2015 IEEE 39th Annual International Computers, Software & Applications Conference

Security Design for Configuration Management of Android Devices

Cheng-Liang Kuo
National Kaohsiung Normal University
Kaohsiung, Taiwan
f74721348@gmail.com

Chung-Huang Yang
National Kaohsiung Normal University
Kaohsiung, Taiwan
chyang@nknu.edu.tw

***Abstract*—In recent years, due to increasing popularity of mobile devices and the mobile Internet service, as well as device functions becoming more diverse and powerful, ensuring device security is gradually being taken seriously. Vulnerabilities of mobile devices are mostly caused by improper configuration settings; therefore, remediation of these settings can reduce vulnerabilities and thus improve system security. This research created a configuration management system for Android mobile devices that complies with Google Compatibility Definition Document (CDD) based on the Security Content Automation Protocol (SCAP) proposed by the National Institute of Standards and Technology (NIST) and conforms to the CIS Google Android OS Benchmark test items. The objective of this research is to reduce the vulnerabilities generated in the system due to incorrect user configuration settings and to uphold information security in Android mobile devices.**

Keywords-system configuration; Security Content Automation Protocol; Android; mobile security

I. Introduction

Since Apple and Google launched smart mobile devices, the development and functions has become more powerful. The smart mobile device now plays an important role in people's lives. At present, the period when mobile devices are popular and mobile Internet prevalent, personal information are mostly stored in a person's device. If protection is not done well, private information could be leaked or stolen. The 2014 report published by the well-known Internet security technology company Symantec [1] mentions that cases of personal information leakage have gradually increased throughout the years. In 2013 alone, there was a total of 552 million incidents of personal data leakage. Therefore, mobile device information security protection is a very significant issue.

Among smart mobile devices, Google's Android operating system has the highest share of the market. The international survey research and advisory firm Gartner [2] expressed that in 2013, the global market share of the Android ranked first, reaching 78.4%, higher than 15.6% for iOS. Android is created on Linux-based open source operating system for mobile devices. The Linux operating system often encounters system vulnerabilities caused by improper settings made by users during operation, causing the system to be unsafe. This is also similar in Android. As an example, the Gartner IT Infrastructure & Operations Management Summit 2014 mentioned that 75% of mobile device security vulnerabilities are generated by inappropriate application of configuration settings by users [3]; therefore, if erroneous user configuration settings can be detected and repaired, then the system vulnerabilities that they produced can be eliminated, reducing the risk of private information theft [4], thus improving device security.

Taking the above mentioned reasons in consideration, this research utilizes Eclipse, Java SE Development Kit, Google Android APIs as developmental tools for Android mobile devices. It used the third-party software jOVAL [5] as online scan engine and created a configuration repair system for Android mobile devices that complies with Google's Compatibility Definition Document (CDD) [6] based on SCAP [7] proposed by the National Institute of Standards and Technology (NIST) and conforms to the CIS Google Android OS Benchmark [8] test items. The purpose of this research is to eliminate the vulnerabilities generated in the system due to incorrect user configuration and to uphold information security in Android mobile devices.

II. Security Application Design For Android Devices

The research is according to SCAP which is proposed by NIST and to the Google Android OS benchmark document on system configuration guidelines released by the CIS, aims to develop, implement systems configuration detection and repair on Android devices. Related literatures were investigated for relevant terms and definitions used in this research.

A. *Android Mobile Devices*

Android mobile devices refer to smart mobile devices powered by the Android operating system, such as Android smartphones and tablets. The Android operating system is jointly developed and modified by Google and the 84 manufacturer members of the Open Handset Alliance (OHA). It is a set of open source operating system for mobile devices with Linux as the kernel; the system architecture [9] is shown in Figure 1. Various manufacturers and traders can freely utilize and customize it according to need. It currently has a higher market share among the available systems for mobile devices.

In the Android system architecture, the Linux kernel at the bottom layer is mainly used to manage operating system memory, program execution, system security, as well as drivers to ensure successful implementation of applications. The next layer, libraries, is composed of C and C ++

languages; the application program can only use the functions in this layer via API. Android Runtime includes virtual machine and kernel libraries. The application framework on the above layer provides an API to assist program design and development, while the most commonly accessed and widely familiar applications is located on the uppermost layer.

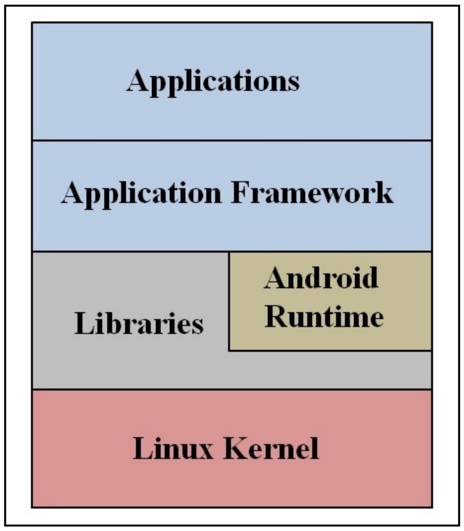


Figure 1. Android system architecture.

B. CDD

The Compatibility Definition Document (CDD) [6] was proposed by Google. Recorded in this document are the functions and specifications that must be met by up to standard Android devices; they are: Application Packaging Compatibility, Multimedia Compatibility, Developer Tools and Options Compatibility, Hardware Compatibility, Performance Compatibility, Security Model Compatibility, Software & Software Compatibility Testing, Android Compatibility Test Suite, Compatibility Test Suite Verifier, Reference Applications, and Updatable Software. Only after passing these requirements can a device conform to CDD specifications. If the Android device customized by manufacturers does not meet the CDD standard, Google will not offer its authorized services to these firms. Among the more popular Android mobile devices that do not meet the CDD's standard are Xiaomi and other imitation devices made in China. Therefore, Google Play Store and other services are not available in these devices.

C. NIST

The National Institute of Standards and Technology (NIST), formerly known as the National Bureau of Standards (NBS) employs more than 5,000 scientists, engineers, science and technology operators, experts, logistics personnel, management staff, and support staff. It also has engineering, information technology, materials measurement, physical measurement, and nanotechnology laboratories, as well as the NIST Center for Neutron Research. The NIST encompasses very broad subject areas including biological sciences, building research, electronic communications, information technology, material science, nanotechnology, neutron research, public safety, etc. Its purpose is to increase industrial productivity and improve the quality of life.

D. SCAP

The purpose of the Security Content Automation Protocol (SCAP) [7] proposed by the NIST is to combine open standards in order to automate and standardize management weaknesses. CVE, CVSS, XCCDF, and OVAL standards are used to itemize and find system or software and hardware weaknesses and vulnerabilities [10]. They also provide a measurement method for assessing possible impacts on the system. At present, this is a relatively mature information security evaluation criteria in the United States.

E. XCCDF

The eXtensible Configuration Checklist Description Format (XCCDF) is a descriptive language and is one of the open standards for SCAP. This is mainly used to compose safety checklists and benchmark files and various documents related to information security [14]. XCCDF collects configuration rules for security settings in a structured way and aims to support information exchange, file generation, and automated testing and scoring in order to promote wider application of document files in security systems.

F. CIS Google Android OS Benchmark

The Google Android OS Benchmark [8] was proposed by the Center for Internet Security (CIS) and is a standard for Android security configuration. This document specifies under what state the Android operating system configuration settings should be in order for it to be more secure. System administrators or users can set an Android operating system configuration based on this document in order to heighten the security of Android mobile devices.

G. jOVAL

jOVAL [5] is an open source software (OSS) that can perform XCCDF and OVAL scans on Linux, Windows, Mac OS, and other operating systems and produce score reports. This research uses this software as the base to develop an online scan engine to scan XCCDF configuration file uploaded by the user.

III. System Architecture And Development

A. System Architecture

This system is composed of two parts as shown in Figure 2: the APP (application) side mounted on mobile devices and the server-side on computers. The APP side mainly detects and repairs device configuration. The user can see the test results for each configuration item in this portion and click the automatic or manual button provided by the system to repair the configuration. The server-side provides online scan and generates reports. After users click the “Create a Scanned File” button on the APP side, the system will generate a configuration file according to the configuration of the mobile device. Then after connecting and uploading to the server, configuration scan will be carried out. Scans are done by the 3rd party open source software jOVAL. Scan results are often similar to the configuration result detected by the APP side; the difference is that the report generated by online scan gives the configuration test information and system score in detail. These can be made available to management as a reference during security audit. Test results for APP side only briefly describe the items and can be used to repair the system configuration. Since the detection format, items, and pass or fail criteria of this system are all based on the proposed XCCDF in NIST's SCAP and Google Android OS Benchmark released by the CIS, therefore, this system is in line with international safety standards and can actually enhance the security of mobile devices.

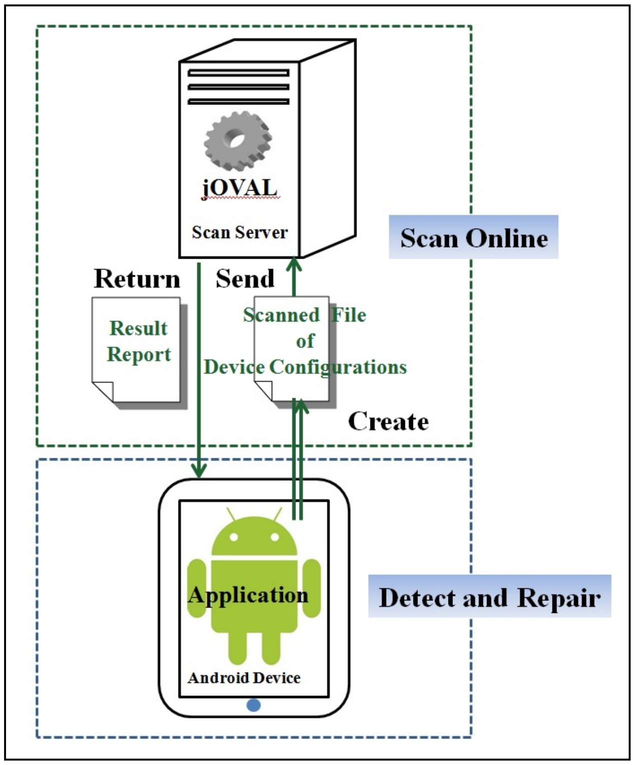


Figure 2. System architecture diagram.

B. *development tools and environment*

The system development and test environment are shown in Table 1, and the detect items are shown in Table 2. This system uses Ubuntu operating system as server for the server-side and uses JSP (JavaServer Pages) language, JavaScript, and third party open source software jOVAL for the development of an online scan server that follows SCAP's XCCDF standards. The APP side uses Java language and Google Developers API on Eclipse and adheres to the configuration of the CIS Google Android 4 Benchmark file, as shown in Figure 2. Because Android is an open source operating system for mobile devices, many manufacturers customize it before use. But if during customization, the terms of Google's CDD are not met, then Google Inc. will not provide authorization and its services will not be available for use. This may cause the system to produce incorrect results when executed. Therefore, this research states this as its limitations. Only Android mobile devices that meet CDD specification can use this system in a normal way to detect and repair system configuration.

TABLE I. SYSTEM AND TEST ENVIRONMENT

Item	Tools
Operation System of Server	Ubuntu Linux
Development Tools	Java SE Development Kit(JDK)8, Eclipse LUNA, Google Android APIs, jOVAL
Program language	Java, JavaServer Pages(JSP), JavaScript
Test Environment	ASUS Nexus 7 (Android2.3.3), ASUS Nexus 7 II (Android4.4.2), Samsung Nexus S (Android4.2.2), Sony Xperia ZU (Android4.2.2)
Execution Environment	Android devices that follow the CDD rules

TABLE II. DETECT ITEM AND CONFIGURATION TYPE

Detect Item	Configuration Type
Android Version	System Configuration
Set Auto-Lock Time	System Configuration
Disable Installation of Third-Party Apps	System Configuration
Set Screen Lock	System Configuration
Set Maximum Number of Failed Attempts	System Configuration
Set Minimum Password Length	System Configuration
Disable Simple Value for password	System Configuration
Set Grace Period for Screen Lock	System Configuration
Disable WiFi	Device Configuration
Disable Camera	Device Configuration
Disable Screen Capture	Device Configuration
Disable Sending Diagnostic Data to Google	Browser Configuration
Disable Browser Autofill	Browser Configuration
Enable Browser Pop-up Blocking	Browser Configuration
Enable Browser Fraud Warning	Browser Configuration
Browser cookie settings	Browser Configuration

IV. System Implementation

After installing APP on an Android device, the user can begin to use the system. Please refer to Figure 3 for the processes. When APP is executed, the screen in Figure 4 will appear. At this time, the user can choose to enter the detect and repair mode or the scan online mode according to his preference. The detect and repair mode provides an outline of the configuration being checked as well as the configuration repair function. These allow the user to know configuration test results and repair settings. Meanwhile, the online scan mode provides a detailed report and configuration information for user reference after completing the scan.

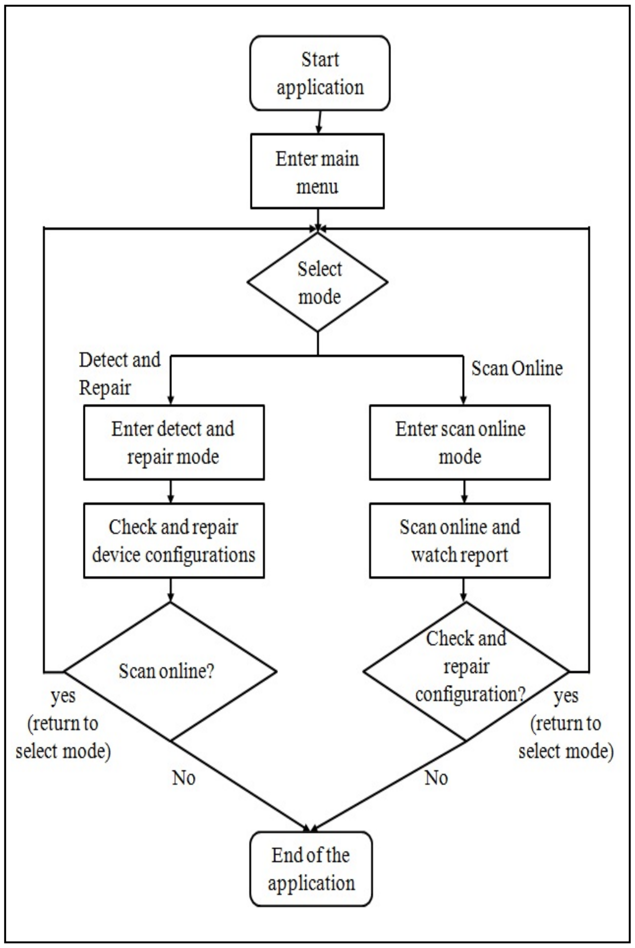

Start application
Enter main menu
Select mode
Detect and Repair
Scan Online
Enter detect and repair mode
Enter scan online mode
Check and repair device configurations
Scan online and watch report
Scan online?
Check and repair configuration?
yes (return to select mode)
yes (return to select mode)
No
No
End of the application

Figure 3. System implementation flow chart.

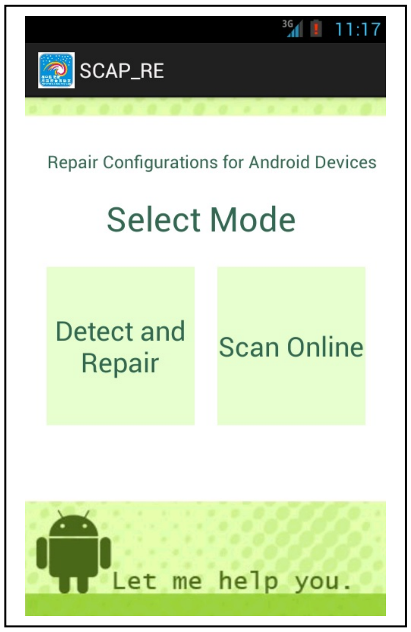
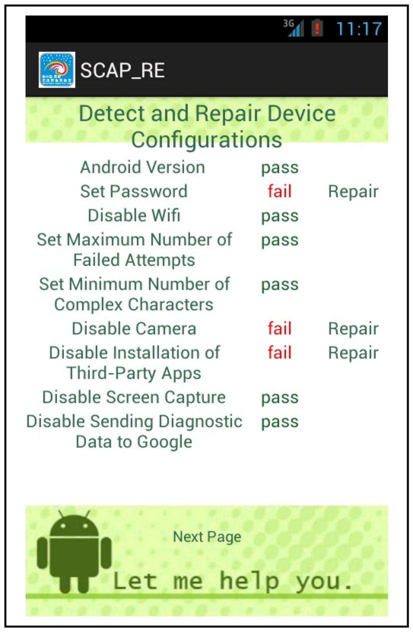


Figure 4. The system mode selection screen.

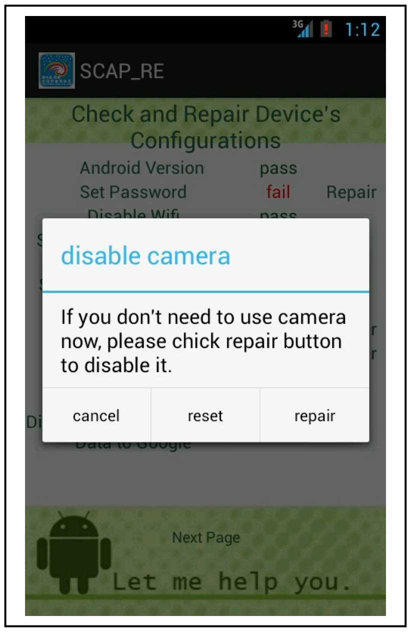
After selecting the detect and repair mode, the user goes into the interface shown in Figure 5. After clicking the title bar at the uppermost part of the Interface, the page will be restructured; after repairing configuration settings, this bar must be clicked in order to update test results. The items shown on the left column are the test items, while the middle column shows the test results. The repair button shown on the right column appears when the device does not pass the test. After clicking, repair would be done automatically or the interface will jump to the settings page in order to allow the user to perform manual repair, as shown in Figure 6.



Detect and Repair Device Configurations

Android Version	pass	
Set Password	fail	Repair
Disable Wifi	pass	
Set Maximum Number of Failed Attempts	pass	
Set Minimum Number of Complex Characters	pass	
Disable Camera	fail	Repair
Disable Installation of Third-Party Apps	fail	Repair
Disable Screen Capture	pass	
Disable Sending Diagnostic Data to Google	pass	

Figure 5. Detect and repair mode screen.



Check and Repair Device's Configurations

Android Version pass

Set Password fail Repair

Disable Wifi pass

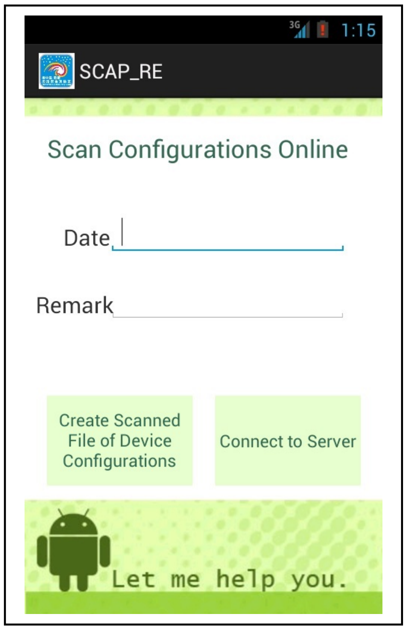
disable camera

If you don't need to use camera now, please chick repair button to disable it.

Data to Google

Figure 6. Repair contents and description.

If the user chooses to enter the online scan mode, he will enter the interface shown in Figure 7. Before entering the online scan server, a configuration file for the mobile device must first be generated in this interface. After data input, click the "Create Scanned File of Device Configurations" button and the system will automatically generate a configuration file for this mobile device, as shown in Figure 8, then the user can connect to the server for online scan.

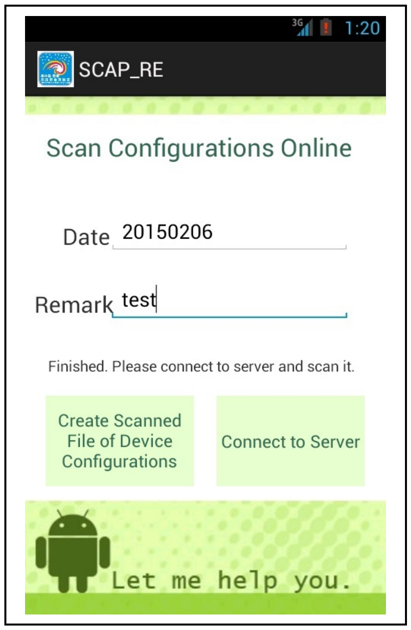


Scan Configurations Online

Date

Remark

Figure 7. Configuration file screen for online scan mode.



Scan Configurations Online

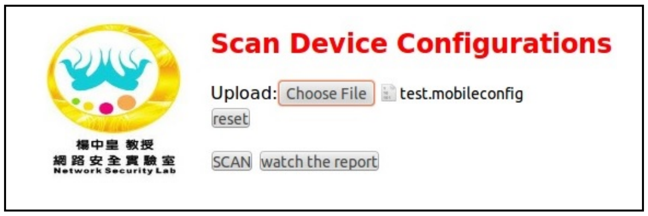
Date 20150206

Remark test

Finished. Please connect to server and scan it.

Figure 8. Finished create configuration setting file.

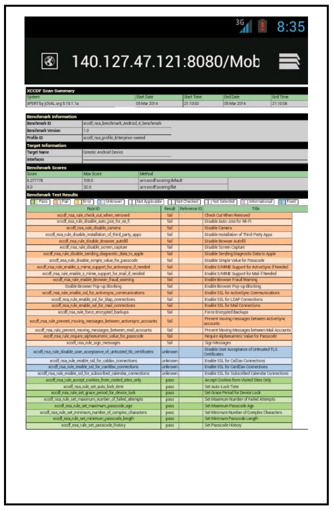
After connecting to the scan server, the scan page will be displayed, as shown in Figure 9. Click the upload button. After uploading the previously generated configuration setting file, click the SCAN button in order to start online scan. When scan is completed, click on the button "watch the report" to see scan results. Reports are shown in Figures 10.



Scan Device Configurations

Upload: test.mobileconfig

Figure 9. Configuration file screen for online scan mode.



[illegible]

Figure 10. Report for scan result.

After online scan is completed and results are obtained, the generated reports can be used for information security verification. When a failed item on the report needs to be corrected, one can return to the selection mode and enter the detect and repair mode to make corrections. The configuration settings can be corrected this way in order to eliminate vulnerabilities in the system and improve the security of mobile devices.

V. CONCLUSION

The research contribution of this study lies in developing a remediation system for Android mobile device configuration settings in line with the specified items in the CIS Google Android OS Benchmark based on SCAP. System vulnerabilities caused by improper user settings were eliminated and Android mobile device security was improved. Currently, the system can operate in CDD certified Android devices. In the future, we plan to do both short-term and long-term research. For the short-term research, the current detect and repair mode of APP side provides only simple test items and test results. Also, the repair button only appears when the standard is not met. A more detailed inspection and pass or fail result can only be known through the online scan mode. Therefore, the short-term goal is to modify and increase APP side interface and information. This will enable users to easily obtain test information during configuration repair in order to facilitate user operation and use. In the long run, this system currently uses the XCCDF in the SCAP standard. In the future, links with other SCAP standard formats will be tried in order to describe the system's weaknesses and vulnerabilities from different angles. Then, it would be easier for users to find the causes of these weaknesses and vulnerabilities and their solutions, thus making the work of maintaining device security easier.

ACKNOWLEDGEMENT

This work was supported by National Science Council grant (NSC 102-2221-E-017-003-MY3).

[1] Symantec, Internet Security Threat Report 2014, http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf.

[2] Gartner, Market Share Analysis: Mobile Phones, Worldwide, 4Q13 and 2013, https://www.gartner.com/doc/2665319.

[3] Gartner, 75 Percent of Mobile Security Breaches Will Be the Result of Mobile Application Misconfiguration, http://www.gartner.com/newsroom/id/2753017.

[4] M. N. Alsaleh and E. Al-Shaer, “SCAP Based Configuration Analytics for Comprehensive Compliance Checking,” In IEEE Configuration Analytics and Automation (IEEE SafeConfig), pp. 1–8, 2011.

[5] jOVAL, http://joval.org.

[6] Google, Compatibility Definition Document, http://source.android.com/compatibility/android-cdd.pdf.

[7] NIST, Security Content Automation Protocol, http://scap.nist.gov/revision/1.2/index.html.

[8] CIS, Google Android OS Benchmark, http://benchmarks.cisecurity.org/community/editors/groups/single/?group=android.

[9] Android Developers, Android System Architecture, https://source.android.com/devices/.

[10] S. Radack and R. Kuhn, “Managing security: The security content automation protocol,” IT Professional, 13(1), pp. 9–11, 2011.

[11] National Security Agency Central Security Service, http://www.nsa.gov/ia/mitigation_guidance/security_configuration_guides/operating_systems.shtml.

[12] R.J.G. Vargas, R.G. Huerta, E.A. Anaya and A.F.M. Hernandez, “Security controls for Android,” Computational Aspects of Social Networks (CASoN), pp. 212-216, 2012.

[13] S. G. Punja and R. P. Mislan, “Mobile Device Analysis,” Small Scale Digital Device Forensics Journal, 2(1), pp.1-16, 2008.

[14] G. Koschorreck, “Automated audit of compliance and security controls,” In IT Security Incident Management and IT Forensics (IMF), 2011 Sixth International Conference on, pages 137 –148, 2011.