Scientific and Practical Cyber Security Journal (SPCSJ) 5(3): 1-10 ISSN 2587-4667 Scientific Cyber Security Association (SCSA)

GETTING STARTED WITH ANDROID MOBILE APPLICATIONS SECURITY TESTING

P. Raghu Vamsi,**Assistant Professor, Department of Computer Science and Engineering, Jaypee Institute of Information Technology, Noida, India.**
Agrah Jain, **Solution Advisor, Delloitte USI, Gurugram, India.**

ABSTRACT: The availability of the Internet, cheaper data tariffs, and easy way of using the mobile phones made the effective use of Android mobile phones for availing Electronic Commerce (e-commerce) mobile Applications (Apps) by the people for purchasing the daily needs and regular household items. The success of the e-commerce platforms is based on their availability to public as web and Android mobile Apps. Further, their success is based on the trust and security that they maintain regarding users personal and payment data. But the poor design and development, unnoticed mistakes in coding of the e-commerce Android mobile Apps lead to many vulnerabilities and thereby becomes the simple target for the hackers. Along with conventional security testing methods, application dependent methods need to be applied on the e-commerce android Apps. To this end, this paper presents various possible practical security methods followed by penetration testers along with countermeasures that can be applicable for avoiding vulnerabilities in e-commerce Android Apps.

KEYWORDS:*Android Applications, e-commerce, penetration testing, security, testing, trust, vulnerability.*

1. INTRODUCTION

Nowadays every Electronic Commerce (e-commerce) company, from early stage startups to rising unicorns, has their Android application (Apps). There exists variety of applications like shopping Apps, dating Apps, gaming Apps, educational Apps, medical Apps etc. Only internal applications of the company cannot be found on Google Playstore due to privacy reasons other than this nearly every application can be found on Google Playstore. According to android.com there are around 50 billion android applications on Google Playstore as of December 2020 and the count is increasing day by day. Out of which around 80% of the application owner does not take any measures for the android security. Taking the example of big giants like McDonalds, its android application is so compromised by the attackers which allow anyone to buy burgers and fries for free from the Android App. According to the one of the leading Indian news agency Times of India [27] report, a company named ixigo's data was breached by the attackers and around 18 millions users' personal data was went to dark web. In this way, there are many of such examples [18, 19, 24, 26].

1.1 Contribution and Paper organization

As the number of e-commerce Apps are increasing exponentially their security concerns are also increasing rapidly. Since android penetration testing is the most underrated thing but it is as important as web application penetration testing. Also it is sometimes more important than web application since generally companies have more number of applications than a single main website. Therefore, Android App security must be the prime focus of companies and this is paper will address Android vulnerabilities in practical ways and countermeasures to protect them [7, 8, 11, 12]. An Android App will be designed generally to run on various Android operating system versions such that the presence of bugs in operating system design or patches, and the permissions used by the App may lead to new vulnerabilities and will become exploitable target. This may lead to any of the OWASP top 10 security vulnerabilities of mobile applications [1-6, 17, 18, 19]. To this end, this paper focused on practical Android App security checks to be made to get started with android security testing. This

paper articulates the Google Playstore security and privacy policies, what hackers can do with the presence of vulnerable App, the vulnerabilities hackers try to exploit, sensitive data exposure, static and dynamic analysis etc. In this way, this paper is focusing on important basic bugs that have low to medium impact to check during App security testing.

There exist two ways of analysis during security testing: 1) Static and 2) Dynamic [20-23, 25]. The static security testing is the testing of internal structure of App program and will be performed without running it. This will be done by code inspections, code walkthroughs. On the other hand, dynamic testing is the testing of program during its runtime. This paper focuses more on the static analysis and few concepts of dynamic analysis. In the Section 2, a short description the App structure and the Google security policies for Android Apps are explained. Section 3 focuses on Android Apps security testing. In this section, we present reverse engineering an App, debugging the App apk, stealing information from shared preferences folders, OAuth API call back locking (SDK testing), Hardcoded secrets, and weak logouts. Section 4 concludes the paper with future work. List of abbreviation used in this paper are illustrated in the Table 1.

API	Application programming interface
APK	Android application package
APPS	Applications
Pen-Testing	Penetration testing
POC	Proof of concept
OWASP	Open Web Application Security Project
IDOR	Insecure Direct Object Reference
Burp	Burpsuite
RCE	Remote Code Execution
ADB	Android Debug Bridge
GUI	Graphical User Interface

Table.1*. List of abbreviations*

2. ANDROID PRILIMINARIES AND GOOGLE PLAYSTORE SECURITY

Before finding vulnerabilities in Android Apps, it is important to understand basic information about its structure. This section presents the preliminaries of Android Application structure and the security scanning method of Google Play store [14, 16].

2.1 Basic structure of the Android Applications

The basic structure of the Android Apps includes the following components:

Activities: It is one of the most important things in an Android App. Since every screen on which the user interacts is an activity, when the user starts an application then the function *onCreate()* is launched and it may also start more than one activity. In other words it is the UI screen with which the user interacts. Security tester main focus should find hidden information and check for the exported status of activity.

Intents: It is a way of telling an activity to perform something or intent is a messaging object that allows communication between different application components such as Activity, Content Providers, Services, etc. It can be used for communication between application components of the same or different applications. Intents contain three items: action, data and category.

Web Views: Android allows developers to display web content directly into their application through Webviews. Security testers can consider Webview as a dedicated web browser of an application. If

the web view is not implemented properly then the attacker can open its URL in the webview of the application. To check for Web Views in application the following sequence to be followed in App: Go to manifest file search for loadUrl or Webviews details of webview to view the details.

Broadcast receiver: It helps to send and receive the system's event notification. For example if the battery is low, downloading is complete, no signal then the Android system sends broadcast messages which are received by application for their functioning like during offline the YouTube App shows no internet. The attackers mainly focuses on changing the message and makes the application behaves inappropriately.

2.2 Description of Google Play Protect

Google has developed Artificial Intelligence based systems to detect malicious applications on their platform. It can be on Google Playstore as 'Google Play Protect'. Every application that is downloaded is scanned by the Play Protect, if it is found to be malicious then Google will not allow it to be installed. Also the Play protect scans the installed applications for security threats. Fig. 1 shows the Google Play Protect after scanning Apps for security threats.

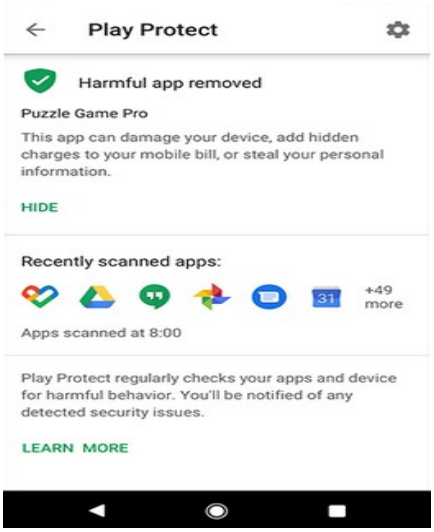

Fig.1. *Result after Play Protect scanned the Apps*

Play Protect regularly scans Apps for any sign of malware, it also monitors installed applications for any unintended behavior and Google play protect also helps to track the mobile phone when lost and wipe data if needed. Apart from playstore Apps there are non playstore apps i.e., internal Apps of any organization which is not published on playstore. To scan these types of Apps, users should scan it from virustotal.com for any traces of malware or loggers present. The virus total scans the item (Android apk) with 70+ antivirus scanners and also looks for presence of any blacklist URL. Security tester need to do is upload the apk to the virus total before installing and see the score and results of every scanner present as shown in the Fig. 2.

Now the question arises is why the Google play protect or website like virus total is not sufficient for scanning Apps. The answer is that it is because they work on phone security level i.e., scanning the apk for virus, malware, loggers, privacy etc. But to find vulnerabilities like authentication bypass of any application, sensitive data leakage, parameter tampering while payment from any application, code tempering, insecure data storage, insufficient cryptography, reverse engineering etc, and to get application level security proper vulnerability assessment and penetration testing is needed. Further, these tests cannot be performed by either by Google play protect and any other scanning website like virus total.

3. ANDROID APPLICATION SECURITY TESTING

In this section, the basic and medium level vulnerabilities that appear in the Android Apps were explained. For this, the tools we will be using are frida, drozer, jadx-gui, android debugging bridge, mob-sf and the Burpsuite. From which frida and mob-sf is used for automated vulnerability scanning, drozer for interaction analysis, adb is used for debugging, jadx-gui is used for reversing the android app and Burpsuite is used for dynamic analysis [9, 10, 13, 15].

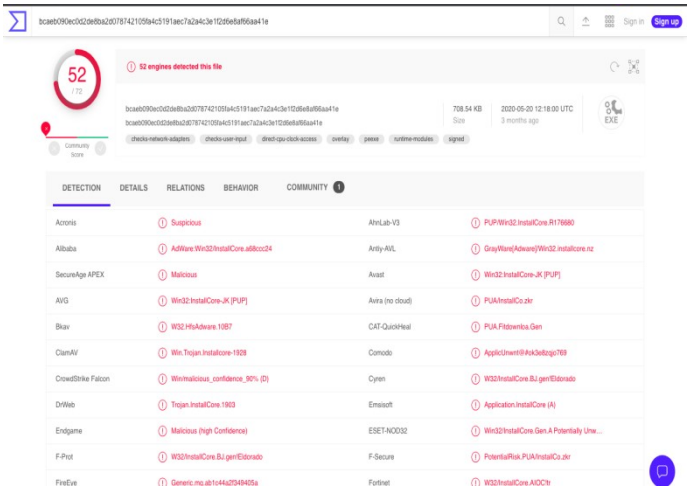

***Fig.2**. Result after Virus total scan*

3.1 Reverse engineering and Debugging Apps

Reverse engineering is one of the important steps in any testing also known as back engineering with the help of it attacker can do source code analysis of android application, manipulate the raw data, reconstruct the application, extract the knowledge from the data, and present this repackaged apk to end user to initiate attacks such as personal information stealing, server side information gathering etc. Now the most important part is how to decompile an android application. To do it, we use the tool named jadx-gui which is graphical user interface tool to decompile android application. Run the jadx tools and load the apk such that it will do rest of the things for analysis of the application. Fig. 3 shows the manifest.xml file generated by jadx-gui after decompiling apk file.

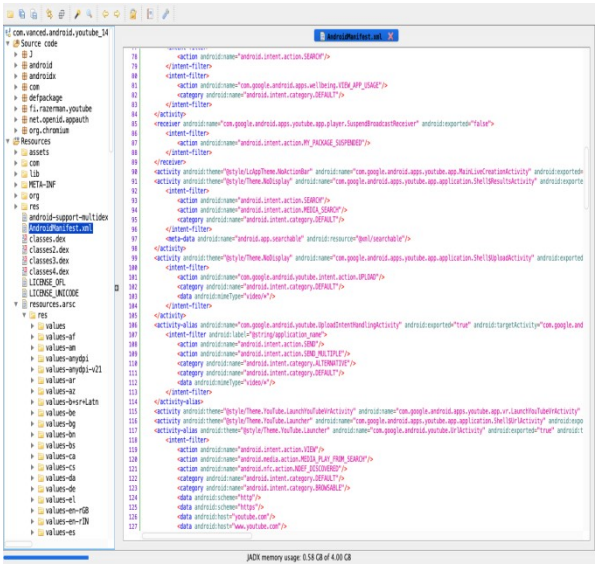

***Fig.3**. Manifest.xml file in Jadx-gui after decompiling*

After the source code analysis, the debugging is done in android studio to analyze the working of functions, intents, actions, data storage details etc. For this first decompile an application using jadx-gui and save it then import this file as grade in android studio and can debug at any specific section of the code. Also the testers need to understand the working of application using debug option in Android studio. As a precautionary step, it is recommended to turn on the anti debugging flag in the code so that the attacker cannot debug it. This anti debugging does not allow injecting garbage data in the communication channel of manually set breakpoints to confuse the debugger.

3.2 Stealing sensitive information from shared preferences

In general, Apps write sensitive information in shared_prefs files which can be anything from access token to some credentials. To this vulnerability, tester need to check MIME type in android manifest file, MIME type must be application/pdf, application/*. Navigate to the activity in which MIME type is defined in the jadx-gui and look for functions named onCreate(), onResume(), onNewIntent(). Remember that whenever tester will trigger these types of exploits using apk it always comes via the intent function. Once it is done, analysis to be conducted in the above mentioned functions which is sending intent with some data as stream and to which destination path and file name is set. For this two types of exploitations are possible: 1) using a rogue app and 2) using adb (Android debugger) to access the shared_Prefs folder. The second method is used for explanation. To do this, try to find the access token in the shared_prefs using ADB for MIME type. Setup adb in the system terminal and use the below commands to access the shared_prefs to see the data as shown Fig. 4.

adb devices

adb shell

cd data/data/com.package_name

cd shared_prefs

cat prefs.xml

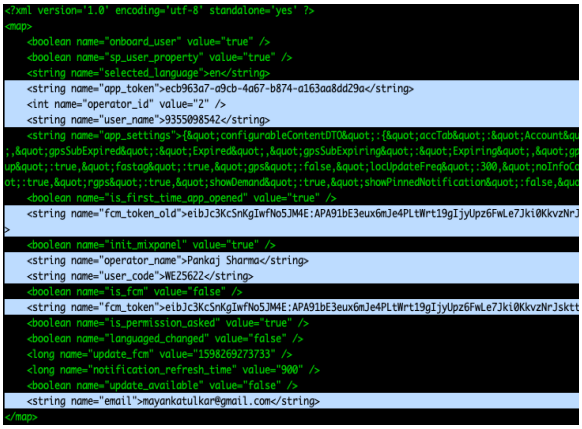

Fig.4*. Secret data and access tokens in shared prefs*

From Fig.4 it can be seen that so much sensitive information can be found in shared prefs which includes fcm token, access token, user code, email, phone number etc. To get protected from this never ever store sensitive information in the shared prefs folder because it is the main target of attackers. Further, it is recommended to place the data in an encrypted format to avoid data theft.

3.3 OAuth API call back Locking (or SDK Testing)

Oauth is token based authorization used commonly by every application to authenticate the user to access their account without entering password using Google, Facebook, Pinterest, Yandex etc. For example, if the user is using an e-commerce App X, and there is a provider which connects with your application say Paypal for payments. Paypal provides a dashboard where user can define redirect uri in android application there is a “scheme” to redirect uri is the url for the web domain similarly scheme is the url for the mobile application. And if user does not define it then the App X become vulnerable and it can be used to hijack the sessions in the App X. To exploit this vulnerability, an attacker needs an rogue apk installed on user device so that it can capture the token and hijack user session. We have made an apk token catcher and defined the scheme name praskhar to capture it as shown in Fig. 5. The countermeasure to get protected from this vulnerability is to define ‘scheme’ in intent filter and filter out the url that is passing any unwanted content.

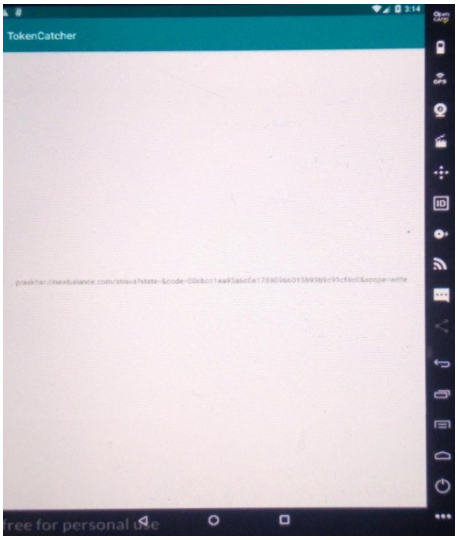

Fig.5*. Rogue apk capturing token with scheme praskhar*

3.4 Hardcoded Scerates

This section focuses on sensitive data exposure, in which we will look for the hardcoded tokens, api keys, keys, Oauth tokens, jwt tokens, username and passwords etc. There are two types of exploitations one is SDK based, and other is API as a service (mostly Google maps). We focus on API as service and find some secrets and try to exploit them practically. Starting with the point where to find these hard coded secrets, for this need Jadx-gui tool used to decompile application. After decompiling the following three paths for any hardcoded keys with proper filters to be checked as shown in Fig. 6.

1. Resources.asrc values strings.xml
2. Resources.asrc values Array.xml
3. Code analysis (also androidManifest)

After obtaining the hardcoded api key (AIzaSyDHfC0q0Ahujq9Pduvjs-757ffUtd), tester need to check whether it is exploitable or not. For Google maps api key, a command line and Python based open source tool called gmapapiscanner is used. To launch this tool use the following command

$ *python3 maps_api_scanner_python3.py AIzaSyDHfC0q0Ahujq9Pduvjs-757ffUtd*

Output of the above command is shown in the Fig. 7. From Fig. 7 it can be observed that the API key found is vulnerable to two paid services. To get protected from this, user need to do following things: 1) try not to hardcode any secret keys; 2) if it is necessary to hardcode then always put the url white listing and referrer check to ensure that only these api keys must be called from App but not by any other anonymous user.

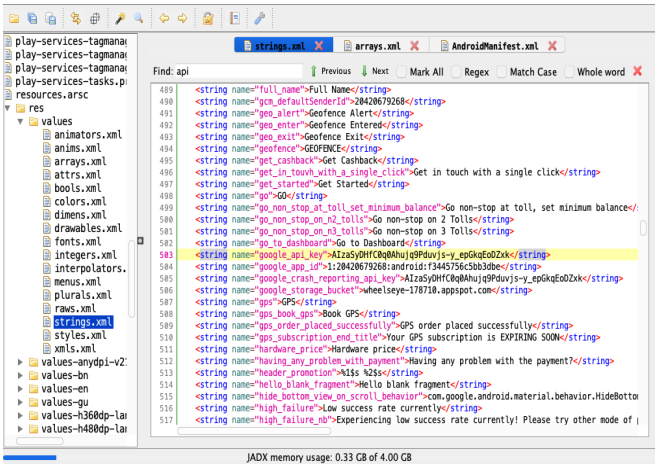

***Fig.6**. Harcoded google api key in strings.xml*

***Fig.7**. Google api key is vulnerable to two services*

3.5 Weak Logouts

This vulnerability is known as authentication bypass which arises due to not implementing proper logout functionality of OAuth clients like login with Google, LinkedIn etc. With this the attacker can login to the victim's account through any of the above client api without entering the login credentials. This vulnerability arises when an application logs out but the application token does not expire or logout the access or session token taking advantage of it. Then the attacker gains unauthorized access to your account.

We present this vulnerability check using Brupsuite. To check this vulnerability, trigger the application login using Google, Facebook or any other client. Turn the intercept on in the Burpsuite and logout the application. Now check for the response at the Burpsuite, if tester observes that there is only app token which is logging out and no session id present then this vulnerability may exist in the application. To have cross verification, logout the application after logging in from third party client then login again. Click on login with Facebook don't put any email and password in the fields as shown in Fig. 8 and press the tiny close button and check for will be logged in automatically in the application or not. It can be observed from Fig.8 that login with Facebook API call is giving success response in the Burpsuite. It is because user was not properly logged out from the application as session id or token is not expired.

As a countermeasure to this vulnerability one must need to properly implement logout functionality both access token and application token must expire after logout. And each new session will be assigned a new session ID which should expire after logging out an application.

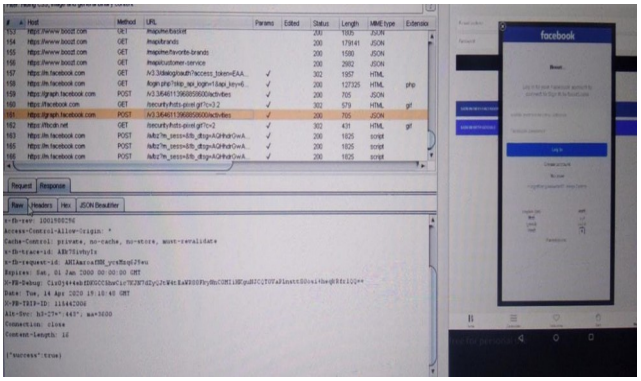

Fig.7*. Finding weak logout functionality*

4. CONCLUSION AND FURTURE WORK

This paper presented the method of starting with the Android mobile Apps security testing. It covered the topics such as the description of Google play protect, how to perform malware analysis of non play store apps using virus total, static analysis of Apps using reverse engineering, API key exploitation, digging hard coded secrets, weak logout functionality, and sensitive information disclosure. This paper also discussed how to find these vulnerabilities, what attacker can do with these vulnerabilities and countermeasures. In this paper, we have limited our research to basic important vulnerabilities present in android applications which have a severity level of low to medium.

As part of the future work, we attempt to perform dynamic analysis (both manual and automated) with the help of open source tools with main focus on advanced critical bugs related to android applications such as Broadcast receiver exploitation, broadcast sniffing, IDOR in android applications, remote code execution, content provider exploitation using drozer and deep link exploitation and others.

REFERENCES

1. Mahmood, Riyadh, Naeem Esfahani, Thabet Kacem, Nariman Mirzaei, Sam Malek, and Angelos Stavrou. "A whitebox approach for automated security testing of Android applications on the cloud." In 2012 7th International Workshop on Automation of Software Test (AST), pp. 22-28. IEEE, 2012.
2. Rai, Pragati Ogal. Android Application Security Essentials. Packt Publishing Ltd, 2013.

3. Avancini, Andrea, and Mariano Ceccato. "Security testing of the communication among Android applications." In 2013 8th International Workshop on Automation of Software Test (AST), pp. 57-63. IEEE, 2013.
4. Salva, Sébastien, and Stassia R. Zafimiharisoa. "APSET, an Android aPplication SEcurity Testing tool for detecting intent-based vulnerabilities." International Journal on Software Tools for Technology Transfer 17, no. 2 (2015): 201-221.
5. Mente, Rajivkumar, and Asha Bagadi. "Android application security." Advances in Computational Sciences and Technology 10, no. 5 (2017): 1207-1210.
6. Fischer, Felix, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. "Stack overflow considered harmful? the impact of copy&paste on android application security." In 2017 IEEE Symposium on Security and Privacy (SP), pp. 121-136. IEEE, 2017.
7. Acar, Yasemin, Christian Stransky, Dominik Wermke, Charles Weir, Michelle L. Mazurek, and Sascha Fahl. "Developers need support, too: A survey of security advice for software developers." In 2017 IEEE Cybersecurity Development (SecDev), pp. 22-26. IEEE, 2017.
8. Dashevskyi, Stanislav, Olga Gadyatskaya, Aleksandr Pilgun, and Yury Zhauniarovich. "The influence of code coverage metrics on automated testing efficiency in android." In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 2216-2218. 2018.
9. Sinaga, Arnaldo Marulitua, P. Adi Wibowo, Ariestoni Silalahi, and Nita Yolanda. "Performance of automation testing tools for android applications." In 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 534-539. IEEE, 2018.
10. Kulkarni, Keyur, and Ahmad Y. Javaid. "Open source android vulnerability detection tools: a survey." arXiv preprint arXiv:1807.11840 (2018).
11. Montealegre, C., Njuguna, C.R., Malik, M.I., Hannay, P., & McAteer, I.N. (2018). "Security vulnerabilities in android applications", In proceedings of the 16th Australian Information Security Management Conference (pp. 14-28). Perth, Australia: Edith Cowan University.
12. Pan, Yuanyuan. "Interactive application security testing." In 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), pp. 558-561. IEEE, 2019.
13. Morgado, Inês Coimbra, and Ana CR Paiva. "The iMPAcT tool for Android testing." Proceedings of the ACM on Human-Computer Interaction 3, no. EICS (2019): 1-23.
14. Alkindi, Zainab R., Sultan Qaboos Unviresity, Oman Muscat, Mohamed Sarrab, and Nasser Alzidi. "Android Application Permission Model." In 4th FREE & OPEN SOURCE SOFTWARE CONFERENCE (FOSSC'2019-OMAN). 2019.
15. Almeida, Diego R., Patrícia DL Machado, and Wilkerson L. Andrade. "Testing tools for Android context-aware applications: a systematic mapping." Journal of the Brazilian Computer Society 25, no. 1 (2019): 1-22.
16. Lai, Duling, and Julia Rubin. "Goal-driven exploration for android applications." In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 115-127. IEEE, 2019.
17. He, Yongzhong, Xuejun Yang, Binghui Hu, and Wei Wang. "Dynamic privacy leakage analysis of Android third-party libraries." Journal of Information Security and Applications 46 (2019): 259-270.
18. Alanda, Aide, Deni Satria, H. A. Mooduto, and Bobby Kurniawan. "Mobile Application Security Penetration Testing Based on OWASP." In IOP Conference Series: Materials Science and Engineering, vol. 846, no. 1, p. 012036. IOP Publishing, 2020.
19. Li, Jinfeng. "Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST)." Annals of Emerging Technologies in Computing (AETiC), Print ISSN (2020): 2516-0281.
20. Xiao, Jianmao, Shizhan Chen, Qiang He, Zhiyong Feng, and Xiao Xue. "An Android application risk evaluation framework based on minimum permission set identification." Journal of Systems and Software 163 (2020): 110533.

21. Savola, Reijo M., Markku Kylänpää, and Habtamu Abie. "Risk-driven security metrics for an Android smartphone application." International Journal of Electronic Business 15, no. 4 (2020): 297-324.
22. Yasin, Husam N., Siti Hafizah Ab Hamid, Raja Jamilah Raja Yusof, and Muzaffar Hamzah. "An empirical analysis of test input generation tools for android apps through a sequence of events." Symmetry 12, no. 11 (2020): 1894.
23. Pecorelli, Fabiano, Gemma Catolino, Filomena Ferrucci, Andrea De Lucia, and Fabio Palomba. "Testing of mobile applications in the wild: A large-scale empirical study on android apps." In Proceedings of the 28th International Conference on Program Comprehension, pp. 296-307. 2020.
24. Rani, Sangeeta, and Kanwalvir Singh Dhindsa. "Android application security: detecting Android malware and evaluating anti-malware software." International Journal of Internet Technology and Secured Transactions 10, no. 4 (2020): 491-506.
25. Dawoud, Abdallah, and Sven Bugiel. "Bringing balance to the force: Dynamic analysis of the android application framework." Bringing Balance to the Force: Dynamic Analysis of the Android Application Framework (2021).
26. Κούκουνας, Άγγελος Παναγιώτης. "Malware analysis, security evaluation for Android application." Master's thesis, Πανεπιστήμιο Πειραιώς, 2021.
27. News Article: https://timesofindia.indiatimes.com/business/india-business/emails-hashed-passwords-of-18m-ixigo-users-stolen/articleshow/68016866.cms (Last accessed 15-08-2021)