Advanced Research on Information Systems Security

ISSN: 2795-4609 | ISSN: 2795-4560
Print & Online

# Case study to identify vulnerabilities in applications developed for the Android

Lima, Tatiani *

*ISLA – Polytechnic Institute of Management and Technology, Vila Nova de Gaia, Portugal*
*Email: tatiani@vidadeprogramadora.com*

**Abstract**

The growing use of mobile devices has caused many developers to focus more on design and user experience, but with this neglected security issues, whether due to lack of knowledge in this field or lack of delivery time, thus exposing thousands from users to information leaks among other malicious actions. In this sense, this work aims to expose the main vulnerabilities that impact the security of a mobile application, going through analyzes in mobile applications, with the intention of alerting developers about the flaws that are usually present in the applications due to bad coding practices and to reflect on how to make your apps more secure.

***Keywords:*** vulnerabilities, OWASP, Kali Linux, mobile applications.

* Corresponding author. Email address: tatiani@vidadeprogramadora.com

## 1. Introduction

Mobile applications have been developed for different purposes impacting the daily lives of people in different ways, generating daily dependence on this type of technology. According to [1] Anacom - National Communications Authority in Portugal there was a growth of 12.4% in the number of mobile internet users and 23.2% in the use of internet access through PCs, tablets, pen and routers. It was found that this significant increase is the largest recorded since 2010 and may be associated with the Digital School Program launched in 2020, by

providing internet connection equipment such as hotspots and SIM cards, so that students and teachers could have access to digital platforms. In addition, a 39.8% increase in mobile broadband access traffic was observed compared to 2021. This data demonstrates that mobile devices are always active and are therefore valuable targets for malicious elements [2].

The present work aims to identify the main vulnerabilities in applications developed without proper security care, presenting, discussing, and describing the processes performed during data collection using forensic analysis tools available in operating systems such as Kali Linux, with the aim of demonstrating that such vulnerabilities facilitate the action of cybercriminals, compromising data of people and companies.

***Structure***

This section aims to review the literature to identify and summarize studies related to the theme of this article, so that we can compare and point out evolutionary points related to the scenarios that will be tested later, having aligned the same view that proposes the work of Bulbukayr and Almaiah [3] where they point out that it is the responsibility of researchers to educate and inform users and developers of applications for smartphones about the possible security problems and how to avoid them.

In the section that deals with the identified vulnerabilities, begins by presenting authentication and insecure authorization, addresses the hijacking of user session and in the insecure storage section is the hijacking of permissions on components that belongs to healthy applications and that is directly related to the article published as "Direct Resource Hijacking in Android" [4] , where the authors propose an access control structure and also the use of certificates, something suggested also in the article [5] "Network Security Challenges in Android Applications", where we evaluated the use of SSL implementation incorrectly in applications specifically of the Android platform, either by the lack of knowledge of developers, as well as users. Still within this theme, another challenge pointed out by the authors mentions the flexibility as Google treats implementation of certain features, this represents significant security challenges, such as: the non-mandatory SSL implementation, allow all host names, trust all certificates, among other factors described in the work. One of several solutions pointed out would be to verify that a trusted certificate has been issued to the address to which the application will connect, that is, not accepting certificates that have been issued to other domains.

Next is the injection subsection, related to the study by Shalan, Shahriar and Rahman [6] that examined various methods on how to deal with vulnerabilities based on XSS attacks, particularly hybrid applications developed for the Android platform. The article presents a comparison of existing detection tools features present as compression and obfuscation.

They also addressed the theme XSS, Chin and Wagner attacks [7] and the work presented by Nisha and Bhanu [8], focusing on Android applications using WebView and IOS applications that used UIWebView in the development of mobile applications, the study also addressed how efficient existing solutions are in this type of attack.

An institution that organizes and catalogs problems such as those mentioned above is the Open Web Application

Security Project, popularly known as OWASP, publishing key security risks in various fields. This paper considers two lists: the most recent one published in 2021 [9] that carries ten risks that affect software security in general and that will serve to observe the risks inherent in the backend of a mobile application, the other list that dates from 2016 [10] but that provides a list of vulnerabilities focused on mobile applications, this data will be used as a reference and will specifically process, the description of which will be detailed below, such as: misuse of the operating system, unsafe data storage, insecure communication, insecure authentication, insufficient encryption, insecure authorization, client code quality, code tampering, reverse engineering, and foreign functionality.

## 2. Experimental Setup

The description of this scenario aims to enable other individuals to reproduce the results obtained in this study. The work begins by creating a new emulator with the Genymotion program [11], selecting the following settings: Google Nexus 9, Android 6, 1500Mb Ram, network bridge mode by selecting the network card to which you want to get the ip.

For the test was used a virtual machine from oracle VM Virtualbox with kali linux operating system [12]. This operating system provides mobile forensic analysis tools that will be detailed in the following sections. As android emulator will be used the GenyMotion that will allow access to the ADB, emulating a variety of mobile devices. The ADB (Android Debugging Bridge), in various situations, will enable communication between the computer and the device via the command line.

For the virtual machine that will house Kali Linux, the following settings are being used: 8Gb Ram and 85Gb of hard disk space, with additional installation of the VirtualBox Guest Additions Installation tools, the network connection must also be with the network card in bridge mode. Mobile penetration testing is an attempt to assess the security of a mobile app by exploiting vulnerabilities, including decompilation, real-time analysis, and security testing.

To complement the analyses of this work, among the applications chosen, is the Diva application [13], an application intentionally developed to be unsafe, as an object of study for professionals and students in the area of information security. Thus, within this case study, the tests described below were performed.

## 3. Vulnerabilities Identified

### *Unsafe Authorization*

Can be confused understand insecure authorization and insecure authentication, but they are different concepts: authentication aims to identify an individual and in this case the attacker has already authenticated (even if it is with the login of a user with fewer privileges), while the authorization is the act of ascertaining whether the identified individual has the necessary permissions to access certain functionalities, in this case the attacker exploits functionalities that that user should not be authorized to use either by raising user privileges or by

searching for features that are authenticated but have a weak or nonexistent authorization scheme [14].

In this case the attacker can make a legitimate login and can take advantage when the application is in offline mode, at which point the attacker tries to perform some privileged functionality using a session token in the POST / GET requests on the backend server. When identifying missing authorizations, it then performs functionality that should not be entitled as an authenticated user [15].

Another technique applied to the application backend is the hijacking of a session token from an already authenticated user, commonly intercepted in the HTTP headers, so the attacker can break session management without having knowledge of the user's credentials, in case of success in hijacking an administrator's session so the attacker is authorized to travel through all application resources [16].

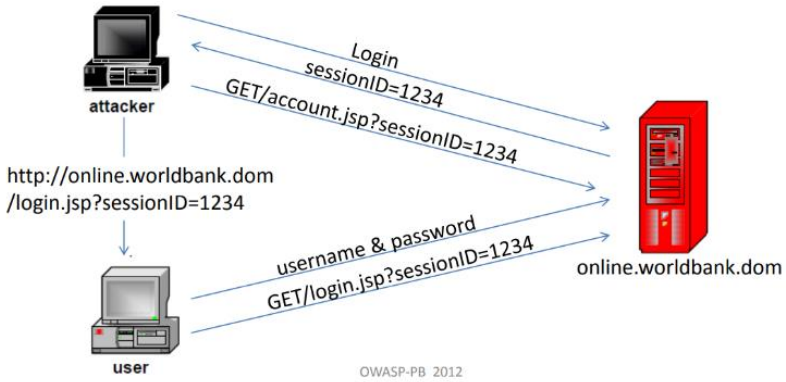


**Figure 1:** OWASP Session Fixation Attack [17]

To prevent insecure authorization, it is important to always verify the roles and permissions of an authenticated user, the authorization should not depend on the permissions coming from the mobile device. The user accesses the application, which in turn accesses the backend through a Rest API, informing its user id and token. The attacker intercepts this action and includes the id of a user known as part of the received URL and their respective access token as a default header in the request. The backend checks for the presence of the token, but fails to parse the user id, as a result the user can enter the user ID they want in the url and get account information from other users as part of the Api Rest request, example: access a url that makes the user's profile edit: profile/edit/99, 99 being the user id that the attacker wants to fetch.

***Insecure Authentication***

The attacker can identify a vulnerable authentication through automated attacks, and by using tools it can spoof or ignore authentication. Finding loopholes when the backend API service does not provide an access token, if the mobile app stores passwords locally on the device if the app uses a weak password policy with short and bad passwords, and if at some point the application could not detect the identity of the user, this action will favor the

attacker who can anonymously perform a sensitive functionality [2].

In mobile apps users are not expected to be online throughout the session time, often have less reliable connections, and the developer should predict the offline authentication time in the application.

When the developer does not include a control to lock the user's account after numerous invalid authentication attempts, the attacker may attempt to authenticate themselves using the brute force technique, using the trial and error method with all possible passwords, whether or not to include a dictionary of words until the attack is successful [14].

Using the Diva app, the Hardcoding Issues option was triggered. Through JD-Guid was decompiled the application code and could be observed authentication with the password "hardcoded", that is, the password was fixed in the code, through reverse engineering, the attacker can easily obtain this password and access the system. Authentication information such as passwords and keys should not be put in the code.

***Insufficient Encryption***

Encryption aims to make information unintelligible in order to provide the confidentiality of the information. According to Nelson [15], often the security of the solution as a whole can be compromised by failure of cryptographic mechanisms.

Anyone with physical access to data that uses weak encryption algorithms or fails in the encryption process can be a potential threat, successfully returning sensitive information. In addition, the way the Android operating system itself checks encryption varies from device to device [2].

The use of the following encryption algorithms is not recommended because they are pointed out by Owasp [9] as unsafe and/or obsolete: MD5, RC2, MD4, and SHA1. In addition to these, the Data Encryption Standard (DES) algorithm and RSA with keys smaller than 1024 bits should also not be used [15].

However, it is not enough to just use the best algorithms if it is not done correctly the management of your keys, among the problems encountered are:

- Weak key management processes
- Use of encoded keys in binary
- Use of unsafe and/or obsolete algorithms
- Key interception through attacks

The use of cryptographic hash functions has been used to protect user passwords, but the security of this mechanism has been questioned [15] as the quality of the password chosen by the user may be weak. In this scenario the attacker creates a dictionary of commonly used words and stores their respective hashes, for later use in access to the system. To prevent this from happening, it is recommended that the developer make use of salts

at the end of each password, before calculating the hash.

***Injection***

Poor programming practices can lead to memory leaks and buffer overflows, which can cause attackers with greater low-level knowledge to exploit these vulnerabilities. DOM-based XSS attacks on mobile applications using Webview also come from low code quality. Webview-based applications have browsers that allow developers to write HTML and Javascript code within the platform, but while it is convenient to program in this way, these custom browsers can also pose a threat to the application.

According to research conducted by Chin and Wagner (2013), among 608 webview-based searched applications, 20% of them offer the potential to grant websites access to code, 54% allowed a user to enter Malicious Javascript into the application code [7].

Using the Diva application, it was possible to verify this problem through the "Input Validation Issues" option, which introduced a URL that directed to a malicious javascript script. Using adb commands, such as the Adb Shell command was included script in the mnt folder that was loaded into the application: file//mnt/test.txt

To avoid this type of attack [8], the developer must appropriately sanitize inputs received from the client side by removing characters, terms related to HTML, Javascript, or SQL code from the input.

***Unsafe Design***

In the top 10 vulnerabilities of OWASP published in 2021 came unsafe design as a new category. It works differently from unsafe implementation since unsafe design is focused on risks related to design and architecture flaws [9]. As an attack scenario, an example we cite is a workflow where credential retrieval is focused on questions and answers, something definitely prohibited by OWASP, because as proof of identity anyone could know the answer, so this method should be replaced by a more secure design, such as multiple authentication factors [15].

***Unsafe Storage***

This type of failure can compromise the application's private folders, sd card, log files, application manifest files on Android, xml data storage, and other elements that make up the file system of a mobile device.

OWASP [10] mention that this type of threat may affect, for example, the user who has lost their mobile device. The attacker can connect the mobile device to a computer with tools that allow you to read the private directories of applications, which may contain personally identifiable information or other sensitive assets that are low or no encryption. Another way would be through malware aimed at modifying a legitimate application to steal this information, or it would even root on a mobile device to circumvent any encryption protection by revealing the protected data. Among the impacts of OWASP (2016), are: identity theft, privacy violation, fraud, reputational

damage, material loss.

Having the mobile device in hand it was possible to perform a penetration test of the device when using the Kali Linux terminal. With the phone connected to the same network of the computer used the command adb connect xxx.xxx.x.xx (where x is the ip that the network assigned to the device), then through the command adb Shell obtained root access (in this case of an Android mobile device), after, using the command "cd storage" I obtained access to the storage folder and, finally, the "cd mnt/sdcard" command allowed quick access to the data stored on the device's memory card, with this technique it is possible to access the folders that applications use to store important information.

For this reason, it is important that, when there is a need to store sensitive data of users in internal storage, that this data is encrypted, as is for example, the case of WhatsApp application data.

***Unsafe Communication***

Many mobile applications use the internet for client-server communication. Attackers can exploit threats through malware on the mobile device, over the network, through network devices such as routers, proxy, etc., A bad SSL configuration can facilitate phishing attacks. However, this risk also includes when other technologies that are used in the development of the mobile application, such as: the use of Bluetooth, NFC, infrared, SMS, etc [10].

For this test, Burp Proxy was used, with the purpose of intercepting and analyzing the traffic of the network. This is one of several tools that is present in the Kali Linux Penetration Test option. After setting up the Proxy, you can observe all the movement of the network in the Burp Intercept Tab, through the Params tab you can change the fields of a form and send this data directly from the proxy.

***Reverse Engineering***

In this type of attack, the attacker downloads the application into an app store and analyzes the package code within its own development environment, performing a binary analysis to verify which tables, libraries, and other elements of the source code are embedded in the application. The attacker is able to discover the information about the backend servers. While most applications are susceptible to reverse engineering, this tactic can be prevented by using glare that allows you to segment the code you want to obfuscate, adjust the degree of obfuscation so as not to impact application performance.

As mentioned earlier, you can reverse engineer most of the apps available on GooglePlay. Through Kali Linux, the apk of a PlayStore app, The Secret Menu for Starbucks, was downloaded. Usually the url provides the name of the package that is the starting point, in this case "com.sepiasoftware.secretmenuforstarbucks". This name is inserted in the search of the ApkPure site [18], which allows you to download the package of any desired application, as long as it has the name of the package. The location was chosen to download the package, then the apk file was renamed to .zip, and finally the file was uncompressed, resulting in access to the original package

files, such as AndroidManifest.Xml.

The Reverse Engineering option, dex2jar of Kali Linux, was used, which generated a Java file. The next action was to decompile the Java file using the Kali Linux JD-GUI tool that displays all the application code.

## 4. Conclusions

Although the process of vulnerability analysis encompasses other aspects such as social engineering, only the most critical aspects officially pointed out by the OWASP were addressed and analyzed. The survey considered the last survey done in 2021 that understands vulnerabilities more generically and also the list focused on mobile applications whose last was published in 2016. It is important to note that the tests performed in this work were performed legally, in a control environment, using for example, the Diva application that is open to professionals and students of information security for the study of vulnerabilities in mobile applications.

In many of the tests performed in this work, ADB was used and although it was created to allow many developers to have greater ease in debugging code, many attackers use ADB to exploit vulnerabilities of Android applications.

The Kali Linux operating system was chosen, because it is specially aimed at forensic analysis and equipped with a range of tools, these allowed these tests to be performed, whether at the time of decompiling a package of type apk, intercepting the network traffic, the system is still many other tools that were not used in this context due to the limitation of the extent of this work.

This article was intended to demonstrate that security issues in mobile applications are largely due to failures during development and provide greater reflection to application developers about the risks and vulnerabilities to which poorly implemented applications expose their users.

## Acknowledgements

We have finished one more step. Many thanks to all the teachers for teaching us and for transmitting such precious knowledge.

My thanks to my classmates, for all the times we help each other, even in difficult times. I also thank the course coordinator who motivated me to persist in the walk and it was an essential piece for me not to give up the course. To family members for their support and for understanding the moments I needed to replace leisure with learning.

## References

[1] "ServicosMoveis1T22.pdf." Accessed: Dec. 23, 2022. [Online]. Available: https://www.anacom.pt/streaming/ServicosMoveis1T22.pdf?contentId=1722575&field=ATTACHED_FILE

[2] J. J. Drake, Z. Lanier, C. Mulliner, P. Oliva Fora, S. A. Ridley, and G. Wicherski, *Android Hacker´s Handbook*. Ed. Indianapolis, IN: John Wiley & Sons, 2014.

[3] M. A. S. Bubukayr and M. A. Almaiah, "Cybersecurity Concerns in Smart-phones and applications: A survey," in *2021 International Conference on Information Technology (ICIT)*, Jul. 2021, pp. 725–731. doi: 10.1109/ICIT52682.2021.9491691.

[4] Y. Gu, Q. Li, H. Zhang, P. Su, X. Zhang, and D. Feng, "Direct Resource Hijacking in Android," *IEEE Internet Comput.*, vol. 20, no. 5, pp. 46–56, Sep. 2016, doi: 10.1109/MIC.2015.138.

[5] D. Buhov, M. Huber, G. Merzdovnik, E. Weippl, and V. Dimitrova, "Network Security Challenges in Android Applications," in *2015 10th International Conference on Availability, Reliability and Security*, Aug. 2015, pp. 327–332. doi: 10.1109/ARES.2015.59.

[6] A. Boyett, A. Shalan, H. Shahriar, and M. Asadur Rahman, "A Taxonomy of XSS Attack Detections in Mobile Environments based on Automation Capabilities," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Jul. 2021, pp. 1339–1344. doi: 10.1109/COMPSAC51774.2021.00188.

[7] E. Chin and D. Wagner, "Bifocals: Analyzing WebView Vulnerabilities in Android Applications," in Information Security Applications, vol. 8267, Y. Kim, H. Lee, and A. Perrig, Eds. Cham: Springer International Publishing, 2014, pp. 138–159. doi: 10.1007/978-3-319-05149-9_9.

[8] J. N. O.S. and S. Mary Saira Bhanu, "A Survey on Code Injection Attacks in Mobile Cloud Computing Environment," in 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Jan. 2018, pp. 1–6. doi: 10.1109/CONFLUENCE.2018.8443032.

[9] "OWASP Top Ten | OWASP Foundation." https://owasp.org/www-project-top-ten/ (accessed Jan. 04, 2023).

[10] "OWASP Mobile Top 10 | OWASP Foundation." https://owasp.org/www-project-mobile-top-10/ (accessed Jan. 04, 2023).

[11] "Genymotion – Android Emulator for app testing Cross-platform Android Emulator for manual and automated app testing." https://www.genymotion.com/ (accessed Jan. 04, 2023).

[12] "Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution." https://www.kali.org/ (accessed Jan. 04, 2023).

[13] "DIVA." https://payatu.com/damn-insecure-and-vulnerable-app (accessed Jan. 04, 2023).

[14] P. E. S. Borba, "Sistemas Legados: Uma Proposta para Centralização de Autenticação," 2017.

[15] N. Uto, Teste de Invasãode Aplicações Web. Rede Nacional de Ensino e Pesquisa – RNP, 2013.

[16] "Session hijacking attack | OWASP Foundation." https://owasp.org/www-community/attacks/Session_hijacking_attack (accessed Jan. 05, 2023).

[17] re alex, "Attacking Session Management," 2012.

[18] "Baixar APK de Forma Rápida, Gratuita e Segura no Android." https://m.apkpure.com/br/ (accessed Feb. 27, 2023).

[19] F. Alves, N. Mateus-Coelho and M. Cruz-Cunha, "ChevroCrypto – Security & Cryptography Broker," 2022 10th International Symposium on Digital Forensics and Security (ISDFS), Istanbul, Turkey, 2022, pp. 1-5, doi: 10.1109/ISDFS55398.2022.9800797.

[20] N. Mateus-Coelho and M. Cruz-Cunha, "Serverless Service Architectures and Security Minimals," 2022 10th International Symposium on Digital Forensics and Security (ISDFS), Istanbul, Turkey, 2022, pp. 1-6, doi: 10.1109/ISDFS55398.2022.9800779.

[21] Nuno Mateus-Coelho, A New Methodology for the Development of Secure and Paranoid Operating Systems, Procedia Computer Science, Volume 181, 2021, Pages 1207-1215, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.01.318.

[22] Mateus-Coelho, Nuno Ricardo, et al. "POSMASWEB: Paranoid Operating System Methodology for Anonymous and Secure Web Browsing." Handbook of Research on Cyber Crime and Information Privacy, edited by Maria Manuela Cruz-Cunha and Nuno Ricardo Mateus-Coelho, IGI Global, 2021, pp. 466-497. https://doi.org/10.4018/978-1-7998-5728-0.ch023