Detection and prevention of Malware attacks in Android

Anjaneya Murthy Gabbiti
Prashanth Mudhelli
Bhuvan Teja Guddanti

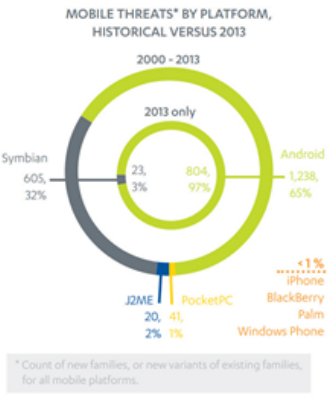
Priyanka Ravikanti
Swarna Viswanathan

Abstract:
In this world people wanted their work to be done easily using smart phones. Mobile phones play a vital role in performing daily activities of individuals. There are many vulnerabilities and malicious attacks targeted on mobiles, so there is a need for an algorithm to detect the malicious software that is being introduced into the system. A new algorithm is to be developed which mainly focuses on detecting the Vulnerability of applications, permissions, shared preferences and critical data present on mobile device. Since Android is an open source platform many third party applications are developed and added to Google Play store without checking the security of the application. Users download the application via APK published or via play store and malicious software enters the mobile device. In this paper we have researched various sources of mobile attacks and identified few of them and proposed solution to few of them by considering SVM data mining techniques by listing various features. The following details describe how the mobiles are used and attacked in current day world. 155% mobile malware increased 2011, 10 Billion Android app downloads reached by 2012 – over which 90% of the top 100 apps have been hacked, 350% increase in WiFi hotspots by 2015, facilitating more opportunities for "man-in-the middle" attacks and 70% of Mobile spam is fraudulent financial services.
Key Words: Android, Malware Detection, Vulnerabilities, Machine Learning in Mobile Security.

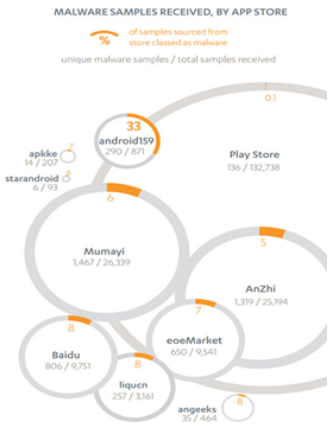
Introduction

In mobile market, Android is among the most popular operating system. Google introduced android in 2005. Since then, it has grown rapidly. According to Gartner report issued in January 2015, over a billion Android devices were shipped in 2014 and this number is expected to grow approximately twenty six percent every year.

Android's runtime is built on Dalvik Virtual Machine, which has architecture similar to JVM. Android includes the core operating system, system utilities and some core/default applications that are shipped with the device. Android developers can develop and submit mobile applications to the Android play store. Apart from Google's official play store, there are third party Android stores to publish Android applications. According a survey conducted by Forbes, about ninety seven percent of the mobile malware is from Android. However, most of the malware is predominantly from the third party market place based out of Asia and Middle East that are unregulated. Below is the distribution of mobile threats by platform.



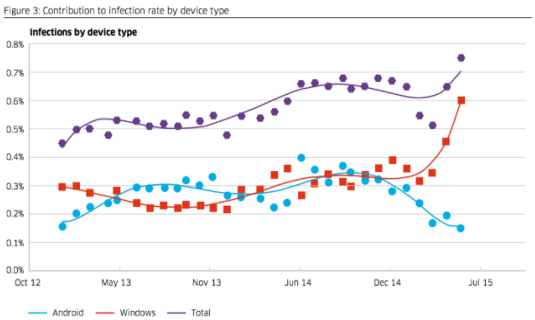
Below is the distribution of malware penetration grouped by app store.



The report also states that one in every thousand applications have a malware for some time in the application lifecycle. E-Markets like AnZhi, Mumayi, eoeMarket, Baidu and liqucn were found to have 5%, 6%, 8%, 8% and 7% malware presence respectively. And a shocking 33% of applications were affected by malware in Android159.

Android's existing security mechanism allows the user to view which resources are required by an application. Currently in Android, if a user wants to install an application, he or she has no choice but to accept the permissions required by the app; Once the user grants these permissions, there is no way to undo them. Android does not allow users to selectively grant permissions to users during installation of apps. Moreover, it also does not provide a way to limit the resource usage based on custom constraints like location of the device, the number of times a resource has been used, etc.

For example, consider a simple use case where an application that gives weather information based on user's current location. The app can have the user location by accessing the location services or by asking the user to manually enter it. For it to automatically read the GPS location, it requires user permissions. As discussed, Android mandates that the application request user permission to read the location information during the install-time. For such a weather app, even though it's not absolutely necessary, the user is forced to give in his or her private data like current location. Denying the permission means that the application cannot be installed. If there is way to implement custom user permissions, we could avoid such situation. In this paper, we discuss some of the ongoing research and projects in the areas of detection and prevention of malware that aims to enhance the end user experience by augmenting Android's current security features and policies. The below image shows how malware attacks have been increased over the period of time among various providers.



Android Security Features and Policies

Android applications make use of local and served data with the help of robust software and hardware. As this data is exposed through the Android platform, it is important that the application platform environment ensures the security of user applications, data, the device, and the user communication over network.

Being an open source platform, Android has a robust security architecture that implements multi-layered security. Multilayer architecture enables flexibility while also providing protection. It has safe defaults implemented for default protection level for beginner level programmers.

Android Security Components

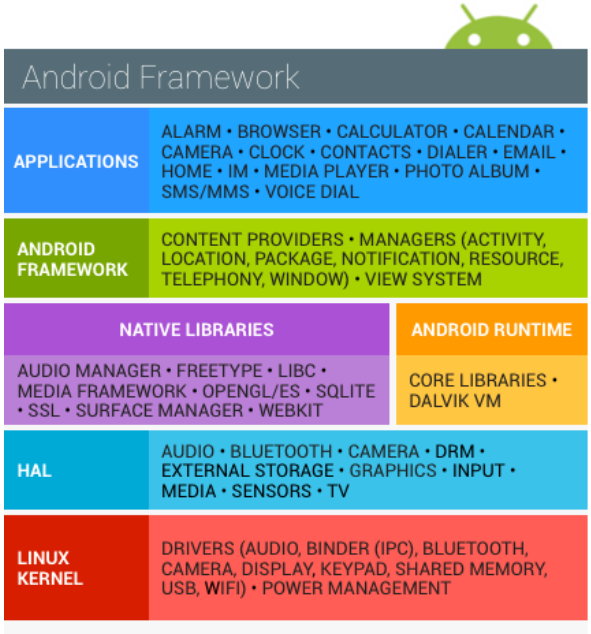


Figure 1: Android Software Stack

Above figure lists the basic security components involved in various levels of the Android's Software Stack. Each component assumes that the components below are secured. Below are the components that form the foundation of Android –

Device Hardware: Android is compatible with a wide range of devices – smart phones, set-top box devices and tablets. Although Android is processor-agnostic, it uses some of the underlying hardware-specific security capabilities.

Android Operating System: Android's core is built on top of the Linux kernel. All of the device resources, such as GPS data, telephony functions, Bluetooth functions, network connections, etc. are accessed through the operating system.

Linux Security

At the bottom of the software stack, Android uses Linux kernel. Linux kernel provides Android with the Linux security features along with the Inter

Process Communication (IPC) – for secure communication between applications. Moreover, each application runs in its own sandbox. All these features ensure the system's security at the OS level.

The Linux kernel provides several key security features to Android –

- Process isolation
- A permissions model that is user-based
- Secure IPC using extensible mechanism
- The ability to remove potentially insecure parts of the kernel

Application Sandbox

Android uses the idea of "Application Sandbox" to provide a secure and restricted runtime for applications. Only a small piece of code is run as root and the sandbox restricts all the code that lies above the Linux kernel. Android's runtime is based on Dalvik Virtual Machine. Every application runs in its own DVM, this contained environment is called the Application Sandbox. In other words, each application gets a dedicated part of the filesystem which can be used to write private data. The Linux kernel enforces security at the process level for the system and applications. This makes use of Linux constructs such as user and group IDs assigned to applications. By default, applications have restricted access to the OS and cannot communicate with each other. For instance, application A cannot read application B's data without proper permissions (which is a separate application). Even if it tries to do so, the operating system protects against this.

User Security Features

Android supports the following user security features –

File system Encryption

Android 3.0 and later supports full file system encryption, so all user data can be encrypted. The encryption key is protected using sophisticated cryptographic techniques. To prevent attacks, Android also provides password rules that can be set by the user or device administrator. These rules are enforced by the operating system.

Password Protection

Android also provides a way of securing the device using a user-supplied password. This password also protects the cryptographic key for file system encryption.

Device Administration

Android 2.2 and later versions provide Device Administration APIs that allow device administration at the system level. For example, default email application in Android uses these APIs to improve Exchange support. Using this application, email administrators can enforce password policies across devices. Administrators can also implement policies like remotely erasing data in case of lost or stolen handsets. These APIs are also available to third party device management providers.

Android Application Security

By default, an Android application can access a limited set of system resources. Underlying OS manages access to resources. So, if used there is a misuse of resource by an application, it throws a security exception. These restrictions are implemented with the help of APIs and permissions

These protected APIs include the following functions-

- Camera functions
- Location related functions (GPS)
- Bluetooth functions
- Telephone functions
- SMS/MMS functions
- Network/data connections

As these resources are managed by the operating system, an application must ask or declare for permissions in the manifest file. Thus, during the app installation, the system asks a user to accept/reject the permissions requested by the app. Once these permissions are granted to the app, they remain valid as long as the app is installed. Permissions are removed when the app is uninstalled.

Inter Process Communication (IPC)

Processes can communicate with each other using Inter Process Communication. IPC is enabled with the use of following components –

Binder: It is a high performance remote procedure call mechanism designed for inter and intra process communication. A custom Linux driver is used to implement Binder.

Services: Services provide with interfaces that can be directly accessed using binder.

Intents: Android Intents are simple message objects that represent an "intention" to do something. For instance, if an app wants to display a page, it will express its "Intent" to view the URL by creating an instance of Intent and passing it to the system. The system then passes this request to the Browser to handle that Intent.

Content Providers: Content Provider is a database like structure that enables data access on the device; Contact list is a classic example of a Content Provider. An application can expose its data using Content Provider for use by other applications.

Apart from these, Android also provides the following to ensure security –

- System Partition and Safe Mode
- File system Permissions
- Security-Enhanced Linux
- Full disk encryption
- Rooting of Devices
- Verified Boot
- Trusted Execution Environment
- Authentication using fingerprints
- Hardware-backed key store

Detection of Malware:

Signature based detection: Signature based detection methods uses static and dynamic models to identify the malware in mobiles. Static analysis targets mainly on considering the application source code without actually running the application. Dynamic analysis includes calculating the memory leaks, network traffic leaks, data flow during the application running. Drawback: Requires huge device in memory to determine the malware.

Behaviour-based detection: This method uses predetermined attack patterns and compares the process behaviour that occur in a system to determine the attacks. Researchers have identified areas such as memory usage, SMS content and battery consumption to identify the threats. So this can be done in two ways, statically and dynamically. Statically includes analysis inside the mobile device while dynamically analysis includes transferring the statistics and performance data over network to a server which processes the data based on machine learning algorithms and responses result.
Dynamic based is another model of study which processes the flow of data.

Android manifest file has a lot of information about each application and manifests of all the applications are stored in a common file and features can be extracted and used to predict the malware. Below are the features that needs to be concentrated:

Hardware components:
All the application which requires access to GPS, camera, network, location settings are declared in a manifest file. Reading these settings will help in determining malware apps which sends network and location data to attacker.

Requested Permissions:
Another important security mechanism is security permission system. Permissions are granted by user when launching the application. So some applications requests permissions more frequently and some of them request SEND_SMS permissions by which attacker gains access to all the phone data. So this feature is also important in identifying.

App components data:
There are various application components that are used in the android app development such as services, Intents, Broadcast Receivers, Activities, content providers. All these are also extracted from manifest file and some malicious applications uses common names which can be identified from this files.

Filtered Intents:
Android application uses Intents to perform inter and intra process communication. These are also main source for malicious activity as some manifests files will have malicious triggers when launched. A typical example of an intent in malware is BOOT COMPLETED message, which triggers malicious activity as soon as you reboot the smartphone.

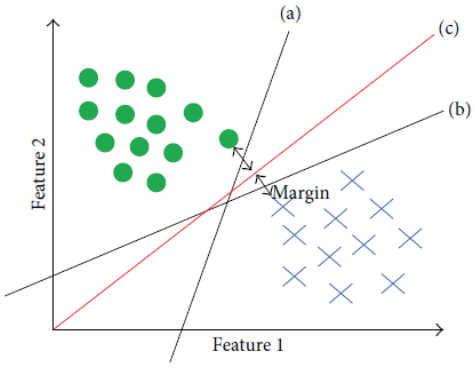
Features extracted from APK – Disassembled code:
Suspicious API Calls from Code:
There are certain API calls which gets the sensitive data from the system. There are some common APIs that needs to be restricted while deploying the application they are as follows:

- API calls for accessing sensitive data, such as getDeviceId() and getSubscriberId().
- API calls for communicating over the network, for example setWifiEnabled() and execHttpRequest() API calls for sending and receiving SMS messages, such as sendTextMessage()

- API calls for execution of external commands like Runtime.exec()
- API calls frequently used for obfuscation, such as Cipher.getInstance()

Network Addresses and Used Permissions:
In the code please check for permissions used as some applications might use a lot of other permissions apart from that actually the app uses. Thus these needs to be monitored. Hardcoded URL and URIs in code needs to be verified as some might use these for malicious purposes as they post the application information to the attacker. Hence these are also needs to be monitored as a part of static analysis.

Malware Detection Using Classifiers:
We have researched extensively on various methods which are required to detect the malware using various data mining techniques. We have researched various data mining algorithms such as DT (decision tree), BN (Bayesian networks), NB (naive Bayesian), Random forest, and SVM (support vector machine). So all these models consider the behaviour based detection.
SVM is one of the machine learning classifiers that is used to detect malware in android device because of its high performance. The SVM also solves the problem of nonlinear data i.e. not binary and varying data, In this method hyperplanes are constructed by considering various features of the android mobile and a hyperplane is selected which has high margin compared to others. Some of the features that are identified are Network data, telephone data, message data, CPU data, Battery data (temperature, voltage), processes data (PID, Process name, App Name). Finally hyper planes are constructed and if any new mobile data is beyond this hyperplane data we categorize that application is responsible for malware and alert the user.



Prevention Measures:

Cloud services are used to prevent malware in android. Cloud infrastructure is used to detect malware continuously and update database by performing complex analysis on the data gathered. Pre Crime malware prevention system is used to predict and compare mobile devices with the next event to check whether the next event is abnormal or normal. Monitoring consumption of power, increasing diversity of platforms and hardware enforcement and box can prevent malware.

Researchers make use of static analysis, dynamic analysis or both to detect and prevent malware. Aubrey Derrick Schmidt gathers the names of the calls and functions featured at the output and next responses, and then uses certain methods for analysis. Monitoring method can be used to detect and prevent mobile malware by observing the electric power of the mobile device and history of energy consumption, as mobile malware open Channels which consumes mobile power, such as Bluetooth and Wi-Fi. Overall, researchers use four methods to prevent mobile malware such as static and dynamic analysis Cloud service, Pre crime and power consumption observation.[1]

Dynamic analysis can detect and prevent malware but it consumes more space. The Pre Crime cloud service scheme can detect malware but cannot do both malware prevention and increased performance of mobile.[2]

The combination of dynamic analysis method and the Pre Crime cloud service can be focused to overcome both methods weaknesses. Storing the result of dynamic analysis in cloud solves the storage paradox in dynamic analysis and predicts the mobile behavior using cloud, as compared to the dynamic analysis result. The proposed method can detect, prevent and provide storage space to dynamic analysis and use the cloud service.

Incorporating security enhancements at system level:

- Native Code Execution at a stricter controlling level
- Enhancing Anti virus performances in the System through interface
- Native Code Hash and Signature Validation.[3]

Android Security Extension - It provides access control for apps at fine grain level. It uses a

policy-based model, which is a significant preventive measure. It enhances Android permission model by providing control over what kind of contacts an app can access, which sites an app can connect over the Internet. In order to fight the complexity of malwares in future, instead of using the conventional methods, antimalware solutions may have to evolve and adapt multiple countermeasures in a hybrid approach. [4]
Avoiding installation of suspicious or pirated apps is very important. One of the best methods to avoid malware is to notice incidents like suspicious apps with bad reviews, illegitimate app permissions and outrageous promises. Make sure to only install or download apps from a trusted developer's Web site.

There are numerous settings provided by Google for the Android operating system which can prevent malware attacks and other malicious corruption. All the Android Devices running 2.2 or higher have complete access to Google's malware scanner. If we downloaded any app outside play store, Google would scan the app and warn the user of any potential threats prior to installing an application. By default, this feature is enabled and can be accessed from the Google Settings app in the app drawer of the device. On the other hand, in devices that are running Android 4.2 or higher, user will be able to access the feature by clicking Settings, go on Security, and scroll down to the option Verify apps.

Other preventive measures users must take:

Avoiding downloading apps from unauthorized or illegitimate apps stores, protecting phone with passwords, avoiding sharing sensitive personal information in the public Wi-Fi, choosing best antivirus for the phone(ESET, F-Secure, Kasper, McAfee, Norton, Trend Micro), reading and understanding the permissions before downloading any app, downloading apps that are scanned through Bouncer (or any malware scanner available in android market), prohibit providing root access to any app or internet provider, careful while passing intents. Avoiding pass confidential data through intents. All these measures enhance protection against malware.

Conclusion:

In this paper we have discussed various security features in android and threats malware is posing to android users. We have included in detail the ways of detecting malware and preventing them. We have analyzed the latest research to come up with novel techniques for malware detection that can be used in near future. We have also identified important enhancements in system level for the Android devices as well as innovative measures that can be used for battling against advanced malware attacks. We have proposed few effective solutions to malware prevention using various cloud and data mining techniques. We hope to build a application for detecting and preventing malware and privacy impeachment in the near future by leveraging the various strategies and methodologies discussed in this paper.

References:

[1] Abdullah Mohammed Rashid & Ali Taha Al-Oqaily. “Detect and Prevent the Mobile Malware” - International Journal of Scientific and Research Publications 2015

[2] Nadji, Y., Giffin, J., & Traynor, P. (2011, December). “Automated remote repair for mobile malware”. In Proceedings of the 27th Annual Computer Security Applications Conference ACM.

[3] Rahul Raveendranath, Venkiteswaran R, Anoop Joseph Babu. “Android Malware Attacks and Countermeasures: Current and Future Directions” .

[4] Russello, Giovanni et al. "Yaase: Yet another android security extension" - 2011 ieee third international conference.

[5] Ozdemir, M., & Sogukpinar, I. (2014). An Android Malware detection architecture based on ensemble learning. *Transactions on Machine Learning and Artificial Intelligence*, *2*(3), 90–106. doi:10.14738/tmlai.23.261

[6] Zou, S., Zhang, J., & Lin, X. (2014). An effective behavior-based Android malware detection system.*Security and Communication Networks*, *8*(12), 2079–2089. doi:10.1002/sec.1155