Malware Prevention and Detection System using Smart Phone

Sachin M. Kolekar
Department of Computer Engineering,STES's Smt. Kashibai Navale College of Engineering, Pune-41

Parikshit N. Mahalle, Ph.D
Department of Computer Engineering,STES's Smt Kashibai Navale College of Engineering, Pune-41

ABSTRACT
Mobile malware is a malicious software.This software used to disrupt computer operation. This paper surveys current state of mobile malware in the wild. Types of malware includingviruses,Trojans,Rootkits,Zombies,worms,Spyware,adware,spam,email and Denial Of Services(DOS) attaks. Survey the different types of operating system in different differ mobile.In this paper, we present the different types of malware detection techniques & discuss the smart phone security challenges.

Keywords
Kirin security,Cloud-base detection,malware, Access Contro, spyware l, prevention,Decision table.

1. INTRODUCTION
Malware is malignant software that is specially built to assail mobile phone or keenly intellective phone systems. botnets, worms,and Trojan horses.[1]Initially, malware merely highlighted a software system's security susceptibilities, but the motivations behind it gradually transmuted, and its authors now use malware to gain financial benefits on a more sizably voluminous scale.

Malware aimed at Android smart phones alone has grown 76% over the last few months, threatening Android security, and other platforms are additionally coming under attack. In additament to malware, the other two major categories of threats to mobile contrivances are personal spyware and grayware. Spyware amasses information such as utilizer location,SMS messages, and call history without the victim's cognizance.Spyware can't be labeled as illicit because it doesn't send information to the application's author, but installing personal spyware on a mobile phone without the contrivance owner's sanction could be considered unethical.

2. MOBILE THREAT MODEL
In this paper present three types of threats posed by third-party smart phone applications and discuss the security measures that are intended to detect and prevent them.

2.1 Types of Threat
In mobile threat model includes main two types of threats: grayware, and Anti-spyware. We distinguish between the three predicated on their distribution method, licit-ity, and notice to the utilizer. This paper focuses Specially on malware; personal spyware and grayware use different attack vectors, have different motivations, and require different bulwark mechanisms.

2.1.1 Grayware
Greyware refers to a malignant software or code that is considered to fall in the "grey area" between mundane software and a virus. Greyware is a term for which all other maleficent or exasperating software such as adware, spyware, trackware, and other maleficent code and malevolent shareware fall under.

2.1.2 Anti-spyware[1]
Anti-spyware is a type of software that is designed to detect and abstract unwanted spyware programs. Spyware is a type of malware that is installed on a computer without the utilizer's cognizance in order to amass information about them. This can pose a security risk to the utilizer, but more frequently spyware degrades system performance by taking up processing puissance, installing supplemental software, or redirecting users' browser activity.

2.2 Security measures in smart phone[2]
Smartphone operating system vendors use curated mar-kets and/or application sanctions to bulwark users. Wefixate on iOS, Android, and Symbian 9.x.

Markets Application**:** Smartphone users are emboldened to download and purchase applications from centralized application mar-kets. Apple, Google, and Nokia promote the utilization of central-ized markets with decrementing rigor.Apple iOS contrivances sanction the utilizer to install applications only from the Apple App Store, and applications in the Apple Store are reviewed by Apple for security purpose. If iOS users want to install applications from any other sources, then they must jailbreak their contrivances, which involves exploiting a vul-nerability in iOS to gain super-user access. This process carries the jeopardy of rendering the phone inoperable, and it voids the phone's warranty. Apple's review process is intended to avert malware from being distributed through the Application Store. Their review standards withal disallow personal spy-ware, but an assailant with physical access to the victim's contrivance could jailbreak the phone without the victim's knowledge to install personal spyware. The App Store is kenned to have included grayware in some cases, the grayware has been abstracted from the App Store.

Android additionally provides users with an official application store, the Android Market.Most Android phones sanction users to additionally install applications from unofficial though the utilizer is admonished that installing non-Market applications may expose the utilizer to malware. Google does not any type of review applications prior to listing them in the Android Market, albeit they may review some applications later. Personal spyware (e.g., GPS Spy Plus) and grayware are listed in the Android Market. The Android security team has abstracted malware from the Android Market following utilize complaints, and they are able to remotely uninstall kenned malware from users' contrivances.

Nokia runs Ovi, which is currently the official Symbian application market. Like the Apple App Store, all applications are reviewed prior to being listed in Ovi. Symbian does not avert or dismay users from installing applications from

other sources. Several popular alternative markets are available, and they lack review processes. However, Sym-bian offers an application signing accommodation that incorporates security reviewing. Only Symbian Signed applications are sanctioned to access hazardous privileges. All Symbian Signed applications must undergo an automated security review.Applications that utilize the most hazardous privileges are additionally reviewed by humans, and some number of other Symbian Signed applications are withal human-reviewed. As a consequence of the signing process, many applications in third-party Symbian markets have undergone review.

3. PREVENTION APPROACH

For our malware prevention approach to be usefull in practice, it must gratify the following properties:

3.1 Lightweight-ness

The approach should be lightweight in terms of the sundry required resources available on the phone, such as recollection, computation and battery power.

3.2 Efficiency

The approach should incur small delay.Otherwise, it can affect the overall usability of the system. We believe that no more than a few seconds should be spent executing the approach.

4. MALWARE DETECTION TECHNIQUES [2]

Malware detection techniques available for detecting mobile malware and other security susceptibilities have varying strengths and weaknesses.

4.1 Signature-based malware Detection

A pattern-marching approach commercial antivirus is an example of signature predicated malware detection where the scanner scans for a sequence of byte within a program code to identify and report a malignant code. This approach to malware detection adopts a syntactic level of code injuctive authorizations in order to detect malware by analysing the code during program compilation. This technique conventionally covers consummate program code and within a short period of time. However, this method has constraint by ignoring the semantics of injuctive authorizations,which sanctions malware obfuscation during the program's run-time.

In smart phone operating systems, the comportment of malware may occur in multiple locations, the occurrence of these acts amalgamated according to certain timing in order to constitute maleficent software comportment, one or a few of these separate demeanor can't determine whether they are malevolent behaviour's or not. This amassment is then processed by temporal cognations, after all the demeanors are abstracted and signed to software demeanor patterns. Code packing, simple scrambling does not transmute the demeanor of software, malware and its variants are generally in the same run-time deportment patterns, the signature of these malware can be detected through the same deportment Compared with feather-predicated malware detection method, the signature database of comportment signature predicated is becoming more minute, so the be behaviour's predicated detection of malevolent software signature.is ideal for resource-constrained mobile contrivances. Incipient malevolent software customarily include incipient comportment signature that is inconsistently erratic with the antecedent kenned mundane demeanor, so comportment-predicated malware detection signatures can detect incipient and unknown malware.

4.2 Specification-based malware detection

Designation predicated detection makes utilization of certain rule set of what is considered as mundane in order to decide the maleficence of the program contravening the predefined rule set. Thus programs transgressing the rule set are considered as maleficent program. In designation-predicated malware detection, where a detection algorithm that addresses the deficiency of pattern-matching was developed. This algorithm incorporates ordinant dictation semantics to detect malware instances. The approach is higer resilience to prevalent obfuscation techniques. It used template T to describe the malevolent demeanors of a malware, which are sequence of ordinant dictations represented by variables and special symbolic constants. The circumscription of this approach is that the attribute of a program cannot be accurately designated. Designation-predicated detection is the derivate of anomaly predicated detection. Instead of approximating the implementation of a system or application, specification based detection approximates the requisites of application or system. In designation-predicated system there subsists a training phase which endeavors to learn the all valid comportment of a program or system which needs toinspect. The main constraint of designation predicated system is that it if very arduous to accurately designate the deportment the system or program. One such implement is Panorama which captures the system wide information flow of the program under inspection over a system, and checks the deportment against a valid set of rule to detect malevolent activity.

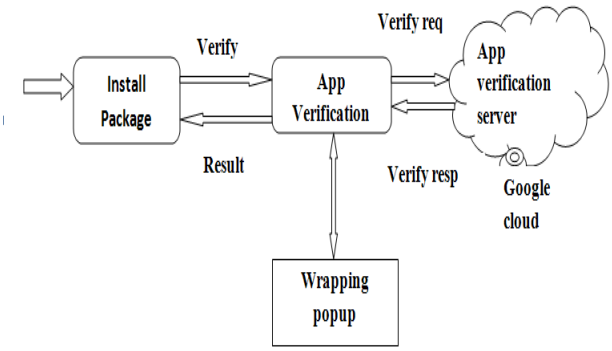
4.3 Data Mining Technique of Detecting Malware

In data mining methods for detecting maleficent executables, a maleficent executable as a program that performs function, such as compromising a system security, damaging a system or obtaining sensitive data without any user's sanction. Their data mining methods detect patterns in immensely czolossal amounts of data, such as byte code, and utilize these patterns to detect future instances in kindred data. Their framework used classifiers to detect incipient maleficent executables. A classifier is a rule set, or detection model, engendered by the data mining algorithm that was trained over a given set of training data. They designed a framework that used data mining algorithms to train multiple classifiers on a set of maleficent and benign executables to detect incipient examples. The binaries were first statically analysed to extract properties of the binary, and then the classifiers trained over a subset of the data. Their sizably voluminous sets of programs from public sources were dissevered into differnt two classes that is malevolent and benign executables. Example of this data set is a Windows and MS-DOS formats executable, which is additionally applicable to other formats. Since the virus scanner was updated and the viruses were obtained from public sources, it was surmised that the virus scanner has a signature for each malevolent virus. They then split the dataset into two subsets: the training set and the test set.The data mining algorithms utilized the training set while engendering the rule sets. The test set was then used to check the precision of the classifiers over unseen examples. This data mining method was able to detect aforetime undetectable maleficent executables by comparing the results with traditional signature-predicated methods and with other learning algorithms. The Multi-Naive Bayes method had the highest precision and detection rate of any algorithm over unknown programs, 98.76%, over double the detection rates

of signature based methods. Its rule set was withal more arduous to subjugate than other methods because all lines of machineinjuctive authorizations would have to be transmuted to evade detection.

4.4 Cloud-Based Detection [5]

In this scheme a lightweight client application monitors the system calls in the contrivance and sends it to the server in cloud to detect malignant department.Paranoid Android is a cloud-predicated malware bulwark technique that moves security analysis and computations to a remote server that hosts multiple replicas of mobile phones running on emulators A tracer, located in the smart phone, records all the indispensable information required to reply to the mobile application's execution. The tracer transmits the recorded information to the cloud-predicated replier,which replays the execution in the emulator. The replier can deploy several security checks, such as dynamic malware analysis, recollection scanners, system call anomaly detection, and commercial antivirus scanning from the cloud's ample resources.Crowdroid is a demeanor-predicated mobile malware detection technique for Android. Crowdroid is a lightweight client application that monitors system calls invoked by the target mobile application, preprocesses the calls, and sends them to cloud where a clustering technique avails determine whether the application is maleficent. Increased utilization of Crowdroid results in ameliorated malware detection but utilizing the approach initially might cause erroneous positives, as the sample size is still diminutively minuscule.



"Fig1.cloud based Detection"

5. MALWARE ANALYSIS TOOL

5.1 Multiple-Path Pxploration

Automatic dynamic malware-analysis implements engender their reports predicated on a single execution trace of the sample under analysis. The utilization of logic bombs sanctions malware to only reveal its maleficent deportment predicated on arbitrary constraints. For example, a malware sample could defer its malevolent activities until a certain date is reached or stop executing if obligatory files cannot be found on the infected system. To overcome this shortcoming, Moser et al. [2007a] present an implement capable of exploring multiple execution paths for Windows binaries. This implement apperceives a branching point whenever a control-flow decision is predicated on data that originates outside the monitored process.This data can only be introduced to the process via system calls. Thus, a branching point is detected if a control-flow decision is predicated on a return value of a system call.Every time such a situation occurs, the implement takes a snapshotof the running process that sanctions the the system to reset to this state. Execution process is perpetuated and after a timeout the system is reset to the recorded snapshot. Then, the value that is responsible for the control-flow decision is manipulated such that the control flow decision making is inverted, resulting in the execution of the alternate path.

This approach elongates Anubis and applies dynamic taint tracking to analyze how data returned from system calls is manipulated and compared by the process under analysis. The system calls are responsible for introducing the taint-labels handle-file system and registry access, as well as network activities and date/time information. When manipulating a value upon resetting the system state, special care is taken by the system to update the value utilized in the corresponding compare ordinant dictation in a consistent manner. This designates that not only the value directly involved in the comparison must be transmuted, but all other recollection locations that depend on this value must be manipulated in a consistent manner to make the execution of alternative paths feasible. To achieve this, the system stores a set of recollection locations for each branching point that depends on the compared value amalgamated with a set of linear constraints describing these dependencies. During a reset, the set of constraints is evaluated by a constraint solver to engender the values that need to be superseded to coerce execution down the other path. If a dependency cannot be modeled as a linear constraint the system is unable to update the recollection locations in a consistent manner.

5.2 Norman Sandbox

Norman Sandbox[3] is a dynamic malware-analysis solution which executes he sample in a tightly controlled virtual environment that simulates a Windows operating system. This environment is utilized to simulate a host computer as well as an affixed local area network and, to some extent, Internet connectivity. The core conception a baft the Norman Sandbox is to supersede all functionality required by an analyzed sample with a simulated version thereof. The simulated system, thus, has to provide support for operating system-relevant mechanisms, such as memory protection and multithreading support. Moreover, all required APIs must be present to give the sample the fake impression that it is running on a real system. Because the malware is executed in a simulated system, packed or obfuscated executables do not hinder the analysis itself. As described in a packed binary would simply perform the unpacking step and then continue executing the original program. However, to minimize the time spent in analysis, binaries that are obfuscated by a known packer program are unpacked prior to analysis.

Norman Sandbox fixates on the detection of worms that spread via email or P2P networks, as well as viruses that endeavor to replicate over network shares. In additament, ageneric malware-detection technique endeavors to capture other malevolent software.The Norman Sandbox provides a simulated environment to the sample under analysis consisting of custom-madeversionof user-land APIs necessary for executing the sample. The functions providing these APIs are heavily instrumented with the corresponding analysis capabilities. Furthermore, to keep the simulation self-contained, these replacement APIs do not perform any interactions with the real system. Instead, the results of such API calls are created to allow the malware to continue execution (e.g., filling in the correct API function-return values). Bookkeeping takes place if required to thwart some detection techniques applied by malicious software. For example,a malware sample might try to detect the analysis tool by writing to a file and trying to read from that file later on to check if the stored information is still there. If the

analysis tool does not provide the correct results to the read request, the malware can detect that it is being analyzed and will terminate without revealing its true malicious intents.

Special care is taken with respect to networking APIs. All networking requests issued by the sample under analysis are redirected to simulated components. If, for example,a sample intends to spread itself via email, it has to contact an SMTP server to send email. The connection attempt to TCP port 25 is detected, and instead of opening a connection to the real server, the connection is redirected to a simulated mail server.This is not deleted by the sample under analysis, and it will start sending the mail commands, including the malicious payload. An analogous approach is followed when a sample tries to write to a simulated network share or tries to resolve host names to IP addresses via DNS queries.

6. LITERATURE [6]

As part of our survey, we examined the sanctions of Android malware. Android application malware commonly requests the capability to direct send sms messages,which is not common among non malicious applications. However, we were unable to identify any other permission-based patterns for malware classification. Permission-based classification will require future consideration as the set of known Android malware grows. We also observed that none of the malware in our data set was approved by the Apple Application Store, which indicates that human review may be an effective preventative measure for malware. Symbian's automated review-and-sign process fared worse; nearly a third of the Symbian malware in our data set was approved by or evaded this process.Currently, both malware authors and smartphone users are incentivized to find root exploits. The homebrew community publishes root exploits to help smartphone owners customize their phones. However, malware can use these same root exploits to circumvent smartphone security mechanisms; indeed, 4 pieces of malware in our data set do this.We consider the impact of the homebrew community and find that root exploits are available between 74% and 100% of phones' lifetimes. We recommend that phone manufacturers support smartphone customization so that the homebrew community does not need to seek root exploits.

Survey the state-of-the-art analysis techniques as well as the implements that avail an analyst to expeditiously and in detail gain the required erudition of a malware instance's demeanor. study the dynamic analysis tools with the help of dynamic analysis tool process of executing a malicious sample and monitoring its behavior. Most dynamic analysis tools implement functionality that monitors which APIs are called by the sample under analysis, or which system calls are invoked. Several analysis tools provide the functionality to observe how sensitive data is processed and propagated in the system.Automated dynamic analysis results in a report that describes the observed actions the malware has performed while under analysis. These reports can be compiled into behavioral profiles that can be clustered to amalgamate samples with homogeneous behavioral patterns into coherent groups (i.e., families). Furthermore, this information can be used to decide which incipient malware samples should be given priority for exhaustive analysis(i.e., manual inspection). In order to achieve this, behavioral profiles of incipient threats can be automatically engendered by an analysis implement and compared with the clusters.While samples with behavioral profiles near a subsisting cluster probably are a variation of the corresponding family, profiles that deviate considerably from all clusters likely pose an incipient threat worth analyzing in detail. This prioritization has become compulsory as techniques such as polymorphic encodings or packed binaries sanction assailers to release hundreds of incipient malware instances every day. Although such samples might evade static signature matching, their homogeneous deportment observed through dynamic analysis might reveal their affiliation with a given malware family.

7. PROPOSED WORK

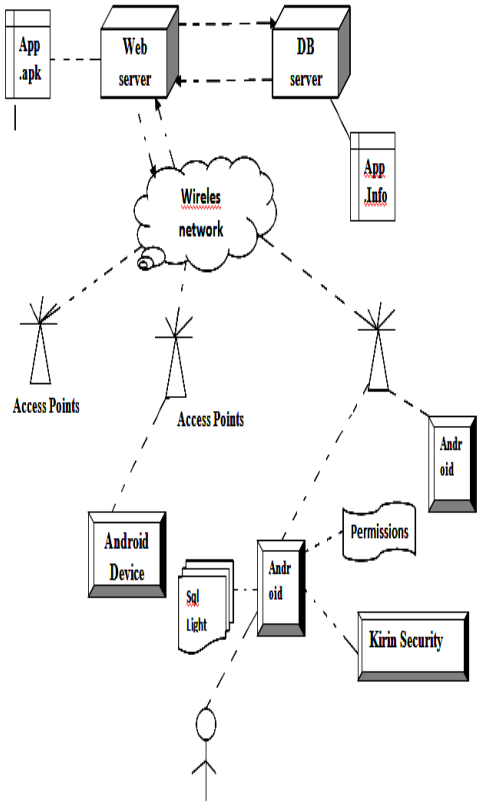


Fig2.Poposed work with Kirin security

An application certification for Android. Kirin[1] performs a sanction check on the application during installation. When a utilizer installs an application, Kirin extracts its security configurations and checks them against the security policy rule that it already has. If an application fails to pass all the security policy rules, Kirin can either expunge it or vigilant the utilizer.

In proposed work when uploading application (.apk file), application owner has to specify information about the list of permissions application is going to use. All the information about application will be stored on centralize server. This information will be used to verify Kirin Security Service, if verification is true then application is allowed to download otherwise application is marked as malware.

8. CONCLUSION

This paper presented a comprehensive overview of the state-of-the-art analysis techniques as well as the tools that aid an analyst to quickly and in detail gain the required knowledge of a malware instance's behavior. Additional work in the proposed system is, the malware dynamic detection which can be achieved with the combination of kirin security service. Information will be used to verify the application at the time installation and if verification is true then application

installation is processed else the installation is rejected which adds more security to a Smartphone by detecting the malware.

9. REFERENCES

[1] A.P. Felt et al., "A Survey of Mobile Malware in the Wild,"*Proc. ACM Workshop Security and Privacy in Mobile Devices*(SPMD 11), ACM, 2011, pp. 3-14.

[2] Adebayo, Olawale Surajudeen, Mabayoje, Amit Mishra, Osho Oluwafemi, "Malware Detection, Supportive Software Agents and ItsClassification Schemes", International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.6, November 2012.

[3] T. Blasing et al., "An Android Application Sandbox System for Suspicious Software Detection," *Proc. 5th Int'l Conf. Malicious and Unwanted Software* (Malware 10), ACM, 2010,pp. 55-62.

[4] M. Egele et al., "A Survey on Automated Dynamic Malware Analysis Techniques and Tools," *ACM ComputingSurveys*,2012;https://www.seclab.tuwien.ac.at/papers/malware_survey.pdf.

[5] W. Enck et al., "A Study of Android Application Security,"Proc. 20th Usenix Security Symp., Usenix, 2011;http://static.usenix.org/events/sec11/tech/full_papers/Enck.pdf.

[6] IDC, "Mobile Phone Market Grows 17.9% in Fourth Quarter, According to IDC," press release, 28 Jan.2011; www.idc.com/about/viewpressrelease.jsp?containerId=prUS22679411.

[7] D. Barrera et al., "A Methodology for Empirical Analysis of Permission-Based Security Models and Its Application to Android," Proc. 17th ACM Conf. Computer and Communications Security (CCS 10), ACM, 2010, pp. 73-84.

[8] A.D. Schmidt et al., "Detecting Symbian OS Malware through Static Function Call Analysis," *Proc. 4th Int'l Conf.Malicious and Unwanted Software* (Malware 09), IEEE, 2009, pp. 15-22.

[9] M. Egele et al., "PiOS: Detecting Privacy Leaks in iOS Applications,"*Proc. ISOC Network and Distributed System Security Symp.* (NDSS 11), ISOC, 2011; www.iseclab.org/papers/egele-ndss11.pdf.