Review of Different Techniques Used For Android Application Security

1Sarita T. Sawale, 2Pravinkumar M. Ghaywat

1Assistant Professor Department of Information technology Anuradha Engineering College
2Student, M.E. (CSE) Department of Computer Sci. & Engg. Anuradha Engineering College

ABSTRACT
Since last-decade, smart-phones have gained widespread usage. Mobile devices store personal details such as contacts and text messages. Due to this extensive growth, smart-phones are attracted towards cyber-criminals. In this research work, we have done a systematic review of the terms related to malware detection algorithms and have also summarized behavioral description of some known mobile malwares in tabular form. After careful solicitation of all the possible methods and algorithms for detection of mobile-based malwares, we give some recommendations for designing future malware detection algorithm by considering computational complexity and detection ration of mobile malwares.

KEYWORDS:- Smart-phones, Malware, Attacks, Static analysis, Dynamic analysis.

1. INTRODUCTION
Now, there is a thin line difference between Smart-phones, PCs(Personal Computers) and other newly emerged devices like tabs, notebooks and laptops as all are now connected technologies. Due to various services like social networking and gaming provided by smart-phones with the help of applications, these are exposed to gain some confidential information from mobile-devices. Smart-phone OSs includes symbian, android, PalmOS and embedded Linux etc. Android is the popular platform for smart-phone based malware authors as any third-party vendor can create applications for android phones and deploy it on android market. Sometimes, even trusted applications are able to leak user's location and phone's identity and share it on server without its consent. Due to this growing skill-set of cyber-criminals who device their algorithms for breaching privacy, embarrassing service-provider and bring inconvenience to the users. So, it requires special care to secure these networked devices from malwares with the help of anti-developed techniques and algorithms for detection. This paper focuses on describing mobile-based threats and its counter detection techniques.

1.1 Current State of Study
This section discusses some current malwares reported by security researcher groups. In 2010, different types of mobile malwares are found including DroidDream, Geimimi, GGTacker, Plankton Tonclank and HongTomTou. These malwares are much like original CabiR worm. Look Out security firm reported that over one million of android devices are affected in first half of 2011[1]. In 2012, it is reported by Homeland security department that 79 percent of the mobile threats were targeted to Android operating systems. In January 2012, Symantec identified Trojan horse named Android Counter clank for stealing information [2]. Security firm Kaspersky found in 2013 that 98 percent of malware was directed at android platform.

1.2 Organization of paper
In this paper section 2 will discuss mobile device attack vectors and types of detection techniques for mobile malwares. Section3 will discuss detection techniques and algorithms proposed by various researchers and section 4 will give conclusion by analyzing various techniques proposed by different researchers followed by some future recommendations.

2. MOBILE MALWARES
Mobile devices are targets to malware authors, as they store sensitive data and connected technology by means of blue-tooth connectivity, SMS/MMS messages and Wireless LAN
2.1 Mobile Device Threats
Numerous attack vectors exist which compromises security of mobile devices [3]. Three main categories of attacks could be carried over mobile devices which includes- malware attacks, grayware attacks and spyware attacks described as:-

2.1.1 Malware - These kind of attacks steal personal data from mobile devices and damage devices [4]. With device vulnerabilities and luring user to install additional apps, attacker can gain unauthorized root access to devices. Some of the malware attacks are listed as:-

- Bluetooth attacks: With Bluetooth attacks, attacker could insert contacts or SMS messages, steals victim's data from their devices and can track user’s mobile location. Blue-bugging is kind of blue-tooth attack through which attacker could listen conversations by activating software including malicious activities [4].
- SMS attacks: Through SMS attacks, attacker can advertise and spread phishing links. SMS messages can also be used by attackers to exploit vulnerabilities [4].
- GPS/Location attacks: User’s current location and movement can be accessed with global positioning system (GPS) hardware and then information can be sold to other companies involved in advertising[4].
- Phone jail-breaking: With jail-breaking, an attacker can remove security implications of operating system like it allows OS to install additional and unsigned applications. Users are attracted to install them as they could get additional functionality [4].
- Premium rate attacks: They posed serious security concerns because premium rate SMS messages could go unnoticed until attacker faces thousands or dollars of bill on his device as they don't need permissions to send SMS on premium rated numbers [4].

2.1.2 Grayware: Grayware include applications which collects the data from mobile devices for marketing purposes. Their intention is make no harm to users but annoy them.

2.1.3 Spyware: Spyware collects personal information from user’s phone such as contacts, call history and location. Personal spyware are able to gain physical access of the device by installing software without user’s consent. By collecting information about victim's phone, they send it to attacker who installed the app rather than the author of the application.

2.2 Behavioral Classification
Malware may also be classified on the basis of their behavior. Table 1 shows behavioral classification of some known malwares:-

<table>
<thead>
<tr>
<th>Malwares</th>
<th>Behavior</th>
<th>Description</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexiSPY</td>
<td>Stealing user</td>
<td>Track user information, emails, photos, browser history and then send it to server.</td>
<td>Symbian, Windows Mobile and BlackBerry</td>
</tr>
<tr>
<td>Fakeplayer</td>
<td>Content delivery</td>
<td>Runs in background when clicking on media player application. Send SMS Messages to premium rated numbers</td>
<td>Android OS</td>
</tr>
<tr>
<td>Zeus (Mobile)</td>
<td>Stealing user</td>
<td>Forwards incoming SMS messages from mobile phones to remote server for access of bank accounts</td>
<td>Android OS</td>
</tr>
</tbody>
</table>
3. MALWARE DETECTION TECHNIQUES
Malwares must be analyzed in order to understand risk associated with malwares. However, many detection techniques exist in literature to expose the behavior and functionality of malwares. Detection techniques are broadly categorized into three types- static analysis, dynamic analysis and permission-based analysis.

3.1 Static analysis
Static analysis investigates downloaded app by inspecting its software properties and source code. However, obfuscation and encryption techniques embedded in software makes static analysis difficult. Static analysis is further categorized into two categories- signature-based detection and behavior-based detection traditionally used by anti-viruses.

Kim et al. [5] proposed framework for detection and monitoring of energy greedy threats by building power consumption from the collected samples. After generating power signatures, data analyzer compares them with signatures present in a database. Batyuk et al. [6] proposed system for static analysis of android applications. First, they provide in-depth static analysis of applications and present readable reports to user for assessment and taking security relevant decisions-to install or not to install an application. Then the method is developed to overcome security threats introduced by the applications by disabling malicious features from them. Ontang et al.[7] proposed Secure application Interaction Framework (Saint) by extending android security architecture for protection of interfaces and enhancing interaction policies between calling and callee applications.

Wei et al.[8] proposed a static feature-based approach and develop system named Droid Mat able to detect and distinguish android malware. Their mechanism considers the static information including permissions, intents and regarding components to characterize android malware, clustering algorithm is applied to enhance malware modeling capability.K-Nearest Neighbor algorithm classify applications as benign and malicious applications. Finally their results are compared with well known tool Androguard, published in Blackhat 2011 and it is found that DroidMat is efficient as it takes only half time than Androguard to predict 1738 applications.

Bose et al. [9] present behavioral detection framework for representation of malware behavior by observing logical ordering of applications actions. Malicious behavior is discriminated from normal behavior by training SVM. System is evaluated for both real-world and simulated mobile malwares with 96% accuracy.

Schmidt et al.[10] describes a method for symbianOS malware analysis called centroid based on static function call analysis by extracting features from binaries and clustering is applied for detection of unknown malwares. VirusMeter [11] is proposed to detect anomalous behavior on mobile devices by catching malwares which are consuming abnormal power. Machine learning algorithms helped to improve its detection accuracy. pBMDS [12] an approach through which user-behavior is analyzed by collecting data through logs of key-board operations and LCD displays and then correlated with system calls to detect anomalous activities. Hidden markov model(HMM) is leveraged to learn user-behavior and malware behavior for discrimination of differences between them.
3.2 Dynamic analysis
Dynamic analysis involves execution of application in isolated environment to track its execution behavior. In contrast to static analysis, dynamic analysis enables to disclose natural behavior of malware as executed code is analyzed, therefore immune to obfuscation attempts.

Batuk et al. [13] proposed an android application sandbox (AA Sandbox) system for analysis of android applications consists of fast static pre-check facility and kernel space sand-box. For suspicious application detection, both static and dynamic analysis is performed on android applications. AA Sandbox takes APK file and list out following files by decompressing them-Androidmanifest.xml, res/, classes.dex. Manifest file holds security permissions and description of application. Res/ folder defines layout, graphical user interface (GUI) elements and language of application. Classes.dex file contains executable code for execution on dalvik virtual machine which is then de-compiled to java files with baksmali and then code is searched for suspicious patterns. Monkey program designed for stress testing of applications generates pseudo random sequences of user-events such as touches and mouse-clicks. It is used to hijack system calls for logging operation and helpful to get the logging behavior of application at system level. Around 150 applications are collected for testing and evaluation.

Min et al. [14] proposed run-time based behavior dynamic analysis system for android applications. Proposed system consists of event detector, log monitor and parser. Event trigger is able to simulate the user's action with static analysis. Static analyzer generates manifest.xml and java code with the help of application .apk file. Semantic analysis find list of risk based permissions, activities and services including other information such as hash code and package name. Data flow analysis creates control flow graph (CFG) of the application by mapping of user-defined methods and API calling. By running application in a customized emulator with loadable LKM, sensitive information about application can be captured such as sent SMS, call log and network data for entry address of system calls. Logs recorded with debugging tool logcat for sensitive behavior sent to Log parser. Log monitor gathers log data as the application runs and parser analyzes log data by picking sensitive information and filtering out unnecessary information. By collecting 350 apps from the Amazon Android Market, results found that about 82 applications leak private data.

Enack et al. [15] proposed Apps-playground framework for automatic dynamic analysis of android applications. Designed approach is able to analyze malicious applications in addition to applications leaking private data from smartphones without the user's consent. Dynamic analysis should possess detection techniques including ability to explore application code as much as possible and the environment should be as much real that malicious application could not obfuscate. Automatic analysis code integrates the detection, exploration and disguise techniques to explore android applications effectively. Detection techniques detect the malicious functionality while app is being executed. It includes taint tracing which monitor sensitive APIs with TaintDroid such as SMS APIs and kernel level monitoring for tracing of root exploits. Automatic exploration techniques are helpful for code coverage of applications by simulating events such as location changes and received SMS so that all application code is covered. Fuzzy testing and intelligent black box execution testing is used for automatic exploration of android applications. Disguise techniques create realistic environment by providing data such as International mobile equipment identity (IMEI), contacts, SMS, GPS coordinates etc.

Enck et al. [16] proposed TaintDroid for dynamic analysis. First dynamic analysis tool used for system wide analysis of android applications by tracking flow of sensitive information through third-party applications. TaintDroid integrates multiple granularities at object level i.e., variable, method, message and file level. It is able to monitor how the sensitive data are used by applications and then taints are labeled. TaintDroid is tested on around 30 applications and it is found that 15 of them uses personal information.

3.3 Permission-based analysis
With the help of listed permissions in manifest.xml, various researchers are able to detect applications malicious behavior. [17] These permissions have the ability to limit application behavior by controlling over privacy and reducing bugs and vulnerabilities.
Johnson et. al. [18] proposed architecture for automatic downloading of android applications from the android market. Different algorithms employed for searching of applications such as downloading applications by application category. With static analysis, required permissions can be obtained based on its functionality. Permission names are searched in android source code and then mapped with API calls to know that whether requested permissions are correct or not. Program examines all smali files of application to obtain list of method calls used in an application. Each method call is then compared with method call listed in permission protected android API calls to know exact permissions. Restricted permission set is compared with all the permissions specified in AndroidManifest.xml file to find out extra permissions, lacking of permissions and exact permission set required for its functionality.

Zhou et al. [19] proposed DroidRanger for systematic study on overall health of both official and unofficial Android Markets with the focus on the detection of malicious apps. DroidRanger leverages a crawler for collection of apps from the Android Market and saved into local repository. Features extracted from collected apps include requested permissions and author information. Two different detection engines are used for detection of known and unknown malwares. First detection engine is permission-based behavioral foot-printing scheme able to distil apps requiring dangerous permissions such as SEND_SMS and RECEIVE_SMS permissions. Therefore, number of apps to be processed for second detection engine is reduced. In second step, multiple dimensions for behavioral foot-printing scheme chosen for listening of all system-wide broadcast messages if they contains receiver named android.provider.Telephony.SMS_RECEIVED. Obtained callgraph associates API calls to specific components specified in a rule. For example- by calling abortBroadcast function with specific rule, a method is obtained to detect apps monitoring incoming SMS messages. Second detection engine includes some heuristics to detect suspicious apps and zero-day malwares. Heuristics attempts to dynamically fetch and run code from untrusted websites which is further monitored during run-time execution to confirm whether it is truly malicious or not.

<table>
<thead>
<tr>
<th>Type of Approach</th>
<th>Name of Method</th>
<th>Features Extracted</th>
<th>Classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static Approach</strong></td>
<td>Bote</td>
<td>Application Actions</td>
<td>SVM</td>
</tr>
<tr>
<td></td>
<td>pBMDS</td>
<td>Keyboard Operation and LCD Display Logs</td>
<td>Hidden Markov Model (HMM)</td>
</tr>
<tr>
<td></td>
<td>VirusMeter</td>
<td>Power Consumption</td>
<td>Logistic Regression, Neural Networks and Decision Trees</td>
</tr>
<tr>
<td></td>
<td>DroidMat</td>
<td>Permissions and application components</td>
<td>Clustering and K-Nearest Neighbor</td>
</tr>
<tr>
<td></td>
<td>Schmidt</td>
<td>Function Calls</td>
<td>Clustering Algorithm</td>
</tr>
<tr>
<td><strong>Dynamic Approach</strong></td>
<td>Batsyak</td>
<td>Android Application Sandbox</td>
<td>Kernel Space Sandbox</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>Event Detector</td>
<td>Simulate the User Action</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>User Define</td>
<td>API Calling</td>
</tr>
<tr>
<td></td>
<td>Enack</td>
<td>Automatic Dynamic Analysis</td>
<td>Malicious Application</td>
</tr>
<tr>
<td></td>
<td>Earck</td>
<td>TaintDroid</td>
<td>System Wide Analysis</td>
</tr>
<tr>
<td><strong>Permission Based Approach</strong></td>
<td>Johnson</td>
<td>Automatic Downloading</td>
<td>Android Market</td>
</tr>
<tr>
<td></td>
<td>Zhou</td>
<td>Systematic Study on Overall Health</td>
<td>Official &amp; Unofficial Android Market</td>
</tr>
<tr>
<td></td>
<td>AbortBroadcast</td>
<td>Incoming SMS Messages</td>
<td>Detect Application Monitoring</td>
</tr>
</tbody>
</table>
4. CONCLUSION
Smart-phones are becoming popular in terms of power, sensor and communication. Modern, smart-phones provide lots of services such as messaging, browsing internet, emailing, playing games in addition to traditional voice services. Due to its multi-functionality, new security threats are emerged for mobile devices. In this paper, we presented survey on various techniques for detection of mobile malware. We have categorized various mobile malware detection techniques based on features extracted from them and monitoring system calls as they provide us low level information. We have analyzed that information-flow tracking, API call monitoring and network analysis provide more deeper analysis and useful information for detection of mobile malware.

5. FUTURE SCOPE
Following are some recommendations for designing algorithm to detect mobile-based applications containing malwares.
1. Multiple sources for feature extraction should be used for building feature-set to detect mobile malwares.
2. There should be national or international database for reporting malware incidents so that developers are aware of distinct vulnerabilities related to mobile malwares.
3. Artificial intelligence algorithms(neural network-based) should be used to improve detection ratio.
4. Machine to machine communication and authentications tools must be used in between multiple device platforms.

REFERENCES


