Advances In Multidisciplinary & Scientific Research
A Multidisciplinary & Interdisciplinary Journal
Volume 6, 2020 Edition

Content Analyzer for Information Leakage Detection and Prevention in Android Smart Devices: A Conceptual Approach.

Okebule, T., Adeyemo, O.A, Olatunji, K.A. & Awe, A.S.
Department of Mathematical and Physical Sciences
Afe Babalola University,
Ado-Ekiti, Ekiti State, Nigeria.
E-mails: okebulet@abuad.edu.ng, adeyemo@abuad.edu.ng, olatunjika@abuad.edu.ng
Phone: +2348064771341; +2348121592400; +2348035161562

ABSTRACTS

The advent of android operating system introduced tools to keep track of users' information activities and prevent information leakage which bridged the trust between application developers and consumers. Literature shows that several phenomena had been developed to prevent malicious applications from stealing personal sensitive information from smart phones but there is still the need for efficient solutions. This study proposes a conceptual approach for the development of a contentAnalyzer for information leakage detection and prevention on android-based devices. The concept will help to minimize false positives that will in turn lead to increase in code coverage towards detecting the maximum number of data leaks. The proposed concept combines both static and dynamic analysis, and if implemented will improve checking through the codes in the file activities and vulnerabilities that could be a problem.

Keywords: Android, ContentAnalyzer, Static Analysis, Dynamic Analysis, Information leakage, Information leakage detection, Information leakage Prevention.

Aims Research Journal Reference Format:
Okebule, T., Adeyemo, O.A, Olatunji, K.A. & Awe, A.S. (2020): Content Analyzer for Information Leakage Detection and Prevention in Android Smart Devices: A Conceptual Approach. Advances in Multidisciplinary Research Journal. Vol. 6. No. 1, Pp 79–90.
Article DOI: dx.doi.org/10.22624/AIMS/V6N1P7

1. BACKGROUND OF THE STUDY

In recent years, handheld devices usage has exceeded that of desktops, while security and privacy concern about data in these devices are increasing exponentially in personal and corporate environment. Android has been the most commonly used operating system in smartphone due to its ease of use in the transferring and receiving of information on various forms. Today, professionals in diverse spheres of life currently prefer to use their personal Smartphones and tablets as the case may be for carrying out corporate work related tasks like email, documents, calendar, corporate apps among others, which has greatly helped in achieving a balance between personal and corporate life [33]. Programs that steal information also known as malicious software affects the user's mobile devices by exploiting the vulnerabilities. It is the major threat to the security of information in a system.

The types of malware that are most commonly used are viruses, worms, Trojans, among others. There is another widespread use of malware which allows malware author to get sensitive information like bank details, contact information among others. Most of the malware that affect mobile devices are embedded into mobile application or files accessed from the mobile device. These programs can destroy or steal sensitive and private information in any system. A lot of advances can be seen these days in the field of smart phones and as the number of users is increasing day by day, facilities are also increasing [33]. Information helps to clear any form of uncertainty and answers the question of "what an entity is" and thus defines both its essence and nature of its characteristics. Information relates to both data and knowledge, as data represents values attributed to parameters, and knowledge signifies understanding of a concept.

In terms of communication, information is expressed either as the content of a message through direct or indirect observation and that perceptive can be construed as a message in its own right, and in that sense, information is always conveyed as the content of a message [32]. Information can be encoded into various forms for transmission and interpretation (for example, information may be encoded into a sequence of signs, or transmitted via a signal). It can also be encrypted for safe storage and communication [9].

Leakage is the act or process or an instance of leaking in other words it is something or the amount that leaks. Leakage is premium revenue that is lost, often because a policyholder has not been truthful about facts or lifestyle changes or has committed some fraud. Information leakage in this study may be defined as the accidental or unintentional distribution of private or sensitive information to an unauthorized entity that can be caused by negligence or intentional sabotage such as, emails sent to the wrong recipients. Also, asides negligence, it is a universal truth that the motivation to leak sensitive information will exist no matter what countermeasures are taken. The result is that as storage media continually becomes more mobile and smaller in size, more sensitive information is likely to be stored on such media, having a greater likelihood of being lost or stolen [32].

Information Leakage Detection has been described as a situation whereby on-chip malicious circuits are hidden and illegally written to the main memory. It has not been ascertained whether data fetches and reads are a concern, and if some confidential information are read from the memory, it cannot be leaked to the external world unless it is leaked out on to the memory bus. Often time, there may be no external data interface (e.g., data/network ports) on the chip itself other than the address and data bus [30]. Information leakage prevention (ILP) is a set of vital information security tools intended to prevent unauthorized users from sending sensitive or critical information from a private user's devices. Adoption of Information Leakage Prevention, variously called Information loss prevention, information loss prevention or extrusion prevention, is being driven by significant insider threats and by more rigorous state privacy laws, many of which have stringent information protection or access components. Information Leakage Prevention products use business rules to examine file content and tag confidential and critical information so that users cannot disclose it. Tagging is the process of classifying which data on a system is confidential and marking it appropriately. A user who accidentally or maliciously attempts to disclose confidential information that is being tagged will be denied. For example, tagging might even prevent a sensitive financial spreadsheet from being emailed by one employee to another within the same corporation.

Recent studies revealed that malicious applications exists on them. These malicious applications will leak private information without user's authorization. A good example is, TaintDroid shows that among 30 popular third-party Android applications, there are 68 instances of potential misuse of users' private information. In light of these privacy-violating threats, there is an imperative need to tame these information-stealing smartphone applications [21]. Android requires explicit permission applications installed to ensure the user is aware of the information or access that will be needed to run the application, and by showing these permissions to the end user, and Android delegates, the task to the user for approval when the application is being installed. However, this permission mechanism is too coarse-grained for two main reasons. First, the Android permission mechanism requires that a user has to grant all the requested permissions of the application if he wants to use it, otherwise, the application cannot be installed. Secondly, if a user has granted the requested permissions to an application, there is no mechanism in place to later re-adjust the permission(s) or constrain the runtime application behavior[31]. Given the increased sophistication, features, and convenience of these smart-phones, users are increasingly relying on them to store and process personal information and since these are all private information, the safety of these data is concerned [38].

Recent years have witnessed the rapid spread of smartphones, and Android has emerged as a popular smartphone operating system (OS). Application developers can easily develop an Android application and make it available through a Web site such as Google Play Store. However, an application built with malware is capable of accessing administrative privileges by exploiting vulnerabilities in the Android Operating System, and thus send out illegally collected sensitive information. In particular, a major issue is the widespread emergence of malware which performs unwanted or unexpected processing. Furthermore, applications that are not sufficiently malignant to be blacklisted as malware call application program interfaces (Applications) that collect sensitive information inappropriately. This makes it difficult to ensure transparency when the application handles user information. Malware that targets the Android Operating System is usually intended to illegally collect sensitive information. A mobile device contains a large amount of personal information such as names, addresses, and phone numbers, and this information can be easily obtained by applications using the Android Applications. Moreover, many users are unaware of smartphones' lack of security and built-in anti-malware software. Therefore, there is a possibility of information leakage due to malware, and without the user's knowledge [31].

An Android application is executed in a sandbox; communication with other applications is severely restricted, and requires the use of an intent. Key features such as external communications and the collection of sensitive information requires permissions that are granted by the user. However, the user can neither detect the collection of sensitive information by the application nor determine whether that sensitive information has been leaked [31]. Android has been the most commonly used operating system in smartphone due to its ease of use in the transferring and receiving of information on various forms. Information Leakage, Detection and Prevention (ILDP) is the new rising star in Information security. Since the advent of android Operating System, a lot of tools have been developed to keep track of users' information and activities, and prevent information leakage which bridged trust between applications developers and the consumers. Hence, there is a need to come up with an effective solution in order to address information leakage issues particularly in smartphones. The aim of this study is to develop a conceptual approach of ContentAnalyzer for information leakage detection and prevention on Android Based Devices.

2. RELATED WORKS

[18] develop SCANDAL, a Static Analyzer for Detecting Privacy Leaks in android applications. Static analyzer SCANDAL is a technique for providing a formal, found, and automatic static analysis. It has been referred to as a sound and automatic static analyzer for detecting privacy leaks in Android applications. This tool analyzed 90 popular applications using SCANDAL from Android Market and detected privacy leaks in 11 applications and also analyzed 8 known malicious applications from third-party markets and detected privacy leaks in all 8 applications. The limitation of this model is that, SCANDAL does not fully support reflection-related APIs and the time performance and memory consumption during the analysis is very low and this considered for future works.

[5] Performed system call-centric dynamic analysis of Android applications, using Virtual Machine Introspection. The novelty of CopperDroid lies in its doubting approach to identify interesting OS- and high-level Android-specific behaviors. It reconstructs these behaviors by observing and dissecting system calls and, therefore, is resistant to the multitude of alterations the Android runtime is subjected to over its life-cycle because CopperDroid's has reconstruction mechanisms that are doubting to the underlying action invocation methods, it is able to capture actions initiated both from Java and native code execution. Using this technique, it successfully triggered and disclosed additional behaviors on more than 60% of the analyzed malware samples. This qualitatively demonstrates the versatility of CopperDroid's ability to improve dynamic-based code coverage. The limitation of this tool is that, CopperDroid system call tracking would not provide any behavior insights if was not combined with Binder information and automatic (complex) Android objects reconstruction.

[28]designed DroidSafe for static information flow analysis tool that reports potential leaks of sensitive information in android applications. DroidSafe combines a comprehensive, accurate, and precise model of the android runtime with static analysis design decisions that enable the DroidSafe analyses to scale to analyze this model and by a combination of analyses together can statically resolve communication targets identified by dynamically constructed values such as strings and class designators. DroidSafe's reporting is defined by the source and sink calls identified in the Android API. An attacker could exfiltrate API-injected information that is not considered sensitive by DroidSafe, or via a call that is not considered a sink; and it would not be reported. The analysis does not have a fully sound handling of Java native methods, dynamic class loading, and reflection. Different versions exist of Android, and the system analyze an application in the context of Android 4.4.3.

[33].developed DroidScope, a framework to create dynamic analysis tools for Android malware that trades off simplicity and efficiency for transparency that extends traditional techniques to cover Java semantics. However, the problem of analyzing Android applications is not simple as how to capture behaviors from different language implementations. It is hard to conduct effective analysis without considering Android's specific security mechanism. Permission Event Graph, which represents the temporal order between Android events and permission requests, is proposed to characterize unintended sensitive behaviors. However, this technique could not capture the internal logic of permission usage, especially when multiple emissions are intertwined.

3. System Architecture

In this section we describe the proposed conceptual approach. We start by presenting an architectural overview of modules contained in it and a presentation of its main components and functions.

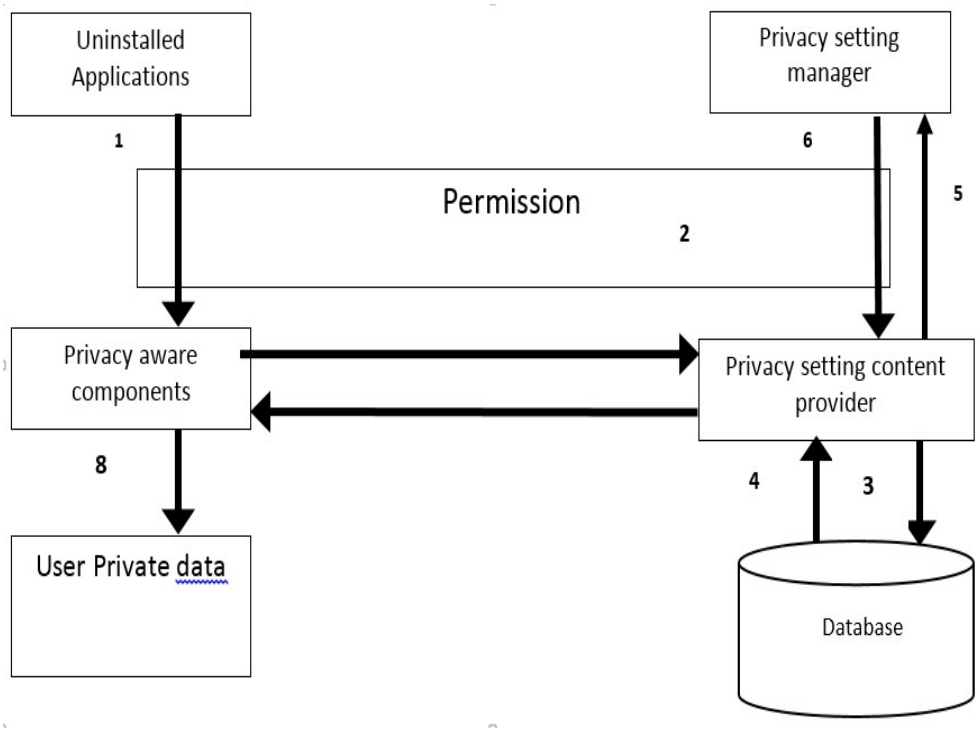

Figure 3.1 A Conceptual Approach of ContentAnalyzer for Information Leakage Detection and Prevention on Android Based Devices.

Research has shown that for any system to be effective, efficient and reliable, modularity is a concept that must be seriously taken into consideration during the implementation of such systems. The system architecture consist of three main modules, each constituted by various elements that worked together to make the system perform flawlessly.

3.1 Privacy Setting Content Provider

The first module is the privacy setting content provider, which is a privileged component to manage the privacy settings for untrusted applications. In the meantime, it also provides an Application Programming Interface that can be used to query the current privacy setting for an installed application. The privacy setting content provider is tasked to manage a local SQLite database that contains the current privacy settings for untrusted applications on the phone. It also provides an interface through which a privacy-aware component can query the current privacy settings for an untrusted app, the privacy-aware component will provide the input that is the package name of the requesting application and the type of private information it is trying to acquire.

Once received, the privacy setting content provider will use the package name to query the current settings from the database. The query result will be an app-specific privacy setting regarding the type of information being requested.

3.2 Privacy Setting Manager

The second module is the privacy setting manager, which is a privileged application that a mobile user can use to manage or update the privacy settings for installed applications. Therefore, it acts as the Policy Administration Point (PAP). The privacy setting manager is a standalone Android application that is signed with the same certificate as the privacy setting content provider in which manager will be given the exclusive access to the privacy setting.

The manager provides the visual user interface and allows the user to specify the privacy settings for untrusted applications. In particular, the manager application includes two activity components. The default one is Privacy Setting Manager Activity, which when activated displays a list of Installed applications. The phone user can then browse the list and click an application icon, which starts another activity called App Privacy Setting Activity and passes the app's package name. When the new activity is created, it queries the privacy setting content provider for the current privacy settings and displays the results to the user.

3.3 Privacy-Aware Components

The third component is privacy-aware module, including those content providers or services that are enhanced in a privacy-aware manner to regulate the access to a variety of user's personal information, including contacts, call log, and so on. These privacy-aware components are designed to cooperate with the first component. In particular, once they receive requests from an application to access private data they manage, they will query the privacy settings, and response to the requests according to the current privacy settings for the application. When an application tries to read a piece of private data, it sends a reading request to the corresponding content provider. The content provider is aware of the privacy requirement. Instead of serving this request directly, it holds the request and makes a query first to the privacy setting content provider to check the current privacy settings for the application (regarding the particular reading operation).

The privacy setting content provider in turn queries its internal policy database that stores user specifications on privacy settings of all untrusted applications, and returns the query result back to the content provider. If this reading operation is permitted (stored in the policy database), the content serves the access request and returns normal results to the app. This may include querying its internal database managed by the content provider. However, if the reading operation is not permitted, the privacy setting option report unauthorized.

3.4 Sequence Diagram

Figure 3.2 shows the sequence of the system and it consists basically three modules as explained:

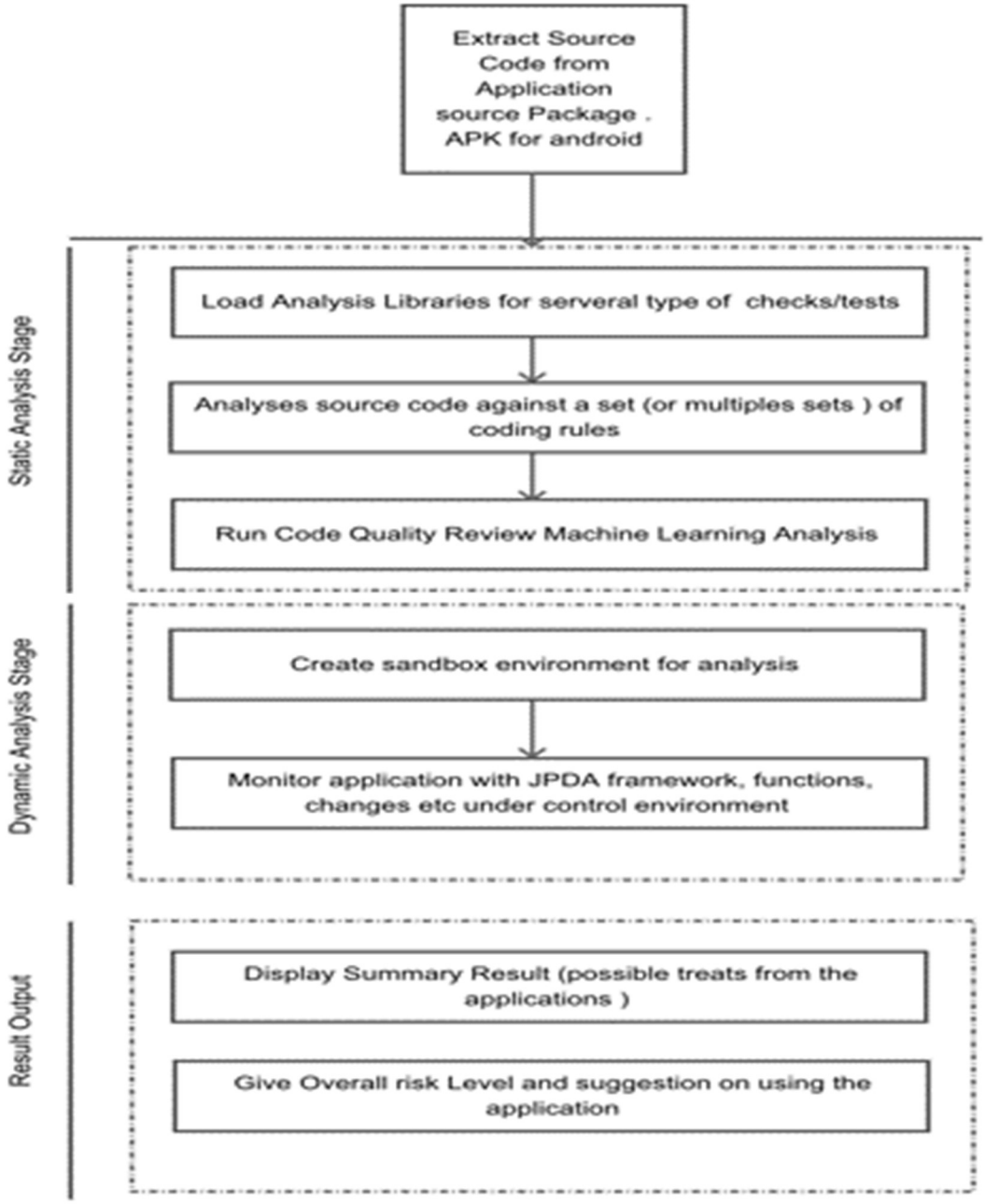

Figure 3.2: Sequence Diagram of the Conceptual Approach.

This system has three stages namely static Analysis stage, Dynamic Analysis Stage and the Report Stage. Each stage output will be an input to the next stage. Each stage runs independently and pass its result to the next one. There will be a code extraction process before the first stage, the last stage will provide the result of the whole analysis and the category of the application.

3.2.1 The Extraction process

This process does reverse engineering to the apk architecture and structure, the intention of this process is to understand (so as to re-build the source code) the buildup of the entire code base and to focus on the part that is relevant for the static analysis stage . The output of this process are details on the architecture of the applications, details of the broken down of the structure, provide information about its functionality and collect technical indicators. The proposed process comply with Android's security architecture and user data privacy is maintained. [18].

.

3.2.2 Static Analysis Stage

Static malware analysis is fairly straightforward and fast. It ensures security and safety of the android devices because of its ability of detecting malicious file without viewing or running, the actual code or without execution, this stage takes the input from the extraction stage. The static analysis stage has some already loaded check libraries (Rules for identifying malicious code or software). It runs the input from the extraction process against these set of rules to detect if there are suspicious instructions files etc. This phase covers both the checks/test and the multiple set operations. Still within the static Analysis stage, there is a code quality review process to examine the code quality which could also be a treat to the devices. One of the important issues about code quality is how easy it is for humans to read and understand the code, and an important measurement for this is the complexity measurement. It is not a trivial task to detect the quality of the code, since it depends on both functional requirements, structural requirements, and complexity. For basic static code analysis, 15-20 rules are used [18].

3.2.3 Dynamic Analysis Stage

Dynamic analysis techniques involve running the malware and observe its behaviour on the system. Typically, malwares have abilities to change several sorts of things on the compromised device, it actually run on a host device such behaviours includes variable modifications, accesses to api calls. Since mobile devices allow easy-to-use, touch-sensitive, and anywhere-anytime access to its resources, some of the resources being monitored includes but not limited to SMS, MMS, Bluetooth, e-mail, Network traffic, file modification and other services that may pose serious threats and lead to financial losses and privacy leakages[27].

In Dynamic analysis one can monitor what network traffic they are sending and receiving, with what kind of Uniform Resources Location (URL) or server they are communicating, you can also check that after installation app is trying to reboot or if it is trying to change some internal file format or trying to run some privileged instruction. The sandbox environment will prevent the application from actually infecting production systems; many such sandboxes are virtual systems that can easily be rolled back to a clean state after the analysis is complete.

So, dynamic analysis consists of monitoring the following behaviors:

- Volatile Memory: Malware can overflow buffers and use the abandoned memory locations to gain access to the device. By capturing and analyzing the device memory, it is possible to determine whether and how the malware uses the memory. Observation of these behaviours can give valuable information about software's intention, which is difficult to be gathered by other detection schemes.
- Registry/configuration changes: Changes in the registry may be an evidence toward dynamic analysis. Malwares often change registry values to gain persistent access to the system.
- File activity: Malware may also add, alter, or delete the files. So by monitoring file activities, valuable information about the malicious behavior can be obtained.
- Processes/services: Malware may disable AV engines to fulfill their functions, jump to other processes to obstruct analysis, or install new services to obtain persistent access to the system.
- Network connection: Monitoring the network connections is the essential part of dynamic analysis to detect the malware's existence. Destination IP addresses, port number, and protocol can be analyzed in order to detect malware's interaction with the command-and-control (C&C) server.

3.2.4 The Result State

The result phase displays a summary of the threats by the applications, the code quality review. This stage also presents the overall risk level of the application and suggestion on what the user can do. Also if user should disable some of the application's access to the devices recourses.

4. APPLICABILITY OF THE CONCEPT.

Below are the application areas where the proposed conceptual approach can be adopted.

4.1 Banking and Financial Services

This concept will aid the development of security in the bank by preventing invasions that uses personal information (e.g., Social Security numbers, bank account information and credit card numbers) to pose as another individual. This may include opening a credit account, draining an existing account, filing tax returns or obtaining medical coverage.

4.2 Electoral Voting:

This concept will assist, if well implemented, it will go a long way in preventing vote rigging which normally occurs as a result of multiple voting and other interferences during election voting. All this has really affected the integrity of elections. Hence, the concept will attempt to provide solution to this problem.

4.3 Student Result Processing

If this concept is used, it will allow generation of accurate and error free student results information by preventing unauthorized access into the server housing. Also, preventing unauthorized access that can effect changes in grades of students which may occur as a result of student or staff mischievous act of changing marks or grades on the result sheet.

5. CONCLUSION

In this paper, we presented the conceptual architectural design and the sequence for information leakage detection and prevention on android-based devices. We believe that if this concept is implemented, false positives will be minimized and in turn lead to increase in code coverage to detect the maximum number of data leaks. Furthermore, this concept could enhance the capacity for the developed contentAnalyser to detect and prevent more information leakages on smartphones than previously considered works. The extent of the implementation of this concept in the development of a ContentAnalyser for smart phones will also allow for various debate and sensitization in the disciplines and the body of knowledge of computer science. This work proposed a concept that can lead to the solution stated in statement of the problem. As future work, this approach needs a more specific way of systematization, perhaps through the sophisticated software. Additionally, the used assumptions in this contribution must face well-grounded scientific evaluations in order to ensure their stability since the described approach is conceptual and has not been evaluated yet.

REFERENCES

1 Adam, P., Fuchs, Avik, C., and Jeffrey, S. (2009). SCanDroid; *Automated Security Certification of Android Applications.* Technical Report CS-TR-4991, Department of Computer Science, University of Maryland, Vol. 12(1), pp 103-108.

2 Adrienne, P. F., Erika, C, Steve, H., Dawn, S and David, W. (2011). *Android Permissions Demystified.* Proceedings of the 18th ACM conference on Computer and communications security, Vol 11, pp 627-638.

3 Anand, S., Naik, M., Yang, H., and Harrold, M., (2012). *Automated concolic testing of smartphone apps.* Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. Vol 12, Pp59.

4 Andrei, S. and Andrew, C. (2003). *Language-based information-flow security*. IEEE J. Selected Areas Comm. Vol 21(1) pp 21, 1, 5–19.

5 Aristide, F., Kimberly, T., Salahuddin, J., Khan, A., and Lorenzo, C., (2014). *CopperDroid: On the*

6 Arzt, S. (2009); "FlowDroid, Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps,". Understanding Android Security, IEEE Security and Privacy. Vol 7(1), pp 50-57.

7 Arzt, S., Siegfried, R, Christian, F., Eric, B., Damien, O., Patrick, M., Alexandre, B., Jacques, K. and Yves, L. (2014). TraonFlowDroid: *Precise Context, Flow, Field, Object-sensitive and Lifecycle-aw are Taint Analysis for Android Apps*.

8 Asavoae, I. M., Blasco, J., Chen T. M., Kalutarage, H. K., Muttik, I., Nguyen, H. N., Roggenbach, M. and haikh, S. A., (2016). *Towards automated android app collusion detection*: Proceedings of the Workshop on innovations in Mobile Privacy and Security IMPS at ESSoS16, London, UK.

9 Burguera, I. Zurutuza, U. and Nadjm-Tehrani, S. (2011). "*Crowdroid: behavior-based malware detection system for Android,":* Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices Chicago, Ill, USA. Vol 11, pp. 15–26.

10 Carvalho, V., Balasubramanyan, R. and Cohen, W. (2009), *Information Leaks and Suggestions:* A Case Study using Mozilla Thunderbird. Paper presented at the CEAS 2009 - Sixth Conference on Email and Anti-Spam, pp.46-53.

11 Egele, M. Scholte, T. Kirda, E. and Kruegel, C. (2012). *"A survey on automated dynamic malware-analysis techniques and tools,"* ACM Computing Surveys, vol. 44(2). Pp6.
12 Enck, W. Gilbert, P. Han S. (2014). *"TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones*," ACM Transactions on Computer Systems, vol. 32(2), pp 5.
13 Enck, W. Octeau, D. McDaniel, P. and Chaudhuri, S. (2011). "*A study of Android application security*," in Proceedings of the 20th USENIX Conference on Vol 11, pp. 21.
14 Enck, W., Peter, G., Byung-Gon, C., Landon, P., Jaeyeon, J., Patrick, M., Anmol, N. (2010): TaintDroid: *An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones*: 9th USENIX Symposium on Operating Systems Design and Implementation.
15 Ferreira, D., Kostakos, V., Beresford, A., Lindquist, J. and Dey A., (2015) Securacy: *An Empirical Investigation of Android Applications' Network Usage, Privacy and Security*. Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, New York, pp 22-26.
16 Firdausi, I., Lim, C. and Erwin, A. (2010). *Analysis of Machine Learning Techniques Used in Behavior Based Malware Detection*. Proceedings of 2nd International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT). Vol 5(2).
17 Haritha, R. and Bhagavan, K. (2019). *Anti –Reverse Engineering Techniques Employed by Malware*: International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. (8), pp2278-3075.
18 Jinyung K., Yongho Y., and Kwangkeun Y. (2018) SCANDAL: Static Analyzer for Detecting Privacy Leaks in Android Applications.
19 Michael, G. (2012); *"Systematic detection of capability leaks in stock Android smartphones"*: Proceedings of the 19th Annual Symposium on Network and Distributed System Security, pp 23-34.
20 Michael, I., Kim, D., Jeff, P., Limei G., Nguyen, N., and Martin, R. (2015). DroidSafe: *Information-Flow Analysis of Android Applications in DroidSafe*. Vol (15), pp 8-11.
21 Nauman M., Khan S., and Zhang X. (2010); Extending Android Permission Model and Enforcement with User-Defined Runtime Constraints. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security.
22 Nickolai, Z., Silas, B., Eddie, K., and David, M., 2006. *Making information flow explicit in Histar*. Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06). Vol 54(11) 263–278.
23 Peng, H., Gates, C., Sarma, B., Li N., Qi Y., Potharaju, R., Nita-Rotaru, C., and Molloy, I. (2012). *Using probabilistic generative models for ranking risks of android apps*. In ACM CCS. pp. 241–252.
24 Roemer, R. Buchanan, E. Shacham, H. and Savage, S.(2012). *"Return-oriented programming: systems, languages, and applications,"* ACM Transactions on Information and System Security, vol. 15(1), pp2.
25 Rose, S., Chandramouli, R. and Nakassis, A., (2009). *Information Leakage through the Domain Name System.* Paper presented at the Cybersecurity Applications & Technology Conference For Homeland Security: P*roceedings* of the *8th Australian Information Security Management. Pp2*
26 Shu, X. Elish, K. O. Yao, D. Ryder, B. G. and Jiang, X. (2015). "*Profiling user-trigger dependence for Android malware detection,"* Computers & Security, vol. 49, pp. 255–273,.

27 Spreitzenbarth, M. Schreck, T. Echtler, F. Arp, D. and Hoffmann, J. (2015). *"Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques,"* International Journal of Information Security, vol. 14(2), pp141–153.
28 Steven A., Siegfried R., Christian F., Eric B. and Yves L. TraonFlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps PLDI '14, June 9-11 2014,.
29 Wei, F., Roy, S. and Zhou, X. (2014) Amandroid: *A precise and general intercomponent data ow analysis framework for security vetting of android apps,*: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, pp. 1329-1341.
30 Wei, X, Sandeep, B., and Sekar R. (2006). *Taint-enhanced policy enforcement: A practical approach to defeat a wide range of attacks.* Proceedings of the USENIX Security Symposium. Pp 121–136.
31 William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Anmol N. Sheth (2010): TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In: 9th USENIX Symposium on Operating Systems Design and Implementation.
32 Wong M. Y., Lie D., Intellidroid: A targeted input generator for the dynamic analysis of android malware, Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS 2016).
33 Yan, L. K. and Yin, H. (2012). "*DroidScope: seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis*,": Proceedings of the 21st USENIX Conference on Security Symposium, Bellevue, Wash, USA, pp. 29.
34 Yan, L., and Yin, H.: *DroiScope: seamlessly reconstructing the OS and dalvik semantic vies for dynamic android malware analysis:* proceedings of the 21st USENIX Security Symposium, Vol. 29.
35 Yang, Z. Yang, M. Zhang, Y. Gu, G. Ning, P. and Wang X. S. (2013). "*AppIntent: analyzing sensitive data transmission in Android for privacy leakage detection,"*: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security . Vol 13, pp. 1043–1054.
36 Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., and Wang, X. (2013) Appintent: *Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection.* Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, pp 1043-1054.
37 Zheng, C., Zhu, S., Dai, S., Gu, G., Gong, X., Han, X., and Zou, W. (2012). SmartDroid: *an automatic system for revealing UI-based trigger conditions in android applications*: Proceedings of the send ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. Pp 93-94.
38 Zhou, W. Zhou, Y., Jiang, X., and Ning, P. (2010); DroidMOSS: *Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces.* Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy.
39 Zhou, Y. and Jiang, X. (2012). "*Dissecting Android malware: characterization and evolution,"* Proceedings of the 33rd IEEE Symposium on Security and Privacy, San Francisco, Calif, USA. pp. 95–109.