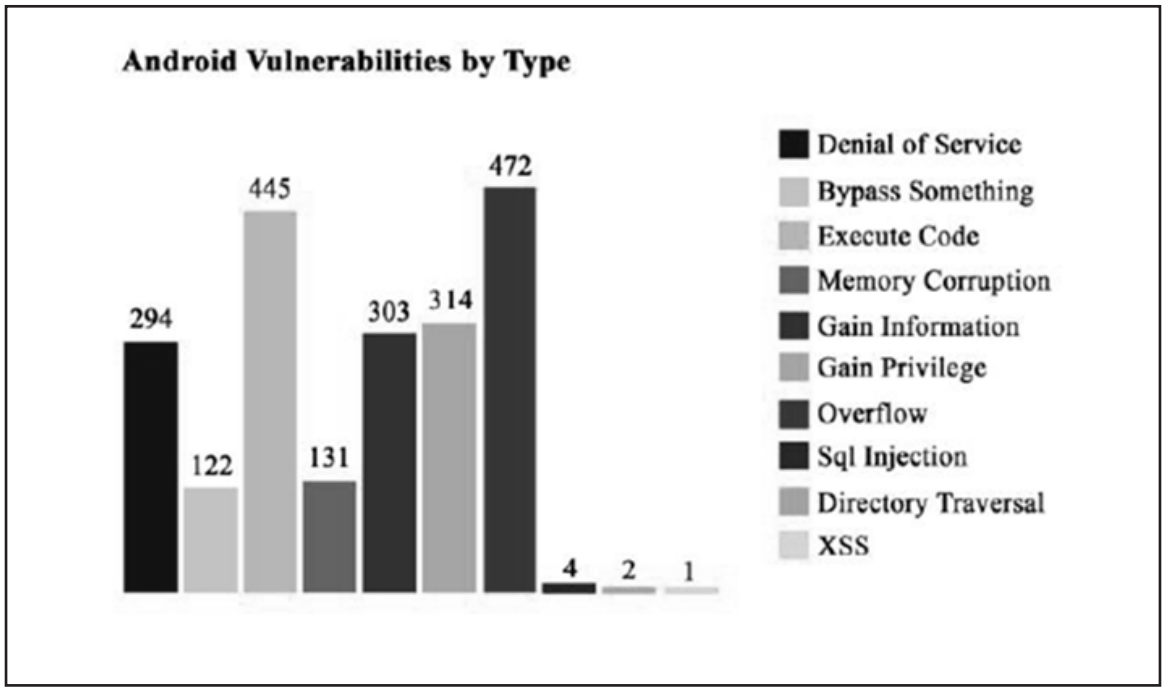

Fig.5. The number of reported vulnerabilities on Android based on the type of malicious behavior for the years from 2009 to 2018 [6]

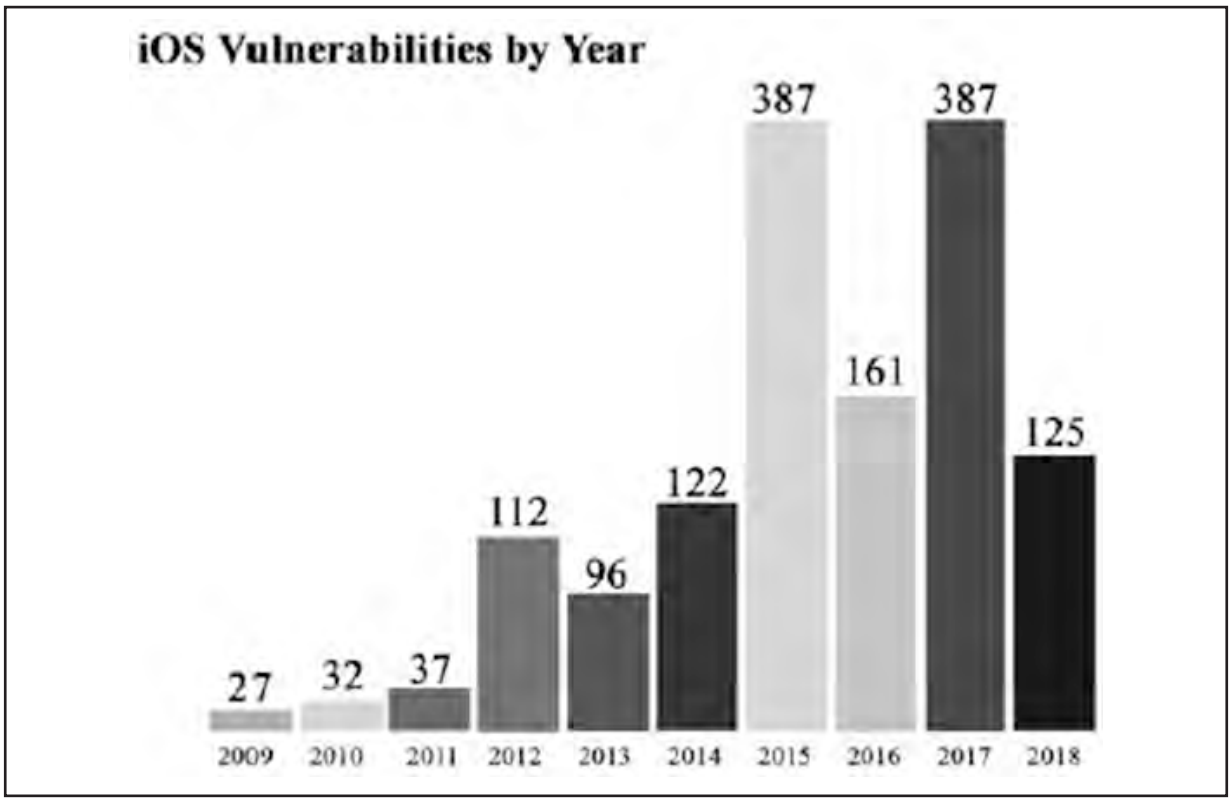

Fig. 6.The number of reported vulnerabilities on iOS year by year, for the years from 2009 to 2018. [6]

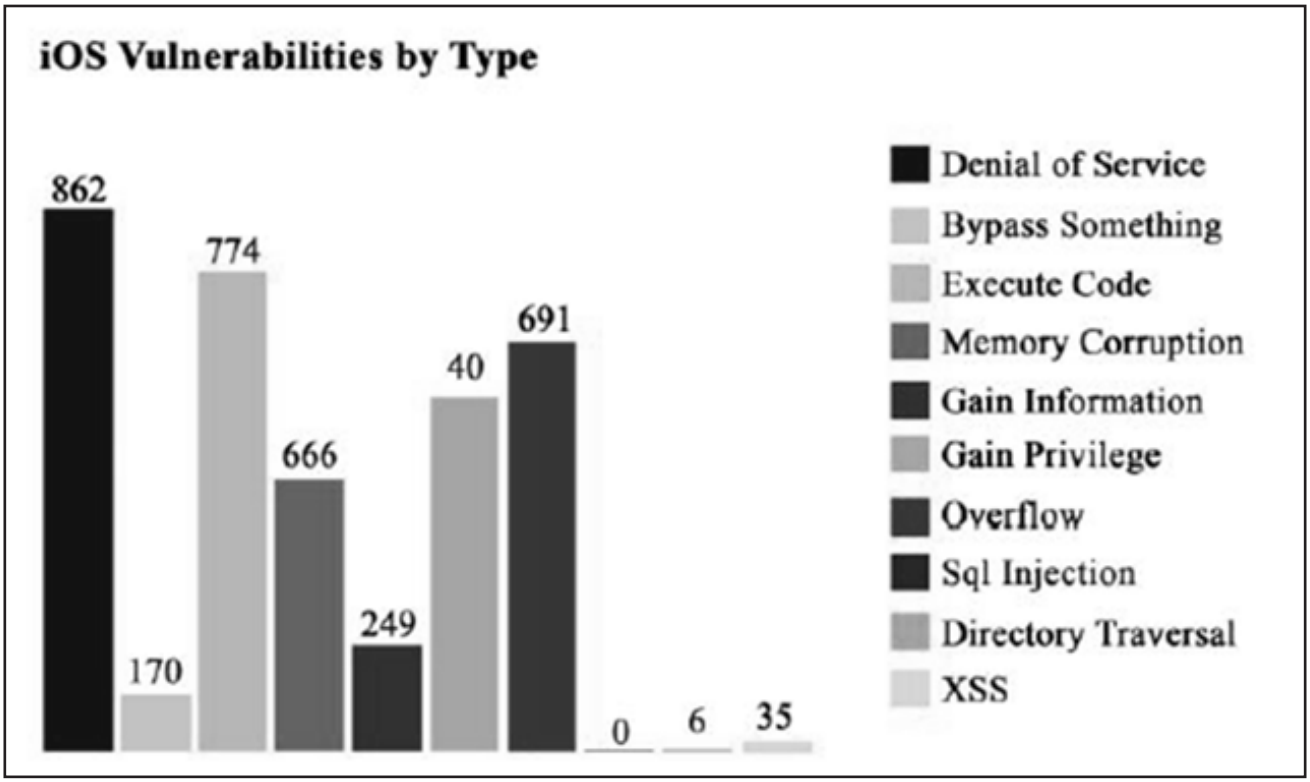

Fig. 7. The number of reported vulnerabilities on iOS based on the type of malicious behavior for the years from 2009 to 2018 [6]

REFERENCES

[1] J. Philip and M.Raju "An overview about the security architecture of the mobile Operating System iOS," *Indian Journal of Computer Science*, vol. *4*, no. *1*, pp. *13-18*, January-February 2019. Doi: 10.17010/ijcs/2019/v4/i1/142412

[2] "App Sandbox Design Guide." [Online]. Available: https://developer.apple.com/library/archive/documentation/Security/Conceptual/AppSandboxDesignGuide/AboutAppSandbox/AboutAppSandbox.html

[3] "File System Programming Guide."[Online]. Available: https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html

[4] "Application sandbox." [Online]. Available: https://source.android.com/security/app-sandbox

[5] "Security-enhanced Linux in Android." [Online]. Available: https://source.android.com/security/selinux

[6] "CVE details Android: Vulnerability statistics," [Online]. Available: https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224

[7] "CVE details iPhone OS: Vulnerability Statistics," [Online]. Available: https://www.cvedetails.com/product/15556/Apple-Iphone-Os.html?vendor_id=49

About the Authors

Jithu Philip received M. Sc. degree in Computer Science from School of Computer Sciences, Mahatma Gandhi Univ.ersity, Kottayam, Kerala, India in 2014. He is currently working as Multimedia Specialist with Philco Media, Kottayam, Kerala, India. He is also focused on conducting simulations as an independent security researcher and has written research papers related to Operating Systems and Computer Security.

Merin Raju received M. Sc. degree in Computer Science from School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala, India in 2014. She is currently working as Lecturer in Computer Science, Department of Commerce, Bishop Kurialacherry College for Women, Amalagiri, Kottayam, Kerala, India. Her research interest is computer security.