De La Salle University
Taft Ave., Manila

Journal Review on

"Android vs iOS: A Comprehensive Evaluation of Security Vulnerabilities"

Operating Systems Lecture

(OPESSYS)

Submitted by:

ARINGO, MICHAEL JUDRICK M.  /MJMA

BANAL, KENAN A.  /KAB

CONCORDIA, JHYREL JULIENE R.  /JJRC

TEVES, FAITH SHAREINE V.  / FSVT

Submitted to:

MRS. ARMIE PAKZAD

August 01, 2024

Title: Android vs iOS: A Comprehensive Evaluation of Security Vulnerabilities

Background

From the time it was invented to today's age, mobile phones have transformed from being a luxury item few can possess to being a necessity for many people. With its numerous capabilities and functions similar to that of a computer, its biggest advantages being portability and cellular communication, it remains to be relevant and convenient to modern society. Throughout the development of mobile phones over time, many improvements have been made to each component of mobile phones, including its operating system (OS). Mobile OS is a software that enables mobile phones, cellular phones, tablets, and devices of similar build to run applications and programs in a convenient and efficient manner (Almisreb et al., 2019). Two of the most well-known and most used mobile OS are Android and iOS.

Android is a mobile OS developed by Google that is derived from the Linux kernel. It is designed for phones and tablets PCs with a touchscreen interface and other hardware such as light sensors, GPS receivers, Wi-Fi and LTE connectivity, cameras, and more (Goel & Singal, 2021). It allows Android applications to make use of said features through abstraction and develop an established and suitable system for the applications. The project development of Android began in 2003 by Android Inc., initially created as an OS for digital cameras, which was then altered to a mobile OS. It was later obtained by Google in 2005. Its existence was revealed in 2007 as the Android Open Source Project (AOSP) and was later commercialized in 2008. Due to its nature as an open-source code, customizable, and low-cost OS with its Google-bound proprietary software, its popularity among technological companies and developer communities continues to grow despite the competition between other rising mobile operating systems (Huang & Wu, 2018).

iPhone Operating System (iOS) is a mobile OS created by Apple Inc., an OS tailored for Apple mobile devices (iPhones and iPads) that continues to revolutionize technology and software development (Rafalski, 2024). Its OS supports multiple hardware and software features like direct manipulation, user gestures, WiFi, VPN, a personal assistant with a voice feature named Siri, and more. It was first unveiled in January 2007 by Steve Jobs, a device that gives

way for opportunities in OS and application development as a new perspective on mobile computing has been introduced to the community. From years 2018-2022, there has been a significant increase in the percentage of iPhones sold around the globe, with the United States as its largest market. As of today, the OS remains to be a popular pick in the market despite its high price due to its quality build and hardware, and annual updates on its software, making it worthwhile to purchase for its near-flawlessness (Mickle, 2023).

Significance

The development of secure software for smartphones has become more vital since people are maximizing the capabilities of those handheld devices that are catching up to what desktops or laptops can provide the people. Hence, people invest their money in mobile devices that they deem to be most suitable for their work and personal lives. Thus, mobile operating systems must remain secure and prevent hijackers from taking users' sensitive and private information. Additionally, strong security measures must be implemented since mobile devices are more commonly used today. The two popular mobile operating systems are iOS and Android, which are both compared with one another every time a new phone is released. Furthermore, the significance of the distinct to various parts of the society are as follows:

To the Community:

1. Provide awareness regarding the possible vulnerabilities and security risks associated with iOS and Android operating systems to enable the community to make informed decisions when choosing a mobile device.
2. Elaborate on the strengths and weaknesses of each operating system in terms of their security measures.

To the Academe:

1. Expand studies regarding the difference between the two mobile operating systems.
2. Effectively compare and contrast iOS and Android security measures by using real-world data and applications.

To the Future Research:

1. Explain where a more comprehensive study should be implemented by highlighting the existing research gaps in security measures of the two mobile operating systems.
2. Serve as a foundational study for future researchers to explore a wider area of vulnerabilities in the updated operating systems of iOS and Android.

Related Literature

The research conducted by Umar and Wakili (2023) performed a comparative case study on the current OS, also called the operating system, that emphasizes memory management and security features, examining Windows, iOS, and Android. The primary objectives are to prepare a comparative case study with an analysis of modern OS (Windows, iOS, and Android) in terms of memory management and security features, and to discuss the relative strengths and weaknesses of each OS (Umar & Wakili, 2023). Furthermore, a case study approach was used to examine the security features of Windows, iOS, and Android, together with the process of managing memory. The researchers found out that iOS, known for its ease of use and flexibility, offers advanced security features like data encryption and Touch ID, which are automatically enabled (Umar & Wakili, 2023). Android, while also user-friendly and flexible, is more susceptible to attacks compared to iOS. Furthermore, the study examined the architecture of each OS, detailing their layered designs and how iOS manages memory and security. This is done by employing automatic reference counting for memory management. Moreover, Android's memory management involves a "mark-and-sweep" garbage collector and a virtual machine for app sandboxing. In terms of security, iOS offers hardware encryption and secure boot features, and Android has app permission controls and encryption. Overall, the researchers concluded that iOS has the most comprehensive security features, and all three OS manage memory effectively according to their respective OS designs.

Another paper written by Nowfeek and Razeed (2022) did a review of security issues within the Android operating system. Moreover, the researchers primarily discussed the rise in security challenges with the advancement of mobile operating systems and hardware. Likewise, the researchers highlight the vulnerabilities that arise from the permission model, where users must approve permissions during app installation, potentially allowing apps to access resources without further consent. This suggests that Android's design may result in security threats and

weaknesses, such as spyware, denial of service attacks, information leaks, permission escalation attacks, app collusion, and the repackaging of apps with malicious code. Additionally, the researchers note that while Android is considered more secure than iOS in some aspects, it faces unique problems because of its open ecosystem. In addition, security features such as sandboxing, file system encryption, and multi-sensor authentication systems have been implemented to mitigate risks. Likewise, the researchers also discuss proposed solutions to enhance security, including Kirin, RiskMon, Paranoid Android, Crowdroid, and DroidScope. Furthermore, the listed solutions range from risk assessment frameworks to behavior-based malware detection and dynamic analysis frameworks for detecting privilege escalation attacks. In conclusion, the researchers emphasized the importance of Android security in protecting user privacy and data. Some suggestions involve directions to develop more secure versions of the Android OS and call for continued efforts to address the evolving landscape of mobile security threats.

Kollnig et al. (2022) conducted a comparative study of iOS and Android apps to determine which platform offers better privacy protections. Their findings reveal that neither platform is definitively superior in protecting user privacy. Both iOS and Android engage in extensive third-party tracking and share unique user identifiers, often without adequate user consent. This study highlights widespread potential violations of privacy laws in the US, EU, and UK, suggesting a complex interplay between app ecosystems and privacy regulations. The researchers emphasize the need for enhanced transparency and user control over data practices on both platforms (Kollnig et al., 2022).

A comprehensive review by Ahmed and Sallow (2017) discusses the current security threats and solutions for the Android operating system. The authors highlight the challenges posed by the open-source nature of Android, which, while fostering innovation, also opens up numerous security vulnerabilities. These vulnerabilities arise from the diverse and often unregulated ecosystem of Android applications, leading to issues such as malware, data leakage, and unauthorized data access. The study classifies various security solutions into static, dynamic, and hybrid approaches, evaluating their effectiveness in mitigating Android-specific threats (Ahmed & Sallow, 2017).

In a study by Holmberg (2022), the author compared the security of Inter-App Communication (IAC) between iOS and Android. The research aims to bridge the knowledge gap on Apple's iOS security and IAC mechanisms, along with the security issues regarding Android Intents. Thus, he explored the iOS URL-scheme, which makes requests between applications and specifies what application will be launched. Additionally, the researcher also compared the differences between the Android Intent and iOS URL-scheme, and tried to identify techniques to help minimize "common exploits" or prevent hijacks with the two mechanisms. The identified possible exploits are the following: intent spoofing, denial of service (DOS), intent hijacking, privilege escalation, data leak, and scheme takeover. All of the aforementioned exploits have one goal, which is to trick or gain unauthorized access to the data and features in an application by utilizing how applications communicate with one another. The researcher discovered that the iOS system allows numerous applications to register the same URL-scheme, which may potentially allow "man-in -the-middle" attacks. Next, the author also identified that the iOS URL-scheme is much simpler than the Android Intent since it is only used to switch apps. In contrast, Android Intents is more versatile since it allows the sending of messages within the same or between different applications. Lastly, the mitigation strategies identified are encrypting sensitive information between applications, using universal links for iOS and app links for Android to remove the hijacking threat, and removing vulnerable APIs to prevent exploits.

Furthermore, a comparative study between Android and iOS conducted by Quissanga (2023) in terms of security concluded that iOS is more secure than Android. Both mobile operating systems, although safe from computer viruses, are vulnerable to other attacks and likely to be affected as many use either of them. Android, however, is reportedly more likely to be affected than iOS since it is an open-source platform, with programming languages involved in the OS that are commonly known by developers and programmers, making it susceptible to specific attacks. It is also possible for an external virus to take effect on Android because of installations of programs outside the official app store and by enabling users to take control of its systems through rooting. In iOS systems, these attacks may not affect it due to its nature of being a close-source platform, its restricting mechanisms, and having a different programming language that is not commonly used by developers and programmers. Quissanga recommended

in the end that information security policies like encryption, blocking, and firewalls should be implemented to improve the security of both operating systems.

Fu et al. (2024) developed an innovative technique to improve Android malware detection by merging multi-scale convolutional neural networks (MSCNN) with residual networks (ResNet). The study introduces a hybrid model designed to address Android's open-source vulnerabilities through enhanced feature extraction and detection capabilities. This improved MSCNN model is structured into three tiers, each responsible for extracting and integrating features across various dimensions. By maintaining a comprehensive range of semantic features, the model ensures effective malware detection in conjunction with ResNet. Experimental data show that this approach leads to a detection accuracy of 99.20% and precision of 99.49%, enhancing the F1-Score for MSCNN+ResNet18 by 4.8% over traditional methods. These findings highlight the potential of deep learning to significantly bolster mobile security.

Yang et al. (2022) proposed an effective Android malware detection method utilizing static features in their study published as "Android malware detection method based on highly distinguishable static features and DenseNet" in PLoS One. Their research introduces a machine learning framework using a fully densely connected convolutional network based on DenseNet, focusing on feature selection to efficiently handle large feature sets. They achieved a high accuracy of 99.72% in detecting malicious applications, highlighting the potential for robust security measures in the mobile ecosystem. Their findings are crucial for addressing security vulnerabilities in Android's widespread application environment, particularly considering the rise of mobile devices for sensitive operations like payments and data storage.

In a study by Janaka et al. (2023), a vulnerability detection model for Android was developed using LCVAndro. This comprehensive dataset is specifically designed to aid in the creation of AI models for detecting harmful vulnerabilities, specifically in the code of Android’s operating system. To evaluate the model's performance, the AutoML technique was applied to the dataset, showing its efficacy in identifying vulnerable code with a 94% accuracy, together with a 0.94 F1 score in terms of binary classification. Lastly, it has an accuracy of 94%, together with a 0.93 F1 score in multi-class classification, specifically on CWE. The analysis also compared models trained on different data subsets, demonstrating that a combined approach

produced superior results. Overall, the LCVAndro dataset makes a significant contribution to Android application security by providing a robust resource for training AI-based models to detect source code vulnerabilities. The high performance of the models trained on this dataset highlights its potential to significantly enhance the security of Android applications.

Issues

In a tech-driven world, both Android and iOS face significant security challenges. Android's open-source nature allows flexibility but makes it more vulnerable to issues like permission misuse, rooting, and malware. iOS, while known for its security, also faces risks, including the recent Blastpass vulnerability and code execution threats. Both platforms must continually enhance their defenses to protect users. This section examines these vulnerabilities and the ongoing efforts to keep mobile devices secure.

Android Vulnerabilities

Security has always been a major concern for Android devices. With the operating system's nature as an open-source project and one of the most popular mobile operating systems in the global market, the probability of anyone modifying these devices is high. Despite the multi-layered security features and a permission-based framework operating within its security architecture, Android remains susceptible to a number of vulnerabilities (Nowfeek & Razeed, 2022). Even small errors existing in the system or from an application can be exploited by a hacker and have the information stored in the device leaked, exposing the security gaps present in Android devices and risking the privacy and safety of its users.

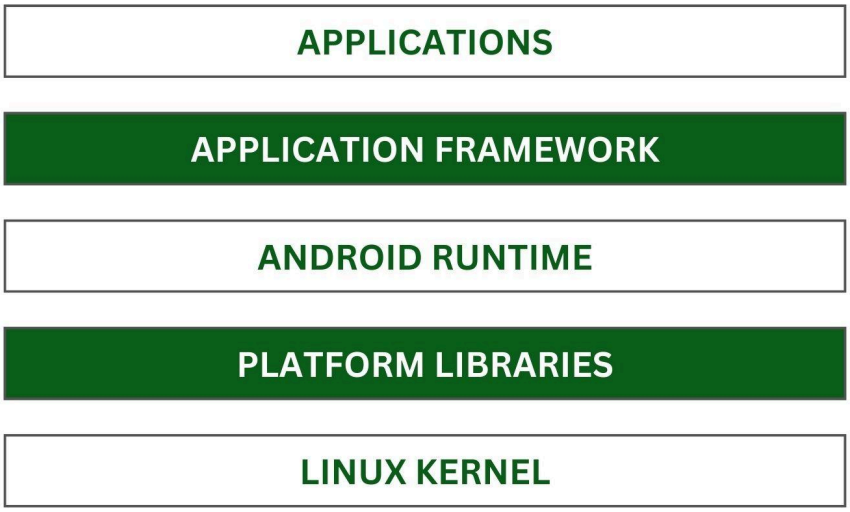

Figure 1. Architecture of Android

According to Chen's research (2021), the android architecture follows the software stack architecture wherein independent components within the system collaborate together to support the execution and implementation of the application easily. In the bottom layer of the architecture lies the Linux kernel, the basic layer of the operating system as the various drivers here are the basic functions for the Android operating system to start and work. The Linux kernel used for Androids are based on the Linux 26 kernel due to the system services the kernel provides (security, memory management, etc.). Next are the platform libraries that help implement C++ programming language to the system, which also serves as the foundation for the various components on the upper layers of the architecture. Included in here are other services for developers to work on for the application framework such as Core library, SQLite, OpenGL, and WebKit. The second and third layers are the application framework and android runtime respectively. The application framework's deals with abstraction and management on hardware access and user interface to serve for the development of the application. The Android runtime provides the base for the application framework and works well together with the platform libraries layer to power up the applications of the operating system. Lastly, we have the application layer. The significant part of this layer would be the API framework, the core mechanism for the Android platform due to the fact that it contains the core applications and everyday applications people always use.

Common Android Vulnerabilities

Application Permission

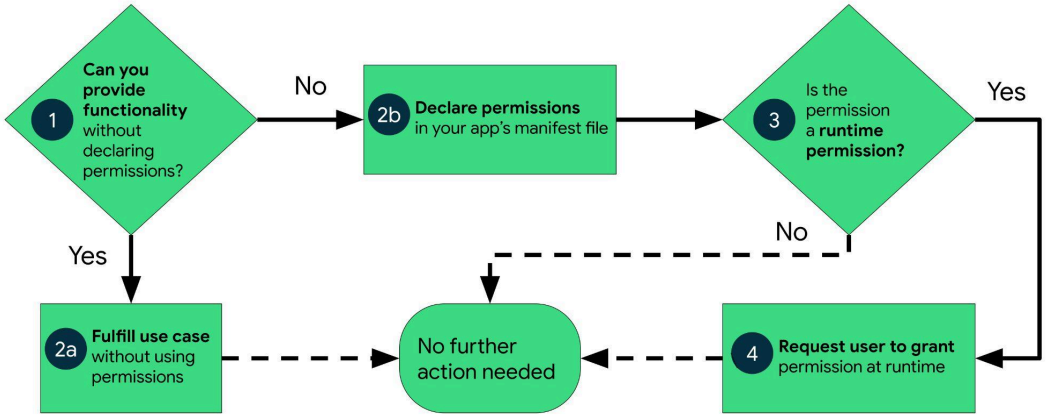

Figure 2. Android Application Permission Workflow. From *Permissions on Android,* by Android Developers, 2023 (https://developer.android.com/guide/topics/permissions/overview).

As stated before, the Android operating system is based on a permission-based framework that restricts an application from gaining access to confidential information (system network, GPS, contact information, etc.). Android permissions are arranged into three categories based on the levels of access: normal, dangerous, and signature permissions (Alrammal et al., 2022). Normal permissions are automatically granted access to requesting apps as the system deems these permissions to pose minimal risk to other applications, systems, and users. Dangerous permissions are high-risk permissions that require user permission as the requesting application asks for access to private data such as private messages, location, camera, and more, which can be potentially harmful if the application is not safe. Signature permissions are permissions that the system can give to the requesting application only if the permission has the same signature as the needed permission. All applications that are downloaded and installed will prompt the user to accept or deny specific permissions. If the user denies at least one of these permissions, there is a potential that the application may not work properly because of the lack of access to the data they need. This will then lead the user to accept every permission they request, including dangerous permissions that can leave the device in a critical state. This flaw of the

permission-based framework would be easily exploited as the hacker can access user-sensitive data from an application, resulting in information leakage (Nowfeek & Razeed, 2022).

Rooting

Rooting is a process that allows the user to have access to the superuser account, or root, of the operating system. Gaining root access to the operating system meant overriding advanced permission of the kernel and controlling all system functions (Bialon, 2020). One of the usual reasons why a user roots a phone is to gain complete control over their own device and bypass the user limitations to install an application they originally cannot install, modify certain settings, and more. There are two types of rooting that can be performed. First is soft rooting, where the user gains unlimited access to the file system of the device, which will make the Linux kernel of the system vulnerable. Second is hard rooting, where the user gains full access to the file system through flashing a custom firmware, modifying the bootloader and boot partition during the process (Blegen, 2021). Though soft rooting can be fixed by software updates, the impact of both of these rooting methods will compromise the safety of the Android ecosystem as its system functions have now been tampered with by unauthorized users. Data leakage may occur as former secured data channels are now vulnerable to the alterations made, application data and functions may be manipulated, and system behavior may differ, causing more challenges to the security of the operating system.

Malware

Malware is a common yet growing and significant threat to Android security. It is an umbrella term for the many types of malicious software variants (adware, spyware, ransomware, viruses, trojans, etc.) that can disrupt the operating system or servers of a device and impact its performance. Once malware enters the system, the operating system will undergo a massive attack that will compromise the safety of the Android environment and the user. From the analysis of Alrammal et al. (2022) on Android vulnerabilities and malware, it details the number of functionalities malware posed towards the targeted devices.

Aggressive advertisement - a well-known and widespread malware functionality wherein unwanted advertisements keep aggressively appearing on the user's affected device, impacting

device usage. Depending on the level of access the adware has on the device, it may take control of the operating system to continually bombard it with advertisements, forcefully change the default search engine of the device, and even input Plankton, a malware that secretly steals device and user information.

Remote control - a functionality of malware that makes the targeted device a bot controlled in a remote server. As a bot, information and vulnerabilities will be easily exploited by the malware and give remote control to the hacker. These malware functions are hidden in some applications as an application update in disguise, prompting the user to download them and unknowingly giving remote control to another person.

Privilege and permission escalation - a type of functionality that exploits a colluding attack with the Android applications and the operating system itself. During the collusion, malware present in the affected device may collude as well to share and escalate permissions and privileges. With this attack, users may not be able to disable privileges given to the malware, affecting device productivity.

Financial charges - malware may financially charge the users once they have privilege control over the device. From charging SMS messages to subscribing to priced deals to unknown bank transactions made in the user's account, this is where malware can potentially impact the user's life. This function is commonly found in ransomware, as the malware locks the device access toward the users, forcing them to pay an amount of money in order to unlock the device.

Information leakage - a common functionality for all malware, stealing and transferring user information to others without the user's consent. With information such as network operator, SMS messages, contact numbers, emails, and passwords exposed to the hacker and the receiver of the stolen information, cases of fraud and information breaches happened.

Detection Methods for Android Vulnerabilities

Due to the nature and vulnerabilities of Android, several mitigation strategies were researched and proposed to strengthen the security of the operating system. A conventional method that was present in most mitigation research is a legal and improved detection method. The detection methods that are made follow different analysis approaches (Sabbah et al., 2023).

The static analysis approach deals with APK feature extractions, meaning that it analyzes and extracts important information from Android files and applications. This approach is widely used in Android malware detection, with the extracted features being used to integrate it with statistical methods and detection (Meijin et al., 2021). Additional methods that are relied on in this approach include reverse engineering, signature-based analysis, permission-based analysis, resource-based analysis, and semantic-based analysis.

On the other hand, the dynamic analysis approach detects errors and threats on the system, either in hardware, software, or both. In hardware, it will collect data on the storage, CPU, battery, sensor, and camera, while in software, it will extract features from the network, application, and data access. This approach is usually used to observe application behavior to investigate the actions taken by the application using machine learning algorithms.

In contrast, the hybrid analysis approach is a combination of statistical and dynamic approaches. It observes both permission patterns and application behavior. It is the best approach from the other two, as it combines their strengths and covers the weaknesses of each other.

iOS Vulnerabilities

iOS is widely regarded as a leading operating system for its robust privacy protections and defenses against hacking, and it is often considered safer compared to the Android operating system. iOS maintains a highly controlled ecosystem where Apple manages both the hardware and software. This tight integration allows for more consistent and timely security updates directly from Apple, unlike Android, which often experiences delays due to the need for updates to pass through manufacturers and carriers (Burdova, 2022). However, despite these strengths, iOS is not immune to security threats. Vulnerabilities such as the recent "Blastpass" exploit highlight that even iOS can be targeted by sophisticated attacks (Scroxton, 2023). This underscores the importance of users staying vigilant and promptly installing security updates.

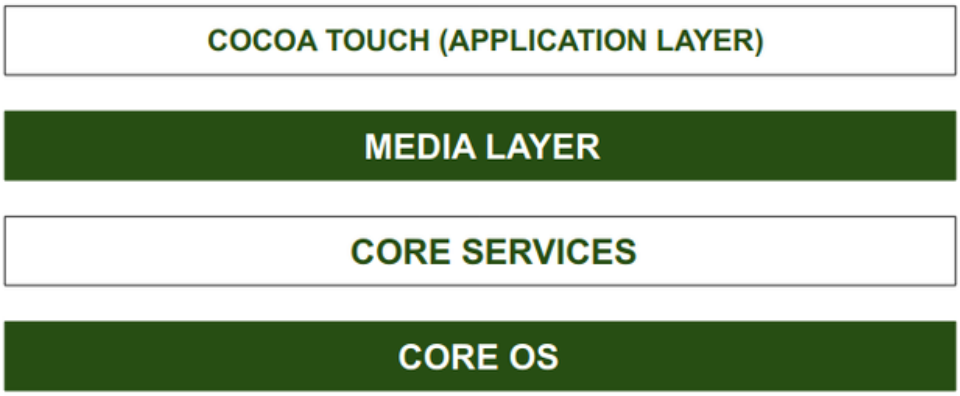

Figure 3. Architecture of iOS

The architecture of iOS is built on a layered design, which helps manage the complexity of the operating system and ensures smooth performance. At the core is the Core OS layer, which is responsible for low-level functionalities like memory management, file handling, and security. Above it lies the Core Services layer, providing essential services such as networking, databases, and inter-process communication. The Media layer handles audio, video, and graphics processing, ensuring efficient media playback and user interface rendering. Finally, the Cocoa Touch layer, at the top, offers frameworks for developing applications with rich user interfaces, including multitouch support and gesture recognition. This layered architecture allows iOS to maintain robust security and performance while offering developers a flexible environment to build feature-rich applications (Ghatmale, 2022).

Common iOS Vulnerabilities

Arbitrary Code Execution

Arbitrary code execution is a type of security flaw where an attacker can execute any commands or code of their choice on a target device (Aqua Security, 2024). This can occur due to vulnerabilities in the software that allow the attacker to bypass security protections. In iOS, arbitrary code execution can be particularly dangerous as it may allow attackers to take full control of a device, steal sensitive data, or install malicious software without the user's knowledge. For instance, Pegasus spyware leveraged this type of vulnerability to infiltrate

iPhones globally. Pegasus spyware, developed by the Israeli company NSO Group, is one of the most notorious examples of spyware that has exploited vulnerabilities in iOS. Pegasus is designed to infiltrate smartphones and turn them into surveillance devices. It can read text messages, track calls, collect passwords, track the phone's location, and harvest information from apps. The spyware can be installed on a phone without the user's knowledge, often through zero-click exploits that require no user interaction to activate. Pegasus has been used to target journalists, activists, and political figures around the world. It operates by exploiting vulnerabilities in the operating system to gain complete access to the device's data and functionalities. Once installed, it can intercept encrypted communications and use the phone's camera and microphone to spy on the user (Pegg & Cutler, 2021).

Remote Code Execution

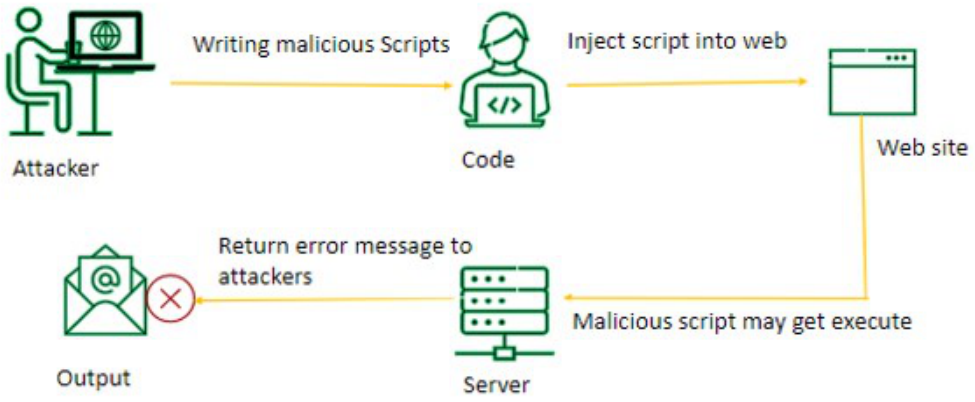

Figure 4. How Remote Code Execution Works

Remote Code Execution is a critical security vulnerability that allows attackers to execute malicious code on a target device without physical access. Remote Code Execution refers to the ability of an attacker to run arbitrary code on a remote device by exploiting vulnerabilities in the software or system. This can be achieved through various means, such as exploiting buffer overflow vulnerabilities, injecting malicious payloads via network protocols, or taking advantage of flaws in web applications or services. This vulnerability is particularly severe as it can lead to complete control over the affected system, allowing attackers to steal sensitive information, install malware, or use the device for further attacks. Despite Apple's stringent security

measures, iOS has experienced several notable RCE vulnerabilities over the years (CrowdStrike, 2023).

A recently disclosed vulnerability in iOS allowed remote code execution (RCE) triggered by malicious image files. This flaw, identified as CVE-2023-27933, exploited how iOS processes images, enabling attackers to embed harmful code within an image. When the image was viewed or processed by an iOS device, the embedded code was executed, granting attackers control over the device. The attack could lead to data theft, additional malware installation, or device control. Bitdefender highlighted this vulnerability, emphasizing the need for regular updates, robust security solutions, and safe browsing practices to mitigate such threats. Apple and cybersecurity experts recommend installing updates promptly, using comprehensive security software, and exercising caution with unknown sources (Arsene, n.d.).

Blastpass

The Blastpass vulnerability is a significant security concern that has drawn attention due to its potential impact on Apple devices. Blastpass is an advanced form of exploit that allows attackers to compromise an iPhone or other Apple devices through maliciously crafted image files, specifically using the WebP format. The vulnerability can be exploited without requiring any interaction from the user, making it particularly dangerous. According to Cluley (2023), government agencies have issued warnings urging users to secure their iPhones against spyware attacks that exploit this vulnerability. The exploit can be delivered through messages, email, or other communication channels that support WebP images. Once the malicious image is processed by the device, the attacker gains the ability to execute arbitrary code, potentially leading to data theft, unauthorized access, or further compromise of the device. Efijemue (2023) highlights the seriousness of Blastpass, noting that the exploit bypasses many of Apple's built-in security measures. This zero-click vulnerability allows attackers to gain control over the device remotely, often without the user's knowledge. The vulnerability affects not just individual users but also organizations, making it a critical concern for cybersecurity professionals who manage Apple devices in enterprise environments.

Magnet Forensics (2023) discusses the technical aspects of detecting Blastpass, specifically within WebP files. The analysis reveals that the exploit can be embedded within

these image files, making traditional security measures insufficient. The detection process involves forensic analysis techniques to identify the malicious code embedded in the WebP file format, which is not typically scrutinized in standard security checks. This highlights the need for enhanced detection methods to safeguard against such sophisticated attacks (Magnet Forensics, 2023).

Jailbreaking

Jailbreaking refers to the process of removing software restrictions imposed by Apple on iOS devices, enabling users to gain root access to the operating system. This allows the installation of unauthorized applications, customization of the device, and removal of pre-installed apps that are otherwise protected by the system's security protocols. According to Laayu et al. (2022), jailbreaking can significantly enhance the forensic acquisition process by enabling more comprehensive data extraction, as demonstrated in their study on iPhone 7 Plus running iOS 14.8.1. The research revealed that jailbroken devices allowed forensic tools to access a broader range of data, preserving the integrity of the data acquired.

However, the practice of jailbreaking comes with significant security risks. Wu et al. (2021) highlight that jailbreaking compromises the inherent security features of iOS, making devices more vulnerable to malware, unauthorized access, and instability. Despite these risks, jailbreaking remains a popular method among users seeking greater control over their devices. Forensic investigators also utilize jailbreaking techniques to recover digital evidence that would otherwise be inaccessible, providing critical insights into legal investigations.

Apple's Effort to Stop Security Vulnerabilities

Apple has implemented several measures to mitigate security vulnerabilities across its platforms, emphasizing rapid response and proactive protection. One key initiative is the introduction of Rapid Security Response updates, which are compact updates designed to address critical security issues between major software releases. This approach has been instrumental in patching zero-day vulnerabilities, such as the WebKit bug that allowed arbitrary code execution through malicious web content (Gatlan, 2023).

Additionally, Apple has focused on preventing fraudulent activities within the App Store. In 2023 alone, Apple prevented over $1.8 billion in potentially fraudulent transactions and stopped more than 3.5 million stolen credit cards from being used. These efforts are part of a broader strategy to maintain the integrity and security of the App Store, ensuring that both users and developers are protected from malicious activities (Apple, 2024a). To further enhance security, Apple employs a combination of advanced technology and human review. For example, Apple's App Review team, consisting of over 500 experts, evaluates every app submission to detect and prevent the distribution of harmful software. This rigorous review process helps maintain a high standard of security and user trust in the App Store. Apple also places a strong emphasis on the security of the App Store. All apps are thoroughly reviewed before being allowed on the App Store, ensuring they meet strict security standards and do not contain malicious code. Furthermore, Apple uses machine learning algorithms to continuously monitor apps for unusual behavior, which helps to quickly identify and mitigate potential threats (Apple, n.d.-a). Apple has been proactive in addressing security vulnerabilities through a combination of technological advancements and best practices. One of the key measures includes the introduction of hardware security features like the Secure Enclave, which is used to manage encryption keys and authenticate biometric data. This hardware-based security ensures that sensitive information, such as Face ID and Touch ID data, is stored securely and is not accessible to the main operating system or any applications (Apple, n.d.).

Rapid Security Response

Rapid Security Response (RSR) is an innovative security update mechanism introduced by Apple to provide users with critical security patches more efficiently than traditional software updates. Unlike standard updates, which are bundled with various features and enhancements, RSR focuses solely on addressing significant security vulnerabilities that may arise between major OS releases. According to Apple's support page, RSR updates are designed to be lightweight and can be deployed without requiring the user to restart their device. This approach ensures that users are protected from emerging threats as quickly as possible without disrupting their normal device usage. These updates are automatically installed on users' devices if they have automatic updates enabled, and they can be manually triggered by users who prefer to manage updates themselves (Apple Support, 2024). The significance of RSR lies in its ability to

swiftly mitigate potential security risks by providing targeted patches that address specific vulnerabilities. As detailed by XDA Developers, RSR updates are distinct from regular iOS updates in that they can be rolled back or removed if necessary. This flexibility allows Apple to promptly address any issues that might arise from the RSR update itself, ensuring the device's security and stability (Weil, 2023).

Apple Security Bounty

The Apple Security Bounty program incentivizes security researchers to discover and report vulnerabilities in Apple products. Offering rewards between $25,000 and $1,000,000 depending on the severity of the issue, this initiative helps Apple improve the security of its devices and software by addressing potential threats promptly. Researchers can submit their findings through Apple's dedicated platform, with eligible vulnerabilities earning recognition and financial rewards (Apple, 2024b).

Analysis and Discussion

The Android operating system presents a number of security challenges, primarily due to its open nature and broad user base, even if the Android operating system offers flexibility and extensive functionality to the user. This means that application permission on Android is a double-edged sword. To clarify, application permission in Android provides necessary functionality; however, it can simply be misused by careless or malicious apps downloaded by the user. This, in effect, calls for heightened user awareness and stringent permission management by developers. Another security challenge with the Android operating system is rooting. While rooting is appealing to tech-savvy users seeking more control, it significantly undermines the security framework of the device, making it a prime target for malicious activities, especially for less tech-savvy or normal users. To add to that, these malicious activities can lead to the leakage of a user’s personal data, which is valuable to data brokers. Moreover, data brokers are people, whether an individual or a group of people, that collect and sell personal and confidential data to companies that deem the data as valuable.

Furthermore, another security challenge for the Android operating system is colluding attacks. Colluding attacks complicate the security landscape by exploiting inter-app

communication, revealing the need for more robust inter-application security protocols. For the last security challenge for Android, it is malware. Malware remains a persistent threat, exacerbated by the vast number of third-party app stores and the ease with which users can install potentially harmful applications. Note that malware isn't just a security challenge for Android but also for other operating systems such as Windows and iOS but the accessibility of third-party apps makes Android more vulnerable than iOS. Likewise, these vulnerabilities highlight the importance of continuous security improvements, user education, and cautious app management to maintain the integrity and safety of the Android ecosystem.

Given its higher vulnerability to malware due to its openness to third-party apps, one of the proposed methods to mitigate this is through detection methods. To clarify, the detection method is a way to identify potential or possible security issues in Android apps. There are three common detection methods: static, dynamic, and hybrid methods of analysis. Note that there are other detection methods, such as manual testing, fuzz testing, behavioral analysis, and security scanners, but the three common detection methods mentioned above were selected due to their high accuracy rating of 95% in categorizing the common weakness enumeration (CWE).

For static analysis, its primary advantage is its ability to find vulnerabilities in the software development life cycle, thus reducing the cost of fixing these issues. However, static analysis may generate false positives and might not detect runtime-specific vulnerabilities. Likewise, the main strength of dynamic analysis is its ability to detect runtime vulnerabilities that static analysis cannot uncover. Additionally, it can offer a more precise evaluation of the application's security by mimicking real-world attacks. However, dynamic analysis can be time-consuming and may miss vulnerabilities that only appear under specific conditions not covered during testing. Hybrid analysis combines both static and dynamic analysis to leverage the strengths of each method while mitigating their weaknesses. While hybrid analysis can be more resource-intensive and complex to implement, it provides a balanced and effective means of identifying a wide range of vulnerabilities, making it a preferred choice for comprehensive security testing.

On the topic of iOS, arbitrary code execution and remote code execution (RCE) are critical vulnerabilities that demand attention. To clarify, arbitrary code execution occurs when an

attacker exploits a vulnerability in the system to execute any command or code of their choice, potentially leading to severe consequences such as data theft, system crashes, or unauthorized access. On iOS, such vulnerabilities can stem from various sources, including insecure applications, improper input validation, or flaws in the operating system itself. On the other hand, remote code execution extends this threat by allowing an attacker to execute code from a remote location, amplifying the potential impact as it can be initiated without physical access to the device. As such, RCE can lead to full system compromise, data breaches, and the spread of malware, which raises both professional and ethical issues for everyone involved with iOS.

In terms of jailbreaking, while often perceived as a method to gain more control over iOS devices by bypassing Apple's restrictions, it opens up additional vulnerabilities. This is because by removing the device's security layers, jailbreaking can expose the system to unauthorized code execution, malware, and other security threats. While jailbreaking offers users increased customization and the ability to install unauthorized applications, it significantly undermines the security model put in place by Apple, leaving the device susceptible to various exploits, such as data theft.

Given several notable security incidents in iOS, such as XcodeGhost and KeyRaid malware in 2015, Apple takes security incidents seriously and makes significant efforts to address and mitigate security vulnerabilities across its ecosystem, particularly with iOS. Central to these efforts is the company's robust approach to software updates. This implies that Apple regularly releases updates that patch security flaws, often addressing vulnerabilities before they can be widely exploited, especially for iOS users. This proactive approach is complemented by a comprehensive security architecture that includes features like sandboxing, which isolates applications to limit the damage that can be caused by a compromised app, and mandatory code signing, which ensures that only trusted software can run on iOS devices. Furthermore, Apple has established rigorous app review processes to scrutinize applications before they are made available on the App Store, reducing the risk of malicious software being distributed to users. One of Apple’s actions to protect iOS users from data compromise is the introduction of features like App Transport Security (ATS), which mandates the use of secure connections for app communications, enhancing the overall security posture of iOS applications.

Moreover, Apple also engages with the security research community through its bug bounty program, offering significant financial incentives for researchers who discover and responsibly disclose vulnerabilities. This collaboration helps Apple identify and address security issues more swiftly and effectively. In addition, the company employs advanced security technologies such as Secure Enclave, which provides a higher level of security for sensitive operations like biometric data processing. Education and transparency are also part of Apple's strategy. By publishing detailed security guides and supporting documentation, Apple helps developers adhere to best practices in security. Additionally, regular security advisories and detailed release notes keep the user and developer communities informed about the latest security updates and vulnerabilities.

From the previously mentioned issues and analysis regarding the current security vulnerabilities of Android and iOS, it is seen that the two still have their own shortcomings despite being the two most popular mobile operating systems. The main argument regarding the vulnerability of Android is that it is an open-source platform, which makes it easier to attack. It is observed that as Android makes room for innovation, it opens security vulnerabilities that lead to data leakage, malware, and other issues. In contrast, numerous studies have pointed out that iOS' close-source platform restricts developers from causing potential attacks. However, iOS still has its fair share of vulnerabilities, compromising the safety and security of its users. Additionally, due to its 'close-source' nature, numerous users have bypassed the software restrictions that were imposed to customize their devices.

Moreover, these two mobile operating systems were stated to be part of "extensive third-party tracking," wherein the activities of the users were collected, which raises numerous concerns regarding the privacy of the users. Hence, no one can claim that any of the discussed operating systems are totally free from security vulnerabilities or attacks. Android and iOS developers must adhere to ethical and moral obligations, prioritize users, and protect user data against threats. Despite that, private user information remains at risk as technological advancements are introduced since unethical programmers still try to evade the mitigation techniques introduced by the developers. Thus, there is a constant effort to strengthen the security of the two leading mobile operating systems by ensuring that their development will result in the betterment of society and avoid the commoditization of user data.

Conclusion

Android and iOS are constantly battling neck-to-neck regarding which mobile operating system is better than the other. Some would say Android, while others would justify that iOS is more secure than the other. However, after rigorously comparing the two, it can be seen that there is no linear answer since both OS are subjected to threats and vulnerabilities.

As established in the previous sections, Android's open-source nature attracts unethical hackers to exploit the vulnerabilities that they see in order to gain unauthorized access to the private information of its users. Three main Android vulnerabilities are discussed in the paper: Application Permission, Rooting, and Malware. In hindsight, it is not noticeable that Application Permission is an issue since it asks for the consent of the user whether or not they allow the application to the things that are listed. However, once the user chooses to restrict some of the application's movement by denying an action, it is highly likely that it will not perform as intended. Hence, the user will now be forced to accept all permissions, regardless of whether or not it is dangerous. The second issue discussed is called "Rooting," and as the name suggests, it allows users to access the 'root' of the operating system. Rooting is commonly observed if the user wishes to modify their device, bypassing the limitations imposed by the developers. Then, "Malware" is an umbrella term encompassing various types of malicious software variants that compromise the safety of the Android environment and the user.

On the other hand, iOS may have implemented enhanced security due to its closed-source nature, but it is still not foolproof. The three iOS issues discussed in the paper are as follows: Arbitrary Code Execution, Remote Code Execution, and Jailbreaking. Arbitrary Code Execution is a security flaw wherein an attacker is able to bypass security protections, enabling them to execute commands on their target device. This iOS vulnerability should be addressed since it can be used to steal user data or even take full control of the device, even if the owner remains unaware of it. Moving on to the second issue is "Remote Code Execution," which allows attackers to run malicious code on a remote device without direct physical access. Next, the third issue raised is "Jailbreaking," which is similar to "Rooting" in Android. Due to the strict restrictions imposed by Apple, some users resort to jailbreaking in order to install unauthorized

applications and customize their devices, resulting in their devices being vulnerable to malware and unauthorized access.

Thus, the developers of the two operating systems continuously employ mitigation techniques that will ensure that the safety and privacy of their users will not be compromised. The detection methods employed to search for Android vulnerabilities are static analysis, dynamic analysis, and hybrid analysis. The static analysis method examines and evaluates the source code of an Android application without executing it. In contrast, dynamic analysis observes the behavior of the system and investigates its actions to detect errors and threats. Then, the hybrid approach combines the two aforementioned approaches to observe both permission patterns and application behavior. In comparison, Apple employed Rapid Security Response (RSR) updates to address vital security issues every time a major software was released. The main goal of RSR is to provide patches and fix updates to users once a security issue is seen. Additionally, Apple has implemented a combined technology and human review, which works together to evaluate every submitted application before distributing it to the masses. Through this, they were able to prevent deceptive activities within the App Store and mitigate potential threats.

Computer Engineering students must observe studies regarding operating system security research and development to be aware of the potential vulnerabilities that can infiltrate the devices students commonly use. In turn, this will enable them to find tools and be equipped with enough knowledge that will help them identify if their private information has been compromised. Additionally, the future studies or line of work of the student can be in line with the development of stronger encryption methods or finding solutions to mobile operating systems vulnerabilities. Hence, as professionals, computer engineers will be more equipped to combat the threats that arise from every advancement that is introduced to the operating system. In addition, professionals have more credibility in educating and promoting the masses regarding the common threats that they may encounter on a daily basis whenever they use their phones. Thus, aside from finding solutions, professionals may also help by providing knowledge on how to protect one’s data and enhance the device’s security.

As people's lives become more intertwined with their personal mobile devices, there is an increase in security concerns. The two popular mobile operating systems are Android and iOS, and each has its own vulnerabilities that developers are trying to address. Hence, there's a need for their users to be aware of the threats that are looming over their devices since they can cause unsafe Android or iOS environments for them. Current computer engineering students and professionals must ensure that they uphold ethical conduct by serving the masses and sharing their knowledge regarding the vulnerability of their phone's mobile operating system, which prevents others from having their security compromised.

References

Ahmed, O. M., & Sallow, A. B. (2017). Android Security: A Review. *Academic Journal of Nawroz University*, 6(3), 135-140. doi: 10.25007/ajnu.v6n3a97

Almisreb, A., Mulalić, H. H., Mučibabić, N., & Numanović, R. (2019). A review on mobile operating systems and application development platforms. *Sustainable Engineering and Innovation ISSN 2712-0562*, *1*(1), 49–56. https://doi.org/10.37868/sei.v1i1.94

Alrammal, M. a. M., Alrammal, M. N. M., Naveed, S. S. M., & Sallam, G. T. S. (2022). A critical analysis on Android vulnerabilities, malware, anti-malware and anti-malware bypassing. 網際網路技術學刊, *23*(7), 1651–1661. https://doi.org/10.53106/160792642022122307019

Apple. (n.d.-a). *App security overview*. Apple Support. Retrieved July 26, 2024, from https://support.apple.com/en-ph/guide/security/sec35dd877d0/1/web/1

Apple. (n.d.-b). *Introduction to Apple platform security*. Apple Support. Retrieved July 26, 2024, from https://support.apple.com/en-ph/guide/security/seccd5016d31/web

Apple. (2024a, June 3). App Store stopped over $7 billion in potentially fraudulent transactions in four years. *Apple Newsroom*. https://www.apple.com/newsroom/2024/05/app-store-stopped-over-7-billion-usd-in-potentially-fraudulent-transactions/

Apple. (2024b). *Apple Security Bounty*. Retrieved from https://security.apple.com/bounty/

Apple Support. (2024). Rapid Security Responses. Retrieved from https://support.apple.com/en-ph/102657

Aqua Security. (2024, April 9). *Arbitrary Code Execution: 6 attack examples and mitigation steps*. Aqua. https://www.aquasec.com/cloud-native-academy/cloud-attacks/arbitrary-code-execution/

Arsene, L. (n.d.). *IOS vulnerability allows remote code execution triggered by image files*. Hot For Security. https://www.bitdefender.com/blog/hotforsecurity/ios-vulnerability-allows-remote-code-execution-triggered-by-image-files/

Bialon, Raphael. (2020). On Root Detection Strategies for Android Devices.

Burdova, C. (2022, December 7). *Android vs. iOS Security: Are They Equal?* Android Vs. iOS Security: Are They Equal? https://www.avg.com/en/signal/android-vs-ios-security

Chen, Y. (2021). Research on Android architecture and application development. *Journal of Physics Conference Series*, *1992*(2), 022168. https://doi.org/10.1088/1742-6596/1992/2/022168

Cluley, G. (2023, June 28). Government Agencies Told to Secure iPhones Against Spyware Attacks. Retrieved from https://www.tripwire.com/state-of-security/government-agencies-told-secure-iphones-against-spyware-attacks

CrowdStrike. (2023, October 5). *What is Remote Code Execution (RCE)? - CrowdStrike*. crowdstrike.com. https://www.crowdstrike.com/cybersecurity-101/remote-code-execution-rce/

Efijemue, A. (2023, June 29). Blastpass: A New Vulnerability Could Put Your Apple Device at Risk. LinkedIn. Retrieved from https://www.linkedin.com/pulse/blastpass-new-vulnerability-could-put-your-apple-device-efijemue

Fu, Xingbing & Jiang, Chaofan & Li, Chaorong & Li, Jiangtao & Zhu, Xiatian & Li, Fagen. (2024). A hybrid approach for Android malware detection using improved multi-scale convolutional neural networks and residual networks. Expert Systems with Applications. 249. 123675. 10.1016/j.eswa.2024.123675.

Ghatmale, P. (2022, April 14). iOS Architecture. Medium. https://medium.com/@pranavghatmale/ios-architecture-614b7a10ceaf

Gatlan, S. (2023, July 11). Apple releases emergency update to fix zero-day exploited in attacks. *BleepingComputer*. https://www.bleepingcomputer.com/news/apple/apple-releases-emergency-update-to-fix-zero-day-exploited-in-attacks/

Goel, M., & Singal, G. (2021, April). *Android OS Case Study.* ResearchGate. https://www.researchgate.net/publication/350992546_Android_OS_CASE_STUDY

Holmberg, A. (2022). *iOS vs Android: Security of Inter-App Communication* [Mid Sweden University]. https://www.diva-portal.org/smash/get/diva2:1691563/FULLTEXT01.pdf

Huang, D., & Wu, H. (2018). Mobile Cloud Computing. ScienceDirect. https://www.sciencedirect.com/topics/computer-science/android

Janaka S., Harsha K., Mhd Omar Al-Kadri, Piras, L., & Petrovski, A. (2023). Labelled Vulnerability Dataset on Android Source Code (LVDAndro) to Develop AI-Based Code Vulnerability Detection Models. OpenAIR@RGU (Robert Gordon University). https://doi.org/10.5220/0012060400003555

Kollnig, K., Shuba, A., Binns, R., Van Kleek, M., & Shadbolt, N. (2022). Are iPhones really better for privacy? A comparative study of iOS and Android apps. *Proceedings on Privacy Enhancing Technologies*, 2022(2), 6-24. https://doi.org/10.2478/popets-2022-0033

Nowfeek, M., & Razeed, M. (2022). A Review of Android operating system security issues. International Journal of Research and Scientific Innovation, 9(1), 26-30.

Magnet Forensics. (2023, June 22). How to Detect Blastpass Inside a WebP File. Retrieved from https://www.magnetforensics.com/blog/how-to-detect-blastpass-inside-a-webp-file/

Meijin, L., Zhiyang, F., Junfeng, W., Luyu, C., Qi, Z., Tao, Y., Yinwei, W., & Jiaxuan, G. (2021). A Systematic overview of Android malware detection. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2021.2007327

Mickle, T. (2023, September 11). As Smartphone Industry Sputters, the iPhone Expands its Dominance. NYTimes. https://www.nytimes.com/2023/09/11/technology/apple-iphone-17.html

Pegg, D., & Cutler, S. (2021, July 20). What is Pegasus spyware and how does it hack phones? *The Guardian*. https://www.theguardian.com/news/2021/jul/18/what-is-pegasus-spyware-and-how-does-it-hack-phones

Permissions on Android. (2023, April 12). Android Developers. https://developer.android.com/guide/topics/permissions/overview

Quissanga, F.C. (2023). Comparative Study of Information Security in Mobile Operating Systems: Android and Apple iOS. IntechOpen. doi: 10.5772/intechopen.109652

Rafalski, K. (2024, April 29). The Complete History of iOS. Netguru. https://www.netguru.com/blog/ios-history

Sabbah, A., Taweel, A., & Zein, S. (2023). Android Malware Detection: A Literature Review. *Communications in Computer and Information Science*, 263–278. https://doi.org/10.1007/978-981-99-0272-9_18

Scroxton, A. (2023, September 8). Apple patches Blastpass exploit abused by spyware makers. *ComputerWeekly.com*. https://www.computerweekly.com/news/366551552/Apple-patches-Blastpass-exploit-abused-by-spyware-makers

Umar, A. U., & Wakili, A. (2023). A Comparative Study of Modern Operating Systems in terms of Memory and Security: A Case Study of Windows, iOS, and Android. SLU Journal of Science and Technology, 6(1&2), 131-138.

Weil, S. (2023, June 10). Apple's Rapid Security Response: Everything You Need to Know. XDA Developers. Retrieved from https://www.xda-developers.com/apple-rapid-security-responses/

Individual Work Breakdown

Name	Assigned Task	Work Contribution (%)
ARINGO, MICHAEL JUDRICK M.	4 Review of Related Literature, iOS issues	25%
BANAL, KENAN A.	3 Review of Related Literature, Individual Analysis of Android and iOS issues	25%
CONCORDIA, JHYREL JULIENE R.	Background of the Study, 1 Review of Related Literature, Android issues	25%
TEVES, FAITH SHAREINE V.	Significance of the Study, 1 Review of Related Literature, Analysis of the two OS, Conclusion	25%