Advanced Science and Technology Letters
Vol.120 (GST 2015), pp.293-298
http://dx.doi.org/10.14257/astl.2015.120.59

A Single-Process Design for Developing Automation Tools for Inspecting the Vulnerabilities of Android Applications

Da-Woon Leem[1], Hyun-Ju Jung [2], Moon-Sung Hwang[3], Jung-Ah Shim[4], and Hyun-Jung Kwon[5*]

[1] Graduate School of Software, Soongsil University, Seoul 156-743, Republic of Korea
[2,3,4,5*] Department of IT Policy and Management, Soongsil University, Seoul 156-743, Republic of Korea
[1]dy626@naver.com, [2]hyunju@hashtag.co.kr, [3]ms.hwang@sabic-ip.com, [4]rosapoodle@naver.com, [5*]hyun_jung00@nate.com

Abstract. As a variety of services are provided through mobile devices, there have been increasing security threats to mobile platforms and applications. Many mobile application security check methodologies are proposed to deal with such threats, but it is not clear exactly how the security vulnerabilities are inspected, and there is no correct methodology for such. One way to increase the execution rate of vulnerability check is to develop an automation tool that simplifies the process. This study designed and proposed a single process of inspecting vulnerabilities as a precedent study on the development of automation tools for checking the vulnerabilities of Android applications. After examining the existing methods of checking vulnerabilities, any redundant process was removed and incorporated into a single process, in which diverse items are inspected.

Keywords: Android, mobile application, vulnerability, inspection process

1 Introduction

Today, mobile services dealing with critical financial information such as mobile banking, Fetch, and Easy Payment System have encountered concentrated attacks by hackers [1]. To deal with such application security threats, domestic public institutions and foreign security standard organizations have presented mobile security inspection methodologies, but most of them do not include detailed inspection stages or criteria [2, 3]. Thus, the application vulnerability inspector has to perform separate detailed inspections to check each item. Moreover, the inspection methods that are checked as such are presented individually, and the inspector must therefore go through a similar process for every inspection. Such complexity decreases the application vulnerability inspection rate and consequently lowers the reliability of the mobile application security. Considering this, this study was conducted with the aim of contributing to the development of an automation tool for inspecting Android application vulnerabilities, and of analyzing the existing mobile security inspection methodologies and proposing a single inspection process.

ISSN: 2287-1233 ASTL

2 Related Research

2.1 Existing research on vulnerabilities of mobile applications

The main objective of this study is to present an efficient method for inspecting the vulnerabilities of Android applications, and the following studies have been conducted regarding this subject.

[4] designed and implemented a vulnerability analyzer for applications of the mobile platform ′bada′. Based on the analysis results of CWE (Common Weakness Enumeration) and CERT (Computer Emergency Response Team), an analyzer that derives and analyzes inspection items related to an event-driven method was implemented. This study has significance in that it improved the efficiency of inspection of vulnerabilities by presenting a tool for application developers to efficiently derive and analyze inspection items. The present study also adopted this method. However, [4] has limitations because the platform ′bada′ researched in [4] is a mobile platform for which development and support have been stopped and the scope of inspection is limited to an event-driven method.

[5] presents the Android App Security Assessment Framework App-Ray which performs inspections according to the inspection items selected by user, and designed a framework for integrating conventional static and dynamic analysis methods. This agrees with the direction that the present study has taken to present an efficient method by integrating several fragmented inspection methods. However, [5] presents only a prototype for analyzing four inspection items related to user privacy in design verification and does not deal with actual integration of different analysis methods.

2.2 Mobile Security Inspection Methodologies

Mobile Civil Service Security Vulnerability Inspection Guide

In October 2014, the Ministry of Security and Public Administration and Korea Internet and Security Agency published the Mobile Civil Service Vulnerability Inspection Guide for developers of mobile e-government civil services [6]. This guide presents useful inspection processes and methods for the mobile services developed by South Korean administrative agencies. The inspection method is roughly divided into security vulnerability inspection and source code vulnerability inspection. As this study dealt with the method in the form of a black-box test where the apk files were used as the analysis data, it did not consider the source code method because it used the source code itself before the compilation as the analysis data.

Open Web Application Security Project (OWASP) Mobile Top

The Open Web Application Security Project (OWASP) is an international web security standard organization that runs diverse projects about the general aspects of

application security, such as information leakage, malicious files, and security vulnerabilities. As mobile security is considered more important than ever at present, OWASP came up with and announced its completion of Mobile Top 10 [7], which is currently being used as the main standard of mobile vulnerability inspection by domestic and international companies and agencies.

3 Main Subject

3.1 Android Vulnerability Inspection Items

Table 1 lists the selected detailed items that can be made to undergo the inspection process based on the existing vulnerability inspection methodologies and detailed inspection items. The execution timing of the inspection was categorized based on six time points: before installation, after installation, during use, after use, after decompilation, and after uninstallation.

Table 1. Detailed Vulnerability Inspection Method for Android Applications & Classifications according to the Existing Mobile Methodologies

Android Application Vulnerability In-Depth Inspection			OWAS Mobile Top 10	Functional Security Vulnerability Inspection
No.	**Description**	**Execution Timing**	**Code**	**No.**
1	Compare the internal lists of the system files before and after installing the app.	Before & after installation	-	2
2	Check if authority not related to the service has been set to user permission in AndroidManifest.xml.	After decompilation	M4	3
3	Compare the internal lists of the system files before installing the app and after uninstalling it to check if they are the same.	Before installation & after uninstallation	-	4
4	By following the netstat instructions, check if any backdoor port is created.	During use	M4	6
5	By following the ls instructions, check if any unnecessary file is created.	During use	M4	
6	By following the ps instructions, check if any unnecessary process is created.	During use	M4	
7	Check if the authority to install the app is identical to its execution.	During use	-	10
8	Check if Shared UserID has been set up in AndroidManifest.xml.	After decompilation	-	11
9	Check the intent-filter items of each component in AndroidManifest.xml to identify any excessive authority.	After decompilation	-	12
10	After logging on, check if the user's	After use	M2	14

	authentication information is saved through the grep instructions.			
11	In the packet captured during the login, check if the user's authentication information is delivered in the form of plain text.	During use	M3	
12	In the packet captured during the login, check if the user's authentication information is using any vulnerable coding algorithm.	During use	M6	15
13	Decompile apk and identify any obfuscation.	After decompilation	M10	20
14	Check if the user's authentication and personal information are saved in the SQLite & shared_pref folders.	After use	M2	-
15	Inspect the suitability of the protect level in AndroidManifest.xml.	After decompilation	M4	-
16	Check if any important information has leaked out of the app log file.	After use	M4	-
17	Search for components with exported setup, and inspect the approach protection in AndroidManifest.xml.	After decompilation	M4	-
18	Check if "debuggable" has been set up in AndroidManifest.xml.	After decompilation	M4	-
19	Validate the suitability of the content provider's permission in AndroidManifest.xml.	After decompilation	M4	-
20	Validate the application signature.	After decompilation	M5	-

3.2 Vulnerability Inspection Process

Based on the detailed methods and the execution timing shown in Table 1, a vulnerability inspection process was designed to perform vulnerability inspection at the general time period during which the application is — installed, used, and uninstalled (Detailed descriptions of each process are outlined in Table 2.).

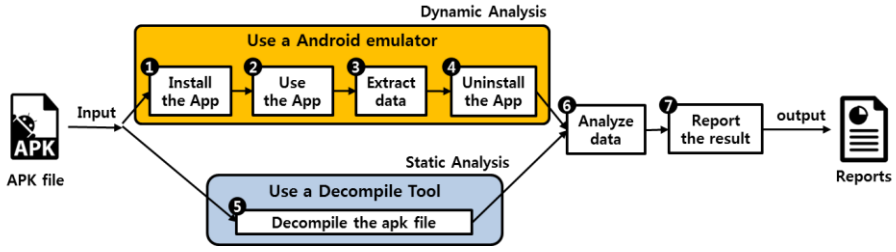

Fig. 1. Vulnerability inspection process

Table 2. Detailed steps and descriptions of the vulnerability inspection process

Detailed steps of the process	Description
Install the App	• Save the system files before installation. • Record the system network conditions before installing the App. • Install the App. • Save the system files after installation. • Save the install file authority data
Use the App	• Record the system network conditions while using the App. • Save the app execution authority data. • Capture the packet
Extract data	• Extract log data • Extract files at data/data/<package_name>
Uninstall the App	• Uninstall the App • Save the system files after uninstallation
Decompile the apk file	• Decompile the apk file
Analyze data	• Analyze the data collected from previous steps based on the inspection items in Table 1.
Report the result	• Report the result

4 Conclusion

Research on the Android security inspections has been conducted in detailed and various ways [8]. However, research on the integration of these inspection methods to improve the efficiency of inspection is insufficient. In particular, inspections related to data exposure which account for a greater part of application vulnerabilities tend to still depend on manual inspection [9]. On this background, the integrated single process presented in this paper has significance as a simplified process that improves the execution rate of inspections and is valuable study for automation of inspections related to data exposure. As a future research subject, an automation tool for inspecting Android security vulnerabilities will be developed based on this process and its effectiveness will be verified.

References

1. THE INVISIBLE BECOMES VISIBLE, https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-the-invisible-becomes-visible.pdf
2. Gimgyeonggon: Research for Android Apps security diagnostics: Korea University Graduate School of Information Security (2015)
3. Hur Hwan Seok, Kang Sung Hoon, Kim Seung Joo: A Proposal for Security Verification Method for Implementation of Secure Android Mobile Application: Korea Information Processing Society (2013)
4. Ilyong Mun, Seman Oh: Design and Implementation of A Weakness Analyzer for Mobile Applications. Journal of Korea Multimedia Society, pp. 1335-1347(2011)

5. Titze, D., Stephanow, P., and Schuette, J.: ′App-ray: User-driven and fully automated android app security assessment′, Fraunhofer AISEC TechReport, (2013)
6. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov
7. OWASP Mobile Security Project, https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks
8. Rashidi, B., Fung, C.: A Survey of Android Security Threats and Defenses′, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) pp. 3—35(2015)
9. Young-Jae Jeon, Han-Seung Woo, Sang-Joon Yoo, Hwa-Jae Choi: Analysis of Mobile Application Vulnerabilities, Institute for Information & Communications Technology Promotion (2015)

