IMPLEMENTATION OF CONTENTANALYZER FOR INFORMATION LEAKAGE DETECTION AND PREVENTION ON ANDROID SMART DEVICES

Okebule T[1]*, Adeyemo. O.A[2], Olatunji K.A[3].&Awe. A.S[4]
Department of Computer Science
AfeBabalola University,
Ado-Ekiti, Ekiti State, Nigeria.

Emails: **okebulet@abuad.edu.ng; adeyemo@abuad.edu.ng; olatunjika@abuad.edu.ng; aweadesegun@gmail.com**
Phone: +2348064771341; +2348121592400; +2348035161562; +2349078971540

ABSTRACTS

Android has been the most commonly used operating system in smartphone due to its ease of use in 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. 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 used a conceptual approach for the implementation of a ContentAnalyzer for information leakage detection and prevention on android-based devices. This implementation will improve through the codes in the file activities and vulnerabilities that could be problem on android-based devices.

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

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 (Sunita*et al.,* 2016).
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 . 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.
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 (William *et al.,* 2010). 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. (Zhou *et al.,* 2010).
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 (William *et al.,* 2010).
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) isthe 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 an enhance ContentAnalyzer for information leakage detection and prevention on Android Based Devices.

2. Related Works

SCanDroid is an Automated Security Certification of Android Applications with the static approach that cannot analyze packaged Android Applications and has not been tested on real-world market applications (Fuchs *et al.,* 2009). 09).
TaintDroid is an Information-Flow Tracking System for real time privacy monitoring on Smart phone with dynamic approach. TaintDroid monitors Android smartphone and tracks how applications leak private but cannot track taintedness of native code and fails to provide users visibility into how third-party applications (Enck, W. et *al.*, 2014.).
DroidRay: A Security Evaluation System for Customized Android Firmwares. DroidRay uses both static and dynamic methodologies to analyze the system security of the Android firmware. DroidRay cannot handle applications with permissions (e.g., sending SMS message and silent installation behavior. (Min Z. *et al.,* 2014).
Aquifer is a developed tool for preventing accidental data disclosure in modern operating systems. Aquifer as a policy system as well as framework for avoiding accidental data disclosure in modern operating systems. In Aquifer, application developers give secrecy restrictions which protect the entire user interface workflow during the user task. (Nadkarni*et al,* 2013).

An approach towards Automated Android App Collusion Detectionis a statistical approach consisting of defining probabilistic model, training of the model that means estimating the model parameter on the training set and validating the model on test dataset. In addition, the study also presented that model-checking is the feasible approach to detect collusion in Android apps. It identified that collusion can cause information theft, money theft or service misuse. They defined collusion between apps as some set of actions executed by the apps that can lead to a threat. However, the statistical approach performance could be due to a bias of validation dataset towards the methodology but this approach did not addresses the issue of scalability (Asavoae*et al.,* 2016).
Amandroid is a precise and general inter component data flow analysis framework for security vetting of android apps. Amandroid proceeded by converting an app's Dalvik bytecode to an intermediate representation (IR) for subsequent analysis. Amandroid go on to generate an environment model that emulates the interactions of the Android System with the app to limit the scope of the analysis for scalability (Wei *et al.,* 2014).
DREBIN is an effective and explainable detection of android malware in your pocket. The method employs a broad static analysis that extracts feature sets from different sources and analyzes these in an expressive vector space. However, this method has not been tested it on other version of handsets (Daniel *et al.,* 2014).
AndRadar is a fast discovery of android applications in alternative markets. AndRadar was a prototype configured to discover apps by their package name as the monitored markets distinguish apps by this identifier. AndRadar exposed the publishing patterns followed by authors of malicious applications on 16 markets. Moreover, their evaluation shows that AndRadar makes harvesting marketplaces for known malicious or unwanted applications fast and convenient (Martina *et al., 2014).*

3. System Architecture

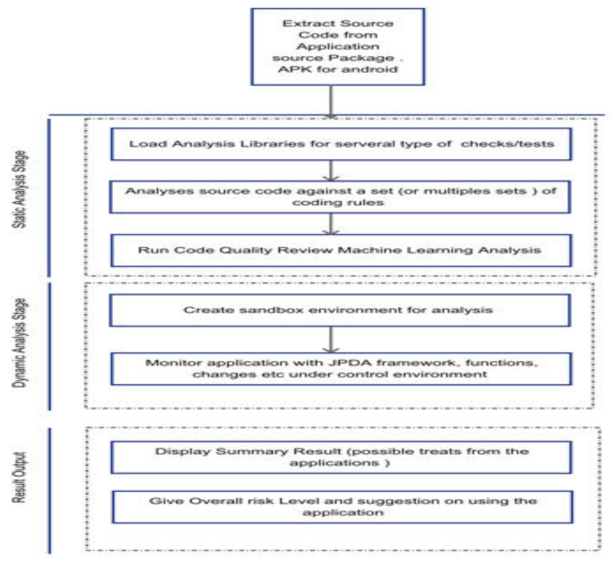

Extract Source Code from Application source Package . APK for android
Static Analysis Stage
Load Analysis Libraries for serveral type of checks/tests
Analyses source code against a set (or multiples sets) of coding rules
Run Code Quality Review Machine Learning Analysis
Dynamic Analysis Stage
Create sandbox environment for analysis
Monitor application with JPDA framework, functions, changes etc under control environment
Result Output
Display Summary Result (possible treats from the applications)
Give Overall risk Level and suggestion on using the application

Figure 3.2: A system Architecture for ContentAnalyzer for Information Leakage Detection and Prevention on Android Based Devices.

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.

.

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

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

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. Content Analyzer

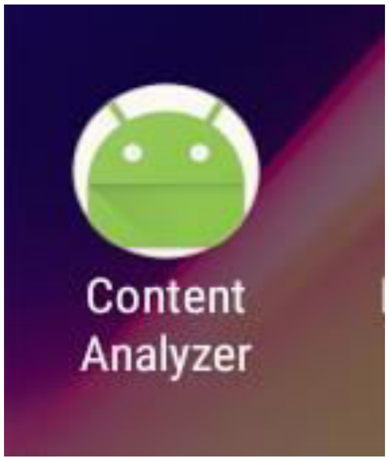
ContentAnalyzer is an Android accessibility service that enabled it runs on top of Android activities and analyzes their content for accessibility issue. This is how it works: the view hierarchy is scanned every time an event occurs that intended to manipulate the user sensitive information on smartphone. Accessibility app is then highlighted on the screen, with a border around the offending view. The Analyzer select an appropriate app based on the users' decision. It then performs the scanning analysis on the app selected and allow the user of the Smartphone to view real time results as per whether the app selected contains a malicious code or not and this reflects what is being shown on the device. The Analyzer then produces a report with these details, with reports on how to fix the issues.

Beyond just a visual scan, users can also discover more a detail about which app is affected. They can learn about the types of issues and how to fix them by accessing the Content Analyzer's output. The next section will analyze the workings of the tool for the purpose of .apk code analysis.To achieve the development goals, the tools and technologies that were used are Android studios version 3.0.1 the application was developed on the desktop, and then installed on the android operating system mobile device. Android studio makes it possible to create applications that work on the Android Operating system, it is the official IDE for the Android platform and is the direct successor to the Eclipse IDE which was previously used for Android application development. It was used for coding the system as well as the various testing phases involved in developing the system. Android Studio create a unified environment where one can develop applications for every Android device without the need to rewrite code for each device, it is a highly scalable platform. The following snapshots of the Android mobile information leakage detection software taken while the android mobile phone application was being tested are illustrated below. This application was developed to run on Android version 4.2.

4.1 **Implementation**.

Figure 4.1 presented the ContentAnalyzer Application icon on the users' android smartphone. Once the Application has been installed, it automatically popped up the icon on the home screen of the Users' devices.

Figure 4.2presented the scan and the history phase.
Figure 4.3 indicated the total number of the app installed on the users smartphone at the point when the Content Analzer is performed on them.
Figure 4.4.presents the selection of file on user smartphone
Figure 4.5 is where the real scanning of the installed applications on the android devices is performed, this scan the numbers of activities in the application and scan all the application on stored on the users smartphone.
Figure 4.6 shows that scanning is completed it returns a detailed result of the scanned app. This depicts the virtualization for the outcome of the application scanned. The result covers the total number of the name of the scanned app, scanned status report, Analysis status. In this part, it basically displayed the result of the scanned so that the user of the applications will be able to know the status of the application at a particular point in time as per whether the application is good or contained any malware code that could cause harm to the stored information on the users devices. If there is any scanned status report indicating that the application contained any code that could cause harm to the system, the user quickly delete such app from the devices to avoid any form of any leakages.



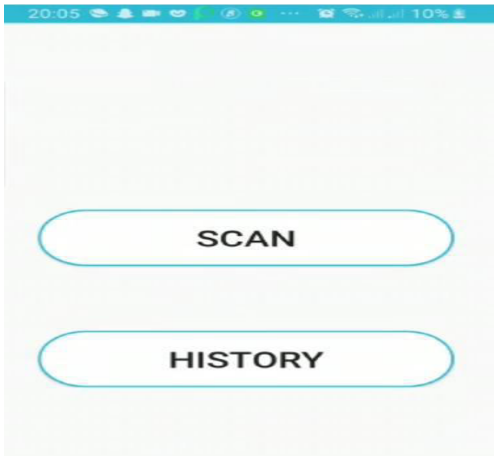
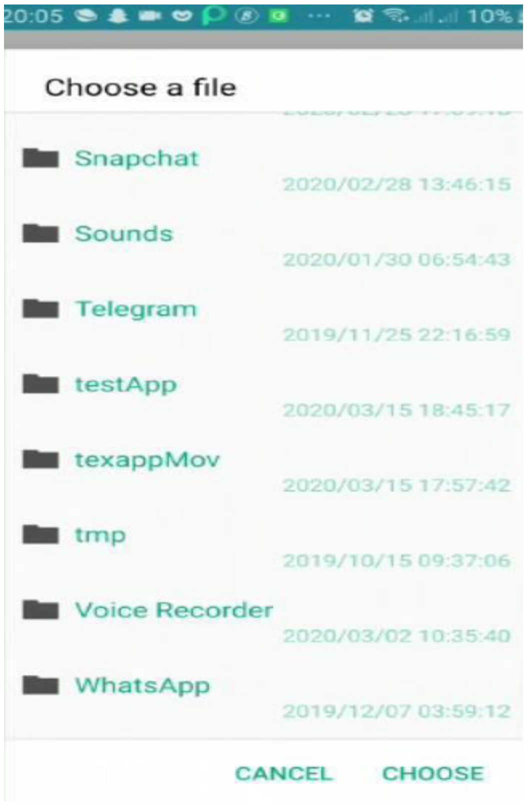
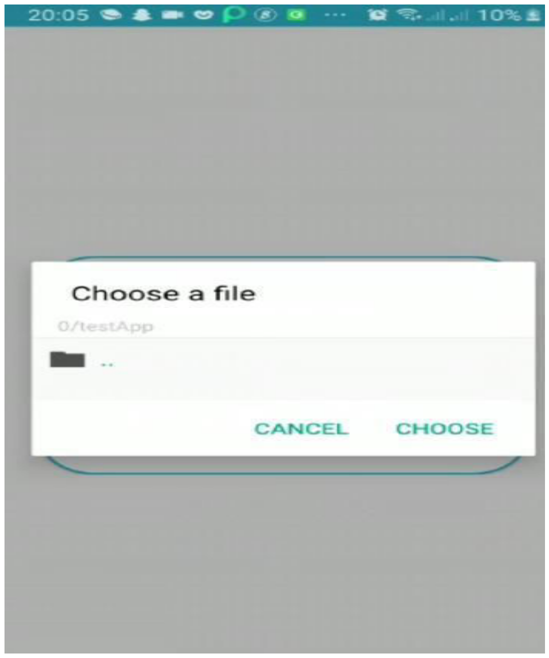


Figure 4.1 ContentAnalyzer icon figure 4.2 The scan and history phase


20:05 10%

Choose a file

Snapchat	2020/02/28 13:46:15
Sounds	2020/01/30 06:54:43
Telegram	2019/11/25 22:16:59
testApp	2020/03/15 18:45:17
texappMov	2020/03/15 17:57:42
tmp	2019/10/15 09:37:06
Voice Recorder	2020/03/02 10:35:40
WhatsApp	2019/12/07 03:59:12

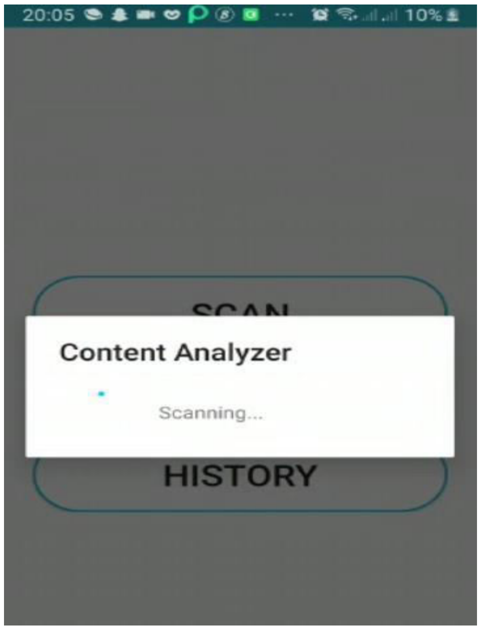

20:05 10%

Choose a file

0/testApp

..

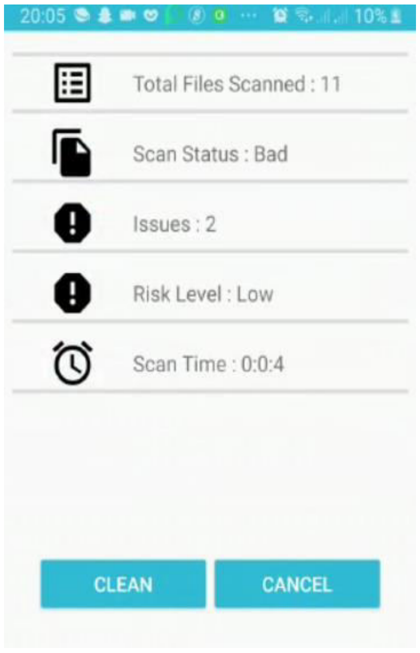
Figure 4.3 Total number of file on user Smartphone Figure 4.4 Selection of File on user Smart phone



Content Analyzer

Scanning...

Figure 4.5 Scanning Analysis Phase



Total Files Scanned : 11

Scan Status : Bad

Issues : 2

Risk Level : Low

Scan Time : 0:0:4

Figure 4.6 The History of Total action performed by the ContentAnayzer

5.0 Conclusion

In this paper, we presented the architectural design, the sequence and implementation for information leakage detection and prevention on android-based devices. Smart phones can contain private and confidential data, when allowed for accessing corporate applications and malicious apps can steal such sensitive data or can track users without their consent. Consequently, the approach used in this study (static and dynamic), it was found that static analysis techniques is one of the best suited for analysis on Android Market site, and dynamic analysis for real time tracking on Android phone. The static and dynamic analysis combinations were used to detect illegal data leakage which proved to be effective. The Content analyzer developed would be seen as a good tool. Although, it has its drawbacks which defer in level of severity but they all generally affect the system's overall performance. The first of these drawbacks is that it requires the prior installation of the application on the target device, this is a direct implication of the use of the tool. The computational time also is significantly larger when there are more activities to be scanned, that is, scanning the larger number installed applications will reducing the suitability of this system to efficiently scan large number of installed applications and we believed that this implementation will minimized false positives and in turn lead to increase in code coverage to detect the maximum number of data leaks.

6.0 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. andKruegel, 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.