ABUAD INTERNATIONAL JOURNAL OF NATURAL AND APPLIED SCIENCES
ISSN: 2955-1021
AIJNAS 2022, Volume 2, Issue 1, pp 12-28
https://doi.org/10.53982/aijnas.2022.0201.02-j
Copyright ©2022
https://journals.abuad.edu.ng/index.php/aijnas

Review of Works Content Analyzer for Information Leakage Detection and Prevention in Android Smart Devices

Okebule T[1*], Adeyemo. O.A[2], Olatunji K.A[3] and Awe. A.S[4]
Department of Mathematical and Physical Sciences.
Afe Babalola University,Ado-Ekiti, Ekiti State, Nigeria.
*Corresponding author: okebulet@abuad.edu.ng

Abstracts
The advent of android operating systems introduced tools to keep track of users' information activities and prevent information leakage which bridged the trust between application developers and consumers. A review of related 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 presents a literature review of works on content Analyzers for information leakage detection and prevention on android-based devices. The review will help to combine different concept to minimize false positives that will in turn lead to increase in code coverage towards detecting the maximum number of data leaks.
Keywords: *Android, Content Analyzer, Static Analysis, Dynamic Analysis, Information leakage, Information leakage detection, Information leakage Prevention.*

INTRODUCTION

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, the later has greatly helped in achieving a balance between personal and corporate life.

Programs that steal information also known as malicious software affects the user's mobile devices by exploiting the vulnerabilities. The types of malware that are most commonly used are viruses, worms, Trojans, among others. There is also another widespread use of malware which allows malware authors 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 (Willems *et al.*, 2007).

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. 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 (Salton *et al.*, 1988).

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 policy holder 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 counter measures 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.

Information Leakage Detection has been described as a situation whereby malicious circuits are hidden on-chip by 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. 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 (Klieber *et al.*, 2014).

Related Works on Content Analyzer for Information Leakage Detection and Prevention in Android Smart Devices.

The paragraphs below are the gathered and reviews of related work for a critical assessment on Content Analyzers for Information Leakage Detection and Prevention on Smartphones Devices.

TaintDroid

Enck *et al.* (2014) developed TaintDroid with dynamic approach (also called taint tracking) alerts information leaks inside an Android application via dynamic taint tracking. AppIntent redefines the privacy leakage as user-unintended sensitive data transmission and designs a new technique, event-space constraint symbolic execution, to distinguish intended and unintended transmission. However the tools could neither analyze other kinds of undesirable behaviors such as stealthily sending SMS, nor examine the internal logic of sensitive behaviors.

DroidScope

Yan *et al.* (2012) 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.

DroidSafe

Steven *et al.* (2014) 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.

Woodpecker

Kim *et al.* (2012) implemented Woodpecker tool that was developed for the Android platform and thus needs to overcome platform-level peculiarities for the control-flow construction and data flow analysis. Most importantly, Woodpecker has a different goal in uncovering unsafe exposure of dangerous capability uses, including both explicit and implicit ones. Signing key, is installed to grant the additional permission. However, due to the fact that the system cannot rule out the possibility of a colluding application being installed at a later time, its mere absence does not indicate such an implicit leak is 'safe' and may not occur later. Meanwhile, the system limit the attacker's scope by assuming the Android framework (including the Operating System kernel) is trusted. Also, the system assumes that the signing key to the system image has not been leaked to the attacker. Given these constraints, a malicious application will not be able to directly access the high privilege Applications.

MapReduce Algorithm

Liu *et al.* (2016) designed MapReduce algorithm for privacy-preserving detection of sensitive data exposure with the capability to arbitrarily scale as well as use of public resources for the process. This solution uses the MapReduce framework for detecting exposed sensitive content, because it has the ability to arbitrarily scale and utilize public resources for the task, such as Amazon EC2. The algorithms support a useful privacy-preserving data transformation. This transformation enables the privacy-preserving technique to minimize the exposure of sensitive data during the detection. This prototype implemented with the Hadoop system achieves 225 Mbps analysis throughput with 24 nodes. This transformation supports the secure outsourcing of the data leak detection to untrusted MapReduce and cloud providers. However, the system is not developed for intentional information exfiltration, which typically uses strong encryption.

SCANDAL

Jinyung *et al.* (2018) develop SCANDAL, a Static Analyzer for Detecting Privacy Leaks in android applications. Static analyzer SCANDAL is a technique for providing a formal sound 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 is considered for future works.

Aquifer

Nadkarni *et al.* (2013) developed a tool for preventing accidental data disclosure in modern operating systems. Aquifer is 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. The limitation of developed tool is that, malicious applications are not taken into considerations.

CopperDroid

Aristide et al. (2014) 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 sixty percentages 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 it was not combined with Binder information and automatic (complex) Android objects reconstruction.

RiskRanker

Michael *et al.* (2011) designed a RiskRanker to automatically collect data such as: the location of the secondary application; background dynamic code loading and related execution path(s); programmed access to internal directories; use of encryption and decryption methods; and native code execution and JNI accesses. The potential security risks into corresponding detection modules of two orders of complexity: the first-order modules handle non obfuscated applications by evaluating the risks in a straightforward manner; the second-order modules capture certain behaviors (for example., encryption and dynamic code loading) that are in themselves not of concern, but that in conjunction with others may form malicious patterns and be instrumental to detect stealthy malware. These analysis modules ultimately produce output that includes a severity rating and related evidence to verify the behavioral pattern in each reported application. This output is then sorted by severity to produce a prioritized list of suspicious applications that merit further analysis. RiskRanker reports, is a medium-risk for the user; which could result in the user being charged for money sneakily or upload undeniably private information to a remote server. For example android has a group of permissions named, which is defined as permissions that can be used to make the developer spend money without their direct invement. If abused, these features can be used to construct malware such as SMS Trojans, which send text messages to premium phone numbers that result in charges being placed on the user's phone bill. Such malware is popular due to the direct return it provides to malware authors, but these same features do have legitimate uses, typically in the form of instant-messaging, reminder, or social-networking applications.

Labyrinth

Shivakumara *et al.* (2019) designed an algorithm for dynamic mechanism of data leakage detection and prevention that performed an enhanced form of value similarity analysis to detect data leakage even when sensitive data (such as a password) is encoded or hashed. Labyrinth supports both Android and iOS the development of automation solutions in which the systems would be able to learn itself for its self-defensive capabilities and the human invement in the defense mechanism will be kept to minimum. However, this approach focus on area of detection of data leakage and prevention algorithms, tools and technologies supported.

Rapid Screening of Transformed Data Leaks

Xiaokui *et al.* (2015) designed rapid screening of transformed data leaks with efficient algorithms and parallel computing with two new algorithms for detecting long and transformed data leaks. The system achieved a high detection accuracy in recognizing transformed leaks compared to the state-of-the-art inspection methods. It parallelize its prototype on graphics processing unit and demonstrate the strong scalability of their detection solution required by a sizable organization. This technique has high level of precision in finding transformed information leaks compared with the state-of-the-art set intersection technique. However, it is time consuming process.

POSTER

Shweta *et al.* (2017) proposed a technique called POSTER for detecting Inter-App Information Leakage Paths. The proposed technique was based on verification method that used efficiently to check all the possible paths generated due to inter-app communication and to verify if the paths are admissible on the requirements of the safe state (no collusion). The paper revealed that, only intents have been explored as a means of inter-app communication mechanism. Based on this, the proposed collusion checking property detected presence/absence of collusion. This method provides a formal representation of the extracted information. This step helps in a compact representation of relevant information that can be given to model-checking tool. However, the malicious app developers generate the leakage path across multiple apps. Hence it is challenging to detect such leakage path.

Attire

Hoyle *et al.* (2013) developed an Attire for Conveying information exposure through avatar apparel. Attire system was built on an app called Attire to convey real-time information exposure in a lightweight and unobtrusive manner. Attire was based on their initial exploration of using an avatar for conveying exposure. As found in the background of computer desktops and smartphones used for this purpose, Attire presents the avatar as desktop wallpaper. Presently Attire handles a user's current location as the piece of personal information being accessed by others. Attire detects the user's location and accordingly situates his or her avatar in one of four typical locations: home, work, school and a place of social interactions such as a restaurant or bar. Each location is associated with contextually appropriate default attire. However, the user is rarely informed whether, when and by whom information was actually accessed within these specified parameters for permitted access. For example, knowing that information was accessed repeatedly within a short time-frame could be useful for making judgments such as urgency or stalking.

MR-Droid

Fang *et al.* (2017) presented MR-Droid: A Scalable and Prioritized Analysis of Inter-App Communication for accurate and scalable Inter component communication risk detection. The goal is to empirically evaluate ICC(Inter component communication) risks based on an app's inter-connections with other real-world apps and detect high-risk pairs by constructing a large-scale ICC graph, where each node is an app component and the edge represents the corresponding inter-app ICCs. The MapReduce based approach is divided into two broad steps. MR-Droid identifies ICC nodes (both sources and sinks) and group inter-app ICCs that belong to an app pair using MapReduce. In the second step, risk assessment module which detects the presence of risk and assigns ranking to the detected risk. Prioritizing risks helps to reduce false alarms. MapReduce based framework to scale up compositional app analysis, detect inter-app communication threats specifically intent on hijacking, intent spoofing and collusion. They also prioritized the identified ICC risks, based on the communication context of apps. However, this approach can handle intent based ICC communications only. Therefore, security risks posed by other inter-app channels like content providers, shared preferences, among others, cannot be detected.

Detecting Inter-App Information Leakage Paths

Klieber *et al.* (2014) presented a model for detecting Inter-App Information Leakage Paths. The approach is divided into four broad steps. The first step is to extract information related to ICC sources, sinks and, intent based communication channels. The second step is dataflow analysis to map sensitive information provided. The third step, PROMELA model is generated for each app. In the fourth step, model checking is done using the generated models and collusion detection property. The approach presented a model-checking based approach for inter-app collusion detection. The authors presented compositional app analysis to identify set of conspiring apps inved in the collusion.

Approach Towards Automated Android App Collusion Detection

Asavoae *et al.* (2016) presented an approach towards Automated Android App Collusion Detection. This is a statistical approach consisting of defining probabilistic model, training of the model for 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 and this approach did not address the issue of scalability.

User-Intention Based Program Analysis

Elish *et al.* (2012) proposed a User-Intention Based Program Analysis for Android Security. This method was basically on ICC Map, which is a hash map data structure. It stores ICC entry and exit points that can be extracted by scanning bytecode of source and target apps respectively. It is used to statically characterize the inter-app ICC channels among the Android apps. They defined collusion between apps as some set of actions executed by the apps that can lead to a threat. The approach statically identifies the predicted risk level associated with the inter-app ICC calls, but it does not confirm the existence of the collusion and has difficulty in performing the analysis on programs that employ obfuscation techniques, dynamic code loading, or use of reflection.

IccTA

Li *et al.* (2015) developed IccTA for detecting inter component privacy leaks in android apps security. IccTA method proposed APK Combiner to disassembled every app in order to obtain manifest and small files using android apk tool as reverse engineering tool. After that all files corresponding to different apps are combined together into a single directory and conflicts are resolved. IccTA is a static taint analyzer to detect privacy leaks between components in Android apps. It claims to improve its precision of analysis by propagating context-aware information. IccTA cannot analyze apps of big size as it requires too much memory consumptions and system often gets hanged. It cannot detect leak through multi-threading. It assumes the execution of threads in arbitrary but sequential order.

Permissionflow

Sbirlea *et al.* (2013). Designed an automatic detection of inter-application permission leaks in android applications for permissionflow. Permissionflow consists of three major modules, i.e. permission mapper, rule generator and decision maker. The approach consists of identifying APIs whose execution leads to permission-checking. Then another module, rule generator will define rules for tainting, then another open-source tool named Andromeda to identify flows and components. Decision maker will allow or disallow the flow based on permissions. Permissionflow is a single-app static analysis that handles attacks related to obtaining unauthorized access to permission protected information. It focuses on three types of attacks viz. permission collusion, confused deputy and Intent spoofing. PermissionFlow uses taint analysis to capture the flow of permissions. Permissionflow records a large number of false positives due to the checking of redundant permissions and data dependent checks. Permissionflow does not handle native code permissions.

Amandroid

Wei *et al.* (2014) designed Amandroid; a precise and general inter component data flow analysis framework for security vetting of android apps. Amandroid proceeds 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. Amandroid builds an inter-component data flow graph (IDFG) of the whole app. Amandroid is a static analysis tool that has the capability of calculating all objects' points-to information in a both flow and context-sensitive way. This tool detects whether there is any information leakage from a sensitive source to a critical sink by providing an abstraction of the app's behavior. Amandroid can be used for a number of useful security analysis as data leak detection, data injection detection and detection misuse of an API. Amandroid does not handle concurrency and reflections. An app may have multiple components and then may run concurrently and when multiple components interleave, this may induce some security issues. Amandroid has limited capability to handle exceptions. Amandroid may not detect an exception, when an app has a security issue where the core of an exception handler plays a role.

DidFail

Klieber *et al.* (2014) designed DidFail : Android Taint Flow Analysis for App Sets. DidFail has modified FlowDroid by adding few intent method calls as sources (onActivityResult ()) and sinks (setResult ()). It has also added code to analyze put Extra call for the intents that are uniquely identified by ID in Transform APK step. FlowDroid accepts transformed APK as the input and conducts taint analysis. It provides flows within the components of an app as the output. DidFail conducts static taint analysis of Android apps augmenting FlowDroid and Epicc tools to detect intra-component and inter component information flow in a set of apps. It performs analysis in two phases where the first phase determines information flow within the app and second phase determines flow across the apps. However, DidFail cannot detect flow of information when static fields are used as a source or sink for intents i.e. it misses the flow if an intent reads information from static field and does not handle native calls and reflection.

ComDroid

Chin *et al.* (2011) developed ComDroid. This Analyzing Inter-Application Communication in Android, which considered two types of analysis: Intent analysis and Component analysis. In Intent analysis, ComDroid statically analyzes method invocation to a depth of one method call. It checks whether the intent has been made explicit; whether the intent has an action; whether the intent has any flags set; and whether the intent has any extra data. In Component analysis, ComDroid examines application's manifest file to get components and translates dalvik instructions to get information about each component. ComDroid treats activities and their aliases as separate components because an alias field can increase the exposure surface of the component. It generates a warning about a potential intent spoofing attack, when it detects that a public component is protected with no permission or a weak permission. ComDroid is a tool that detects application communication vulnerabilities and could be used by developers and reviewers to analyze their own applications before release. The main purpose of this tool comes from the fact that Android's message passing system can become an attack if used incorrectly (personal data loss, information leakage, phishing, etc.) These vulnerabilities stem mainly from the fact that Intents can be used for both intra and inter application communication. However, in the event of verification of the existence of attacks: ComDroid issues warnings and not verify the existence of attacks. For instance, some components are intentionally made public for the purpose of inter-application collaboration. It is not possible to infer the developer's intention when making a component public. It is the role of the developer to verify the veracity of the warnings.

Techniques for Detecting Complex Data-Leak Patterns

Xiaokui *et al.* (2015) designed fast detection of transformed data Leaks which utilized sequence alignment techniques for detecting complex data-leak patterns. The algorithm was designed for detecting long and in exact sensitive data patterns. This detection is paired with a comparable sampling algorithm, which allows one to compare the similarity of two separately sampled sequences. The solution to the detection of transformed data leaks is a sequence alignment algorithm, executed on the sampled sensitive data sequence and the sampled content being inspected. The alignment produces scores indicating the amount of sensitive data contained in the content and this alignment based solution measures the order of n-grams. It also handles arbitrary variations of patterns without an explicit specification of all possible variation patterns. Experiments show that their alignment method substantially outperforms the set intersection method in terms of detection accuracy in a multitude of transformed data leak scenarios. This system was able to achieve good detection accuracy in recognizing transformed leaks. However, detecting the exposure of sensitive information is challenging due to data transformation in the content. Transformations (such as insertion, deletion) result in highly unpredictable leak patterns.

Intellidroid

Wong *et al.* (2016) designed Intellidroid, which is a targeted input generator for the dynamic analysis of android malware. IntelliDroid acts in 6 steps viz. Specifying target APIs, Identifying paths to target APIs, Extracting call path constraints, Extracting event chains, Determining run-time constraints and Input-injection to trigger call paths. IntelliDroid is a generic tool that generates input specific for a dynamic analysis tool to perform analysis more precisely by reducing false positives. Instead of static or dynamic analysis, it performed targeted analysis. It is achieved by preliminarily doing background study about the dynamic analysis tool and static analysis of the application given as input to the dynamic analysis tool. It helps in triggering target APIs and consecutively leads to more efficient and effective dynamic analysis. IntelliDroid is not capable of creating inputs for encrypted and hashed functions. The extracted constraints are sometimes very complex such as trigonometric functions. It cannot be resolved by constraint solver. Currently, human intervention is required to solve such constraints.

Intent Droid

Hay *et al.* (2015) designed IntentDroid, it is for dynamic detection of inter-application communication vulnerabilities in android. IntentDroid tests the applications in three phases viz: Instrumentation, Testing and Reporting. In the instrumentation phase, the app under analysis is instrumented to store library calls and access to user-supplied data. In the testing phase, to detect whether a vulnerability exists in the app or not based on IntentDroid has created attack scenarios. In the Reporting phase, IntentDroid reports the number of vulnerabilities present in an app after implementing all the possible attack scenarios on the app. IntentDroid is a framework that dynamically examines Android apps for IAC (Inter Application Communication). It created attack scenario for 8 vulnerabilities viz. Cross-Site Scripting, SQL Injection, Unsafe Reflections, UI (User-Interface) Spoofing, Fragment Injection, Java Crashing, Native Memory corruption and File Manipulation. It analyzes Activity component of apps by implementing attack scenarios in a way to obtain effective path coverage with minimum overhead. However, IntentDroid does not test Services, Broadcast Receivers and Content Providers for IAC vulnerabilities and does not consider multi-app attack.

FlaskDroid

Bugiel *et al.* (2013) developed FlaskDroid for Flexible and Fine-Grained Mandatory Access Control on Android on Di- verse Security and Privacy Policies. FlaskDroid plants various Object Managers at middleware and kernel layer that are responsible for assigning security context to objects. Related policies are managed by security servers deployed at different layers. The object manager makes access control decisions by using security servers at their respective layer. Also the deployed policies at both the layers are synchronized meaning change of policy in one layers, automatically reflect in another layer. FlaskDroid is policy-driven tool that provides security for kernel resources (like files, IPC, etc.) as well as middleware resources (like Intents, Content Providers, etc.). The security enforcement is through providing mandatory access control on both middleware and kernel layers of Android simultaneously. They extended Android's middleware layer with type enforcement and present a new policy language to capture the semantics of this layer. However, many false alarms while detecting confused deputy and collusion attacks. As FlaskDroid relies on application inputs/outputs and does not consider the information flow within apps. Access control rules are human user trail based.

XManDroid

Bugiel *et al.* (2011) presented XManDroid basically for a New Android Eution to Mitigate Privilege Escalation Attacks. XManDroid consists of three approaches (1) Application Installer. This is responsible for installation and uninstallation of applications. (2) System Policy Installer. This is responsible for the installation of explicitly defined list of system policies in the Android middleware. (3) Runtime Monitor. This is responsible for enforcing mandatory access control in Android like permissions checked at the interface, decisions taken whether to allow an ICC or not based on the information about installed apps and their communication. This approach is used for defining rules in system policies. XmanDroid (eXtended Monitoring on Android) is a dynamic framework that extends the monitoring mechanism of Android to detect and prevent application-level privilege escalation attacks. It is based on runtime system-centric policies. Two types of application-level privilege escalation attacks are handled by XmanDroid viz Confused Deputy attacks and Colluding attacks. Single app analysis is missing, that is, XManDroid cannot detect malicious app, as applications within a single sandbox have equal privileges and cannot perform privilege escalation.

AppAudi

Mingyuan *et al.* (2015) proposed AppAudi for Effective Real-time Android Application Auditing. The mechanism is in three-fold, that is (1) A propose approximated execution, a novel dynamic analysis that can execute part of a program while performing customized checks on its program state at each step. (2) an Android app auditing tool that can check apps effectively and efficiently. AppAudit embodies an API analysis to select suspicious functions and then relies on the approximated executor to prune false positives. (3) AppAudit to examine more than 400 free Android apps collected from various markets. This tool successfully identifies 30 data leaks. First, app market operators require automatic tools to detect and remove data-leaking apps. Second, app developers need to perform self-check before publishing apps. Third, mobile users expect to know if an app is leaking data before installation. AppAudit serves as an effective tool to identify dataleaking apps and provides implications to design promising runtime techniques against data leaks. However, AppAudit also uncovers 30 data leaks in real apps but only identifying data leak by estimation.

Detection and Analysis of Malware for Smart Devices

Suarez *et al.* (2014) presented Eution, Detection and Analysis of Malware for Smart Devices. This is an article that examines the problem of malware in smart devices and recent progress made in detection techniques. It first presents a detailed analysis on how malware has eved over the last years for the most popular platforms. We identify exhibited behaviors, pursued goals, infection and distribution strategies, etc. and provide numerous examples through case studies of the most relevant specimens. The Analysis is strongly biased towards smartphones, since they currently are the most extended class of smart devices and the platform of choice for malware developers and security researchers. This can help to better understand the problem and to improve upon current defense techniques. This taxonomy is only complemented with additional elements, such as where the monitoring and analysis tasks takes place, or the specific detection technique used.

Android Application Security

William *et. al.* (2011) presented A Study of Android Application Security. These findings of exposure of phone identifiers and location are consistent with previous studies. Their analysis framework allows us to observe not only the existence of dangerous functionality, but also how it occurs within the context of the application. The findings of exposure of phone identifiers and location are consistent with previous studies. Their analysis framework gives room for observations not only the existence of dangerous functionality, but also how it occurs within the context of the application. The studied applications were selected with a bias towards popularity, the program analysis tool cannot compute data and control flows for IPC between components; and source code recovery failures interrupt data and control flows. Missing data and control flows may lead to false negatives. It did not find evidence of malware or exploitable vulnerabilities in the studied applications.

DREBIN

Daniel *et al.* (2014) developed DREBIN. This is an effective and explainable detection of android malware in pocket. The method employs a broad static analysis that extracts feature sets from different sources and analyzes these in an expressive vector space. This consists of three modules as follows: a) Broad static analysis. In the first step, DREBIN statically inspects a given Android application and extracts different feature sets from the application's manifest and dexcode. (b) Embedding in vector space. The extracted feature sets are then mapped to a joint vector space, where patterns and combinations of the features can be analyzed geometrically. (c) Learning-based detection. The embedding of the feature sets enables us to identify malware using efficient techniques of machine learning, such as linear Support Vector Machines. (d) In the last step, features contributing to the detection of a malicious application are identified and presented to the user for explaining the detection process. DREBIN cannot generally prohibit infections with malicious applications, as it builds on concepts of static analysis and lacks dynamic inspection.

MockDroid

Alastair *et al.* (2011) designed MockDroid for trading privacy for application functionality on smartphones. MockDroid prototype is based on Android 2.2.1 and runs on the Google/HTC Nexus One handset. The choice of error depends on the API call under consideration. This approach allows users to revoke access to particular resources at run-time, encouraging users to consider the tradeoff between functionality and the disclosure of personal information whilst they use an application. However, this method has not been tested on other version of handsets.

PiOS

Manuel *et al.* (2017) developed PiOS for Detecting Privacy Leaks in iOS Applications. PiOS uses static analysis to check applications for the presence of code paths where an application first accesses sensitive information and subsequently transmits this information over the network. Since no source code is available, PiOS has to perform its analysis directly on the binaries. PiOS employs backward slicing to calculate the contents of these registers at every call site to the objmsgSend function in an application binary. PiOS detect privacy leaks in applications written for iOS. PiOS is able to determine the type of the receiver (R0) and the value of the selector (R1). It annotates the call site with the specific class and method that will be invoked when the program is executed. It is not straightforward to extract the application's decryption key from the device (and the operating system's secure key chain). Furthermore, to use these keys, one would also have to implement the proper decryption routines. PiOS does not track (the addresses of) individual instances of classes allocated during runtime. Also, statically determining the receiver and selector for every call to the objcmsgSend function is not always possible.

DroidRay

Min *et al.* (2014) proposed DroidRay for a Security Evaluation System for Customized Android Firmwares. DroidRay uses both static and dynamic methodologies to analyze the system security of the Android firmware. Specifically, three forms of security analysis were carried out: (a) system signature vulnerability detection (b) network security analysis and (c) privilege escalation Vulnerability detection system levels. DroidRay is a security evaluation system for customized Android firmware. The system uses Android firmwares as the input, then the system analyzes both Android firmwares as well as pre-installed applications. After the analysis, DroidRay outputs the analysis report of these firmwares and pre-installed applications. DroidRay is an effective tool to combat this new form of malware spreading. The system uses both static and dynamic analyses to evaluate the firmware security on application level and system level. A comprehensive study on 24,009 pre-installed applications and 250 Android firmware systems were carried out and discover compromised firmwares can contaminate the system and inject new malware into devices. However, DroidRay only retains the applications which have dangerous permissions (e.g., sending SMS message) or silent installation behavior.

Dissecting Android Malware for Characterization and Eution

Yajin *et al.* (2012) proposed Dissecting Android Malware for Characterization and Eution. It focuses on the evaluation of Android platform and characterizes the existing Android malware. The paper presents a systematic characterization of existing Android malware. The characterization is made possible with our more than one-year effort in collecting 1260 Android malware samples in 49 different families, which covers the majority of existing Android malware, ranging from its debut in August 2010 to recent ones in October 2011. The challenges lie in the large ume of new apps created on a daily basis as well as the accuracy needed for repackaging detection and so there is no clear solution. The defense capability is largely constrained by the limited understanding of these emerging mobile malware and the lack of timely access to related samples.

MOSES

Yury *et al.* (2014) designed MOSES for Supporting and Enforcing Security Profiles on Smart Phones. A policy-based framework for enforcing software isolation of applications and data on the Android platform using different modules: user registration and authentication, security profiles create, check error process and MOSES manager process. MOSES is the dynamic tool used for switching from one security profile to another, MOSES implements soft virtualization through controlled software isolation. Each security profile (SP) can be associated to one or more contexts that determine when the profiles become active. Both contexts and profiles can be easily and dynamically specified by end users. MOSES provides a Graphical User Interface for this purpose. Switching between security profiles can require user interaction or be automatic, efficient and transparent to the user. Leakage of sensitive information can be greatly reduced. Different OS can run separately at the same time, However, this approach fails to provide users visibility into how third-party applications have access to information.

AndroSAT

Saurabh *et al.* (2014) developed AndroSAT ford Security Analysis Tool for Android Applications. This is a framework that allows one to efficiently experiment with different security aspects of Android Apps through the integration of (i) a static analysis module that scans Android Apps for malicious patterns. The static analysis process inves several steps such as n-gram analysis of dexfiles, de-compilation of the App, pattern search and analysis of the AndroidManifest file; (ii) a dynamic analysis sandbox that executes Android Apps in a controlled virtual environment, which logs low-level interactions with the operating system. The effectiveness of the developed framework is confirmed by testing it on popular Apps collected from F-Droid and malware samples obtained from a third party and the Android Malware Genome Project dataset. AndroSAT, is that the developed sandbox allows not only for observing and recording of relevant activities performed by the apps (e.g., data sent or received over the network, data read from or written to files and sent text messages) but also manipulating, as well as instrumenting the Android emulator. These modifications were made to the Android emulator in order to evade simple detection techniques used by malware writers. The implementation prototype does not allow for more useful add-ons that can be used to provide further investigation of the security of Android applications.

AndRadar

Martina *et al.* (2014) designed AndRadar for 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 sixteen markets. Moreover, their evaluation shows that AndRadar makes harvesting marketplaces for known malicious or unwanted applications fast and convenient. In order to attract users and lure them into downloading their apps, malicious authors need an identifiable brand, e.g. by piggybacking on popular apps from the official market. Thus, if malicious authors decide to evade the discovery of their apps by AndRadar, this would invariably lower their visibility to users.

AndroZoo

Kevin *et al.* (2016) developed AndroZoo for Collecting Millions of Android Apps for the Research Community. This is a dedicated web crawler using the scrappy framework. Every candidate app which is available for free runs through a processing pipeline that: 1. ensures this app has not already been downloaded; 2. Downloads the file; 3. computes its SHA256 checksum; 4. Archives the file. AndroZoo presented dataset of millions of Android apps collected from various data sources. They make this dataset readily available to the community to contribute to more generalizable, reliable and reproducible studies based on a large-scale, representative and up-to-date samples. Indeed, the crawlers are managed as a low-priority research project rather than as a mission-critical production system. Collection was regularly interrupted for days, weeks, or even a few months, for issues such as lack of storage space or more generally, limited workforce to invest. In the course of the review, it was found that several market owners took various steps in order to prevent their market from being automatically mined. Thus, for such markets, it cannot guarantee that their whole content has been retrieved.

AsDroid

Jianjun *et al.* (2014) designed AsDroid for Detecting Stealthy Behaviors in Android Applications by User Interface and Program Behavior Contradiction. This is an implementation prototype called AsDroid (Anti-Stealth Droid). A method to transform the DEX file of an app to a JAR file with dex2jar and then use WALA as the analysis engine. The implementation is mainly on top of WALA. Also, collected apps from three different sources. This aim to detect those with the following stealthy behavior: SMS sends, phone calls, HTTP connections and component installations. Hence, it only focus on those having the permissions for such behaviors. The implementation of prototype called AsDroid (Anti-Stealth Droid) is a pool of 182 apps that have the permissions to perform the malicious operations of interest was collected. AsDroid reports that 113 of them have stealthy behaviors, with 28 false positives and 11 false negatives. Currently, to avoid false positives, AsDroid relies on certain rules in detecting intent correlation and avoids reporting some intents incompatible with UI if their correlated intents are compatible. This seems to be working engine given that Android malwares are still in their early stage. AsDroid currently cannot reason about correlations through external resources, leading to false positives. AsDroid cannot analyze dynamically generated text associated with a UI component.

Taint Analysis

Christian *et al.* (2013) developed an Highly Precise Taint Analysis for Android applications. FlowDroid method, a novel and highly precise static taint-analysis tool for Android applications. Showed that many existing approaches do not adequately model Android-specific challenges like the application lifecycle or callback methods, leading to either missed leaks or false positives. FlowDroid can thus generate a main method in which every order of individual component lifecycles and callbacks is possible and it does not need to simulate all possible paths. FlowDroid also has a higher precision resulting in less false positives. FlowDroid extends the Soot framework which provides important prerequisites for a precise analysis, in particular a very accurate call graph. At the moment FlowDroid ignores reflective calls, which is unsound. While specialized static string analyses can be used to simulate reflection to some extent. Past research has found such analyses to be incomplete, as reflective call targets are often determined by external configuration files.

DroidSwan

Babu *et al.* (2015) designed DroidSwan for detecting malicious android applications based on static feature analysis. This an approach which extracts various features from Android Application Package file (APK) using static analysis and subsequently classifies using machine learning techniques. The analysis is carried out using various machine learning algorithms with both weighted and non-weighted approaches. DroidSwan detects possible collusion attack. Rather than looking for over privileged applications, under privileged applications were detected, that is, the application declaring less permissions than what it actually required. The under privileged application then gets required privileges at runtime with the help of another application. The drawback with this approach is the high false positive rate. The reason for high false positive rate is that many developers declare majority of the permissions available irrespective of their usage by the application.

Crowdroid

Iker *et al.* (2011) proposed Crowdroid. This is a Behavior-Based Malware Detection System for Android. The method shown to be an effective means of isolating the malware and alerting the users of a downloaded malware. This shows the potential for avoiding the spreading of a detected malware to a larger community. In collaboration with the Android users community, it will be capable of distinguishing between benign and malicious applications of the same name and version, detecting anomalous behavior of known applications. There is challenge of convincing the Android user community to install the Crowdroid application. It need to manage the perception of loss of privacy when supporting research community with their behavior information, against the benefit of having access to up-to-date behavioral-based detected malware statistics.

ARP

Blagoj *et al* (2015) presented Real-World ARP Attacks and Packet Sniffing, Detection and Prevention on Windows and Android Devices, It uses WireShark filters and so, filtered the communication down to http POST methods which displayed us with the victim's login and password on a test page. The paper explains the purpose and the need of the Address Resolution Protocol (ARP) and different types of attacks that can occur due to its stateless nature. It exhibit a real world example of a Man-in-the-Middle (MitM) attack by sniffing http logins of a Windows PC and an Android device and then suggest methods of both detecting and preventing the attack. However, this approach is limited with methods and applications to detect and prevent ARP attacks.

Fraud Detection in Information Leakage

Poonam *et al.* (2012) presented Fraud Detection in Information Leakage with methods to handle such information leakage namely watermarking and Identifying Guilty agent using probability. The system then extracts the requested data from the main database and performs the addition of fake records according to the request. It then provides the data to the agent. If any of the agent leaks the data to some unauthorized vendor, then the vendor will try to establish a contact with the customers by sending them advertising mails. The job of the mail detection system is to monitor these incoming mails on the email addresses of the fake records continuously. If the system detects unauthorized mail crossing the threshold value, then it starts its process of probability calculation. The threshold value is the minimum number of mails which have to be detected to trigger the calculation. It will check the presence of these fake records in each of the agent and accordingly will evaluate the probability of each agent being guilty. The system basically deals with customer information like email-IDs. When customer data is leaked, the third party tries to create contact with these customers through mail. This mail detection system keeps a track of the fake email aids added at the time of data allocation and distribution to agent and when an unidentified mail having advertisement content arrives in the mailbox, the system informs the administrator about it. However, there are two major disadvantages of this algorithm: It inves some modification of data i.e. making the data less sensitive by altering attributes of the data. This alteration of data is called perturbation. However in some cases, it is important not to alter the original distributed data. For example, if an agent is doing the payroll, he must have the exact salary. We cannot modify the salary in this case. And second problem is that these watermarks can sometimes destroyed if the recipient is malicious.

Systematic Detection of Capability Leaks in Stock Android Smartphones

Michael *et al.* (2012) Systematic Detection of Capability Leaks in Stock Android Smartphones. It employs inter procedural data flow analysis techniques to systematically expose possible capability leaks where an untrusted app can obtain unauthorized access to sensitive data or privileged actions. The system distinguishes two different categories. Explicit capability leaks allow an app to successfully access certain permissions by exploiting some publicly-accessible interfaces or services without actually requesting these permissions by itself. Implicit capability leaks allow the same, but instead of exploiting some public interfaces or services, permit an app to acquire or inherit permissions from another app with the same signing key. The main benefit of performing this kind of analysis is that it models all data flow assignments, not just those relating to branch conditions. As a result, it can trace the provenance of any arguments to the dangerous method. With such information, we can characterize the severity of the capability leak. A capability leak that directly passes through arguments from the external caller is obviously worse than one that only allows invocation with constant values and this design can distinguish between the two. Given that path feasibility is undecidable, the design errs on the side of caution: it will not claim a feasible path is infeasible, but might claim the reverse is true. As a result, this argument information is valuable, as it can be used to generate a concrete test case that verifies the detected capability leak. However, since many pre-loaded apps have the corresponding permissions, the malicious app will have gained access to a high-privilege capability if it can cause one of these apps to invoke the desired API on its behalf. This approach does not know what kind of dangerous call lies at the end of the path beforehand. Allowing unrelated permission checks to mark whole paths as infeasible would therefore introduce false negatives.

DroidBox

Naresh *et al.* (2013) described a review On Data Leakage Detection and Presented a mechanism for proof of ownership based on the secure embedding of a robust imperceptible watermark in relational data. The watermarking of relational database as a constraint optimization problem were formulated and discuss efficient techniques to solve the optimization problem and handle the constraint by proposing data allocation strategies that improve the probability of identifying leakages. The goal of this approach is to detect when the distributor's sensitive data has been leaked by agents and if possible to identify the agent that leaked the data by considering applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made less sensitive before being handed to agents. However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image.

Detection and Avoidance of Data Leakage

Sandesh *et al.* (2012) presented Detection and Avoidance of Data Leakage. The aim was to detect the leakage of data while transmitting the data from server to client via various routers by identifying the guilty router and also avoiding the leakage. This approach study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, it study the following scenario: after giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a website, or may be obtained through a legal discovery process). At this point, the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if Rama is caught with a single cookie, he can argue that a friend gave him the cookie. But if Rama is caught with five cookies, it will be much harder for him to argue that his hands were not in the cookie jar. However, in many cases, there is a need to work with the agents that may not be trusted and it may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks.

OCR

Yeon-kyung *et al.* (2015) proposed Stealthy Information Leakage from Android Smartphone through Screenshot and OCR that utilized one attack path, screen bitmap memory, in order to propose a collection system that retrieves IMEI and IMSI information through screenshot image and extracts the information from the image by OCR (Optical Character Recognition) automatically. Furthermore, it found out that sans font showed very low recognition rate while serif and mono showed relatively high recognition rate. A screenshot activity from users was also hiding. Therefore, the proposed methods were used to leak any information without worry of detection by DroidBox, users, text-based packet inspections tools. However, also found out that sans font showed a very low recognition rate while serif and mono showed a relatively high recognition rate only with font size bigger than 12.

DroidData

Hani *et al.* (2017) developed DroidData: Tracking and Monitoring Data Transmission in the Android Operating System. DroidData, is a novel tool that uses both static and dynamic analysis to track and monitor data transmission in Android applications. This approach minimizes false positives and increases code coverage to catch the maximum number of data leaks. This novel method of combining two types of analyses, which allows DroidData to catch more data transmissions than existing works. A well-developed and easy-to-use user-interface that allows users to understand data transmissions and block applications that transmit their personal information inappropriately. More informative results about the transmissions than other tools, providing information such as the security of the data transmission. However, DroidData, like most other current analysis tools for Android, is unable to track implicit, or control, data flows. Some malicious code uses implicit flows to exploit security mechanisms and avoid detection, so this is something to protect against in the future.

Taintdroid's Functionality And Crowd-Sourcing Tool

Ferreira *et al.* (2015) presented Secrecy that uses TaintDroid's functionality and combines it with a crowd-sourcing tool where users can share information about security issues they experienced with an application and provide a rating for other users to consider before downloading the application. This has the disadvantage according to their own investigation showed that only five out of one hundred of their participants rated the app. rating an application is often considered inconvenient by users and is not always accurate, based on varying opinions and understandings of privacy by different users.

AppIntent

Yang *et al.* (2013) presented AppIntent that uses symbolic execution to determine the GUI manipulations that lead up to a data transmission to allow an expert to determine whether is intended by the user or not. This draws a distinction between true data leaks and transmissions that are necessary for application functionality. It pioneers a technique called Event-space Constraint Guided Symbolic Execution that identifies all possible execution paths using static analysis, then identifies critical events to the transmission using an event constraint graph. It then uses dynamic analysis to determine what UI manipulations led to the data transmission. While this tool is unique in seeking to determine whether data transmission is a privacy leak, it requires that a human analyst be presented with the results of the dynamic analysis to determine user intention and they note that it is probably impossible for that process to be completely automated.

Comnoid

Sunita *et al.* (2017) proposed Comnoid: Information Leakage Detection using Data Flow Analysis on Android Devices. Comnoid is based on FlowDroid open source tool. It takes Android application apk file as input and extracts Dex file, Manifest file and Xml layout files for processing. Comnoid tool was developed to perform Static Taint analysis with inter app analysis which will take Android application APK files as an input and produce a data leakage report. However, it suffers with the drawbacks which are common to that of other static analysis tools like it is not able to analyse the reflexive calls and dynamically loaded code.

RECOMMENDATION

For future work, more Scholarly literature can be carried out thoroughly to summarize their result in a form that will develop research questions of different types regarding the best method that can be advanced on and investigating on basis for designing a research on Content Analyzers for Information Leakage Detection and Prevention Android Based-Devices. We believed that if this related works is reviewed accordingly, the best method can be combined to be implemented and false positives will be minimized and in turn lead to increase in code coverage to detect the maximum number of data leaks. Furthermore, combining different techniques reviews could enhance the capacity for the developed contentAnalyser to detect and prevent more information leakages on smartphones than previously considered works.

CONCLUSION

In this paper, we presented the literature reviews with their different methods, strengths and weaknesses that will guide in the designing of Content Analyzer for Information Leakage Detection and Prevention on android-based devices. The extent of the implementation of these reviews 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

REFERENCES

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, 12(1):103-108.

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, 11(1): 627-638.

Agrawal R. and Srikant R. (2000). Privacy-preserving data mining, in Proceedings of the 2000 ACM SIGMOD International conference,:439-450.

Agrawal R., Gehrke J. and Gunopulos D. (1998) Automatic subspace clustering of high dimensional data for data mining applications, in Proceedings of the ACM SIGMOD International Conference on Management of Data: 94–105.

Alassi D. and Alhajj R. (2013) Effectiveness of template detection on noise reduction and websites summarization, Information Sciences, 219: 41–72.

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. 12:59.

Andrei, S. and Andrew, C. (2003). Language-based information-flow security. IEEE Journal of Selected Areas in Communication, 21(1): 21(1): 5–19.

Aristide, F., Kimberly, T., Salahuddin, J., Khan, A. and Lorenzo, C. (2014). CopperDroid: In Proceedings of the 2007 USENIX Annual Technical Conference. 233–246.

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. 7(1): 50-57.

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. Assessment, 5th International Conference DIMVA): 143– 163.

Azim T. and Neamtiu I. (2013). Targeted and depth-first exploration for systematic testing of Android apps, in Proceedings of the ACM SIGPLAN Conference on Object Oriented Programming Systems Languages & Applications, Indianapolis, Ind, USA : 641–660.

Babcock B., Datar M., Motwani R. and O'Callaghan L.,(2003) Maintaining variance and k-medians over data stream windows, in Proceedings of the Twenty second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS :234–243.

Backes, M., Kopf, B. and Rybalchenko, A. (2009). Automatic discovery and quantification of information leaks. In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy. IEEE Computer Society, Washington, DC, US :141–153.

Bayer, U., Moser, A., Krugel, C. and Kirda, E. (2006). Dynamic analysis of malicious code. Journal in Computer Virology 2(1):67–77.

Bhoraskar R., Han S., Jeon J. and Brahmastra (2014): driving apps to test the security of third-party components, in Proceedings of the 23rd USENIX Conference on Security Symposium, San Diego, Calif, USA: 1021–1036.

Bläsing T., Batyuk L., A.-D. Schmidt, S. A. Camtepe and S. Albayrak (2010). An Android Application Sandbox System for suspicious software detection, in Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE '10) IEEE, Lorraine, France: 55–62.

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. 11: 15–26.

Cacheda, F. and Vina, A. (2001). Experiencies retrieving information in the World Wide Web. In Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC 2001). IEEE Computer Society :72–79.

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.

Cavallaro, L., Saxena, P. and Sekar, R. (2008). On the limits of information flow techniques for malware analysis and containment. In Detection of Intrusions and Malware and Vulnerability.

Chang B. and Jeong Y. (2011). An efficient network attack visualization using security quad and cube, ETRI Journal, 33(5):770–779.

Chen K., Johnson H., D'Silva V. (2013). Contextual Policy Enforcement in Android Applications with Permission Event Graphs, in Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS '13) San Diego, Calif, USA.

Chen, H. and Wagner, D. (2002). MOPS: an infrastructure for examining security properties of software. In Proceedings of the 9th ACM conference on Computer and communications security.

Cohen W. W. (1996).Learning rules that classify e-mail, in Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access: 18–25.

Computer and Communications Security (CCS). 116–127.

Cui J., Zhang Y., Cai Z., Liu A. and Li Y. (2018). Securing display path for security-sensitive applications on mobile devices, Computers, Materials and Continua, 55(1): 17–35.

Dash M., Choi K., Scheuermann P. and Liu H. (2002). Feature selection for clustering - A filter solution, in Proceedings of the 2nd IEEE International Conference on Data Mining, ICDM 2(1):115–122.

DeBlasio J., Savage S., Voelker G. and Snoeren A. (2017). Tripwire: Inferring internet site compromise, in Proceedings of the IMC '17, pp 17: 1–14.

Deerwester S., Dumais T., Furnas. W., Landauer T. and Harshman R. (1990). Indexing by latent semantic analysis. Journal of the Association for Information Science and Technology, 41(6):391–407.

Egele, M. Scholte, T. Kirda, E. and Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools, ACM Computing Surveys, 44(2), Article 6, 42pp.

Enck W. Gilbert P. Chun BG. Cox LP. Jung J. McDaniel P. (2010). Taintdroid: An information-flow tracking system for real time privacy monitoring on smartphones. OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation. 10: 393–407.

Enck, W., Gilbert, P. and Han S. (2014). TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones, ACM Transactions on Computer Systems, 32(2): 5.

Feng, H., Giffin, J., Huang, Y., Jha, S., Lee, W. and Miller, B. (2004). Formalizing sensitivity in static analysis for intrusion detection. In IEEE Symposium on Security and Privacy. 194 – 208.

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: 22-26.

Fritz D., Bierma M., Gustafson E., Erickson J. and Choe Y. (2014). Andlantis: large-scale Android dynamic analysis, in Proceedings of the 3rd Workshop on Mobile Security Technologies (MoST '14) San Jose, Calif, USA.

Gilbert P., Chun G., Cox P. and Jung J. (2011). Vision: automated security validation of mobile apps at app markets, in Proceedings of the 2nd International Workshop on Mobile Cloud Computing and Services (MCS '11) ACM, Bethesda, Md, USA:21–26.

Goyal A., Bonchi F. and Lakshmanan S. (2012) On minimizing budget and time in influence propagation over social networks, Social Network Analysis and Mining,:1–14.

Guha S., Rastogi R. and Shim K. (1998) Cure: an efficient clustering algorithm for large databases, in Proceedings of 1998 ACM SIGMOD International Conference Management of Data,: 73–84.

Haritha, R. and Bhagavan, K. (2019). Anti-Reverse Engineering Techniques Employed by Malware: International Journal of Innovative Technology and Exploring Engineering (IJITEE) (8):2278-3075.

Hinneburg A. and Keim D. (1999) Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering, in Proceedings of the 25th VLDB Conference,: 506–517.

Huang X., Lu Y., Li D. and Ma M. (2018) A novel mechanism for fast detection of transformed data leakage, IEEE Access, 1: 1–11.

Hyde R., Angelov P. and MacKenzie A. (2017). Fully online clustering of eving data streams into arbitrarily shaped clusters, Information Sciences, 382-383.

Intrusions and Malware & Vulnerability Assessment (DIMVA). 17–36.

Islam M., Seera M. and Loo C. (2017). A robust incremental clustering-based facial feature tracking. Applied Soft Computing, 53:34–44.

Jarabek C., Barrera D. and Aycock J. (2012). ThinAV: truly lightweight mobile cloud-based anti-malware, In: Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12) ACM, Los Angeles, Calif, USA: 209–218.

Jiang F., Fu Y., Gupta B. (2018) Deep learning based multi-channel intelligent attack detection for data security. IEEE Transactions on Sustainable Computing. 99:1-9 DOI:10.1109/TSUSC.2018.2793284

Jin R., Si L., Hauptmann A. and Callan J. (2002). Language model for IR using collection information, in Proceedings of the 25th annual international ACM SIGIR conference,: 419-420.

Kalidindi S., Niezgoda S., Landi G., Vachhani S. and Fast T. (2010)A novel framework for building materials knowledge systems, Computers, Materials and Continua, 17(2):103–125.

Katz G., Elovici Y. and Shapira Coban B. (2014) A context based model for data leakage prevention, Information Sciences, 262:137–158.

Katz S. M. (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer, IEEE Transactions on Signal Processing, 35(3): 400-401.

Kim, H. C., Keromytis, A. D., Covington, M. and Sahita, R. (2009). Capturing information flow;

Systematic detection of capability leaks in stock Android smartphones: Proceedings of the 19th Annual Symposium on Network and Distributed System Security,: 23-34.

Kirda E. and Kruegel C. (2012) A survey on automated dynamic malware-analysis techniques and tools, ACM Computing Surveys, 44(2):6.

Li L. Bartel L. Bissyandé TF. Klein J. Traon YL. Arzt S. (2015) IccTA: detecting inter-component privacy leaks in Android apps ICSE '15 Proceedings of the 37th International Conference on Software Engineering, 1(5):280–291.

Liu C. (2010). An analytical method for computing the one-dimensional backward wave problem, Computers, Materials and Continua, 13(3):219–234.

McCallum A., Nigam K. and Ungar L. (2000) Efficient clustering of high-dimensional data sets with application to reference matching, in Proceedings of the KDD 2000, ACM, New York, NY, USA.:169–178.

Michael, I., Kim, D., Jeff, P., Limei G., Nguyen, N. and Martin, R. (2015). DroidSafe: Information-Flow Analysis of Android Applications in DroidSafe. (15): 8-11.

Mitra P., Murthy C. and Pal S. (2002) Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3): 301–312.

Nazar A., Seeger M. and Baier H. (2012) Rooting Android—extending the ADB by an auto-connecting WiFi-accessible service, in Information Security Technology for Applications, P. Laud, Ed., 7161 of Lecture Notes in Computer Science, Springer, Berlin, Germany.:189–204.

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). 54(11):263–278.

Oberheide J., Veeraraghavan K., Cooke E., Flinn J. and Jahanian F. (2008). Virtualized in-cloud security services for mobile devices, in Proceedings of the 1st Workshop on Virtualization in Mobile Computing, ACM, Breckenridge, Colo, USA: 31–3.

Octeau D. McDaniel P. Jha S. Bartel A. Bodden E. Klein J. (2013). Effective Inter-Component Communication Mapping in Android with Epicc: An Essential Step Towards Holistic Security Analysis. (8): 543–558.

Ordonez C. (2003)Clustering binary data streams with K-means, in Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03, pp 12–19.

Pacheco F., Cerrada M., Sánchez R., Cabrera D., Li C. and Valente J. (2017) Attribute clustering using rough set theory for feature selection in fault severity classification of rotating machinery. Expert Systems with Applications, 71:69–86.

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: 241–252.

Portokalidis G., Homburg P., Anagnostakis K. and Bos H. (2010). Paranoid Android: versatile protection for smartphones, in Proceedings of the Annual Computer Security Applications Conference (ACSAC '10), Austin, Tex, USA, pp 347–356.

Praba C. (2017) A technical review on data leakage detection and prevention approaches, Journal of Network Communications and Emerging Technologies (JNCET).

Rastogi V., Chen Y. and Enck W. (2013) AppsPlayground: automatic security analysis of smartphone applications, in Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy (CODASPY '13): New Orleans, La, USA, 209–220.

Roemer, R. Buchanan, E. Shacham, H. and Savage, S.(2012). Return-oriented programming: systems, languages and applications, ACM Transactions on Information and System Security,15(1):2.

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: Proceedings of the 8th Australian Information Security Management: 2.

Shi. Y. (2004). Gatekeeper: Monitoring auto-start extensibility points (ASEPs) for spyware management. In LISA '04: Proceedings of the 18th USENIX conference on System administration. USENIX Association, Berkeley, CA, USA, 33–46.

Salton G. and Buckley C. (1988) Term-weighting approaches in automatic text retrieval, Information Processing & Management, 24(5):513–523.

Salton G., Wong A. and Yang C. (1975) A vector space model for automatic indexing, Communications of the ACM, 18(11):613–620.

Shu X., Elish K. O., D. Yao, Ryder G. and Jiang X., (2015). Profiling user-trigger dependence for Android malware detection, Computers & Security, 49:255–273.

Shu, X. Elish, K. O. Yao, D. Ryder, B. G. and Jiang, X. (2015). Profiling user-trigger dependence for Android malware detection, Computers & Security, 49:255–273.

Smalley S. Craig R. (2013). Security Enhanced (SE) Android: Bringing Flexible MAC to Android. Proceedings of the 20th Annual Network and Distributed System Security Symposium. (2):20–38.

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, 14(2):141–153.

Thomas K., Li F., Zand A. (2017). Data Breaches, phishing, or malware? understanding the risks of stolen credentials, in Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security, CCS 1:1421–1434.

Ullah F., Edwards M., Ramdhany R., Chitchyan R., Babar M. and Rashid A. (2018). Data exfiltration: A review of external attack vectors and countermeasures. Journal of Network and Computer Applications, 101:18-54.

Wang D., Cheng H., Wang P., Yan J. and Huang X. (2018). A security analysis of honeywords, in Proceedings of the Network and Distributed Systems Security (NDSS) Symposium,:18–21.

Wang, D., Li, W. and Wang, P. (2018). Measuring Two-Factor Authentication Schemes for Real-Time Data Access in Industrial Wireless Sensor Networks. IEEE Transactions on Industrial Informatics, 14: 4081-4092.

Wang J. (2005) Information security models and metrics, in Proceedings of the 43rd annual southeast regional conference on ACMSE43:178–184.

Wang W., Yang J. and Muntz R. (1997) Sting: a statistical information grid approach to spatial data mining: 186–195.

Wei, F., Roy, S. and Zhou, X. (2014) Amandroid: A precise and general intercomponent data flow analysis framework for security vetting of android apps,: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM: 1329-1341.

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.

Willems, C., Holz, T. and Freiling, F. (2007). Toward automated dynamic malware analysis with concatenated dynamic taint analysis. International Conference on Availability, Reliability and Security, pp 355–362.

Xiang C., Binxing F., Lihua Y., Xiaoyi L. and Tianning Z. (2011). Andbot: towards advanced mobile botnets, in Proceedings of the 4th USENIX Conference on Large-scale Exploits and Emergent Threats: 11.

Xu W., Xiang S. and Sachnev V. (2018) A cryptograph domain image retrieval method based on paillier homomorphic block encryption, Computers Materials and Continua: 1–11.

Xu, J., Sung, A. H., Chavez, P. and Mukkamala, S. (2004). Polymorphic malicious executable 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,:29.

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,: 1043-1054.

Zhang T., Ramakrishnan R. and Livny M. (1997) BIRCH: a new data clustering algorithm and its applications. Data Mining and Knowledge Discovery, 1(2): 141–182.

Zhang Y., Yang M., Xu B. (2013) Vetting undesirable behaviors in Android apps with permission use analysis, in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS ‘13): 611–622.

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 Device, pp 93-94.

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.

Zhou, Y. and Jiang, X. (2012). Dissecting Android malware: characterization and eution, proceedings of the 33rd IEEE Symposium on Security and Privacy, San.