International Journal of Advance Research in Science and Engineering
Volume No.07, Issue No.04, April 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

INTEGRATIVE APP MISBEHAVIORAL CHECK AND DEVELOPMENT OF EFFECTIVE QUARANTINE SYSTEM FOR VIRUS DETECTION

Dr. S. Ramamoorthy[1], Ms. R. Poorvadevi [2]

[1]*Assistant Professor, CSE Department, SCSVMV, Enathur, Kanchipuram – 631561, (India)*
[2]*Assistant Professor, CSE Department, SCSVMV, Enathur, Kanchipuram-631561, (India)*

ABSTRACT

In the domain specific environment, unused Application would do background activities have significant negative impacts on the user, e.g., leaking private information or significantly taxing resources such as the battery or network. This is no system to deactivate these Zombie Apps. In the proposed System, it seeks to facilitate effective identification and subsequent quarantine of such zombie apps towards stopping their undesired activities. The implementation is, first of all it must be need to design multiple applications and also create an Anti virus like Application so as to monitor. The virus behaviors are like, Sending SMS, Storage of call logs in the Cloud server, Crashing the Gallery, Drain of Battery, Slowness of Mobile by reducing the RAM. All these misbehavior activities are monitored and quarantined successfully with the acknowledgement of the user by our Application.

Keywords:Zapdroid, App Scanning, Network Service Provider, App misbehavior activity, Zombies data, Spoofed network.

I.INTRODUCTION

Statistics indicate that for a typical app, less than half of the people who downloaded it use it more than once; further 15 percent of the users never delete a single app that they download. In more general cases, users may only interact with some downloaded apps infrequently (i.e., not use them for prolonged periods). These apps, which are seemingly considered dead by the user, continue to operate in the background long after the user has stopped interacting with them. Such background activities have significant negative impacts on the user, leaking private information or significantly taxing resources such as the battery or network. Unfortunately, the user is completely unaware of these activities. We call such apps, which are dead from the perspective of the user, but indulge in undesired activities, as "zombie apps".

In this paper, main idea is seek to facilitate effective identification and subsequent quarantine of such zombie apps towards stopping their undesired activities. Since a user can change her mind about whether or not she wants to use an app, a zombie app must be restorable as quickly as possible if the user so chooses. The classification of an app as a zombie app is inherently subjective. After an app goes unused by a user for a prolonged period, the determination of whether the app should be constrained depends on whether the app's resource usage during the period of unused is considered significant or whether the app's access of private data is deemed serious.

Therefore, instead of automatically categorizing apps as zombie apps, we seek to empower the user by exporting the information that she would need to make this decision. Moreover, the manner in which a zombie app should be quarantined depends on whether the user is likely to want to use the app again in the future (e.g., a gaming app that the user tried once and decided is not interesting versus a VoIP app that the user uses infrequently). The apps that a user is likely to use again fairly soon should not be fully uninstalled; real time restoration (when needed) may be difficult if the user does not have good network connectivity.

It seeks to enable users to deal with these different scenarios appropriately. First, zombie apps are active (execute) in the background and hence, it needs an efficient mechanism to track the foreground and background states of apps; continuous monitoring of apps, as proposed in prior approaches , can be too resource-intensive to be practical. Second, it needs to monitor how apps use sensitive resources protected by permissions in a lightweight manner.

Application-level implementations are infeasible since Android does not allow one application to track the permission access patterns of other apps. Third, once a zombie app is quarantined it must ensure that it is not re-activated unless the user chooses to do so. With current approaches, the background activity of apps is constrained only temporarily, until they are woken up due to time-outs or external stimuli. Fourth, ZapDroid should ensure that a previously-quarantined zombie app is restored quickly if the user seeks to access it; the restored app must be in the same state that it was in, prior to the quarantine.

II.EARLIER APPROACHES

In the existing system, unused Application would do background activities have significant negative impacts on the user, e.g., leaking private information or significantly taxing resources such as the battery or network. This is no system to deactivate these Zombie Apps. In previous cases, an application has to verify its own security credentials and analyze the components based access specific factor to induce the application level processing in the clients environment. The service specific access can be automatically creating the security patches to validate the user level access privileges to safeguard the user data. Still, there is an issue in the existing scenarios to protect the application security components.

Limitations of the Earlier Approaches:

- Users are affected by this type of virus.
- No application to stop those viruses.
- No specific tools/ techniques can be adapted in the virus scanning operation.
- Nature of viruses needs to be identified and the access locations are to be finding out in the service access platform.

III.PROPOSED SYSTEM

In the proposed system, we seek to facilitate effective identification and subsequent quarantine of such zombie apps towards stopping their undesired activities. The malicious activities are to be iterated and restricted in the client platform. This mechanism will easily find the occurrences of errors by the form of virus scanning operations. This proposed mechanism will leads to user to freely use the application form the malicious application processed.

Merits:

- ❖ Avoid Data leakage
- ❖ Reliable
- ❖ High data transmission rate
- ❖ Provide antivirus

System Architecture:

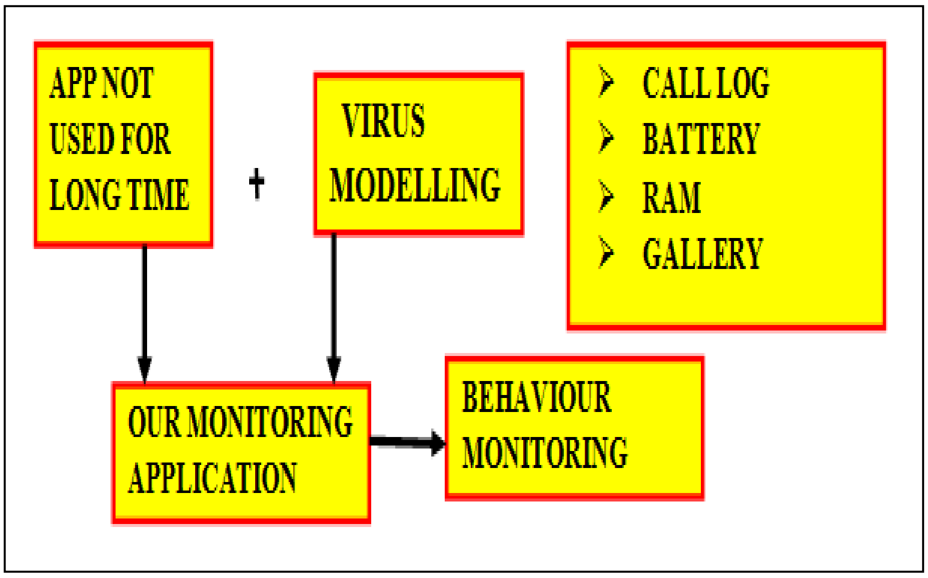


Fig:3.1 Illustration of System Framework

An above system framework will depicts the functional components of the app behavior modeling which specifies the operational sequences of virus modeling and monitoring call log reports.

IV.IMPLEMENTATION WORK

In this work, the implementation is, first of all it design multiple applications and also create an Anti virus like Application so as to monitor. The virus behaviors are like, Sending SMS, Storage of call logs in the Cloud server, Crashing the Gallery, Drain of Battery, Slowness of Mobile by reducing the RAM. All these misbehavior activities are monitored and quarantined successfully with the acknowledgement of the user by our Application. If any application is draining the battery level those application will shows on user mobile and it will be automatically uninstalled from user mobile and send it to recycle bin. User can install that application whenever they need.

Phases of the Work:

Parts of the implementation work are given below:

- Android Deployment
- Server
- Modelling Virus
- Monitoring Call Logs & Crashing Gallery
- Virus Propagation
- Distribution Of Patches
- Cloud & Infrequent App Access

Android deployment:

Mobile Client is an Android application which created and installed in the User’s Android Mobile Phone, so that we can perform the activities. The Application First Page Consist of the User registration Process. We’ll create the User Login Page by Button and Text Field Class in the Android. While creating the Android Application, we have to design the page by dragging the tools like Button, Text field, and Radio Button. Once we designed the page we have to write the codes for each. Once we create the full mobile application, it will generated as Android Platform Kit (APK) file. This APK file will be installed in the User’s Mobile Phone an Application**.**

Server:

The Server is Server Application which is used to communicate with the Mobile Clients. The Server can communicate with their Mobile Client by GPRS Technology. The Server Application can be created using Java Programming Languages. The Server will monitor the Mobile Client’s accessing information and Respond to Client’s Requested Information. The Server will not allow the Unauthorized User from entering into the

Network so that, proposed system can provide the network from illegitimate user's activities. Also the Server will identify the Malicious Nodes activities.

Modeling virus:

In this Module, create the Mobile Virus which is malicious code that will perform malicious activities in the User's Mobile Phones. In this project create a New Folder Virus which will create a Folder inside the Folder virus by developing malicious codes. So that can generate the Mobile Virus. Once the attackers created the Virus, they will spread the Virus via Bluetooth or SMS technique, So that the virus will be spread to other Users Mobile Phones. While sending via Bluetooth technique, the User's has to be present within the communication range. The Attacker can send the virus file via Mobile Application that was installed in their Mobile Phones.

Monitoring Call Logs & Crashing Gallery:

In this module, create the mobile virus and spread the mobiles, call logs are stored in cloud server like missed call and dialled calls. Also spread the gallery so all data's are changed in encrypted format. So does not view the gallery.

Virus propagation:

Once the attack spread the Virus File to other User's Mobile Phone the content of the message of the file will be analyzed by the Server to detect whether the file contains that Malicious Behaviour or not. If the file contains the malicious behaviour, then the Server will detect the file as Virus file. Once the Server detected that the Virus file it will deliver the patches to the User's Mobile Phone and deletes the Virus File.

Distribution of patches:

Once the Server identify virus file was sent to the User's Mobile Phone, the Server will provide the patch files to delete the Virus file. Using an Android Application the patches will be distributed to the User's Mobile phone automatically to clear the Virus. User doesn't want to update their antivirus.

Cloud & infrequent app access:

- Cloud storage is a service model in which data is maintained, managed, backed up remotely and made available to users over a network. The contacts will be automatically backed up to cloud.

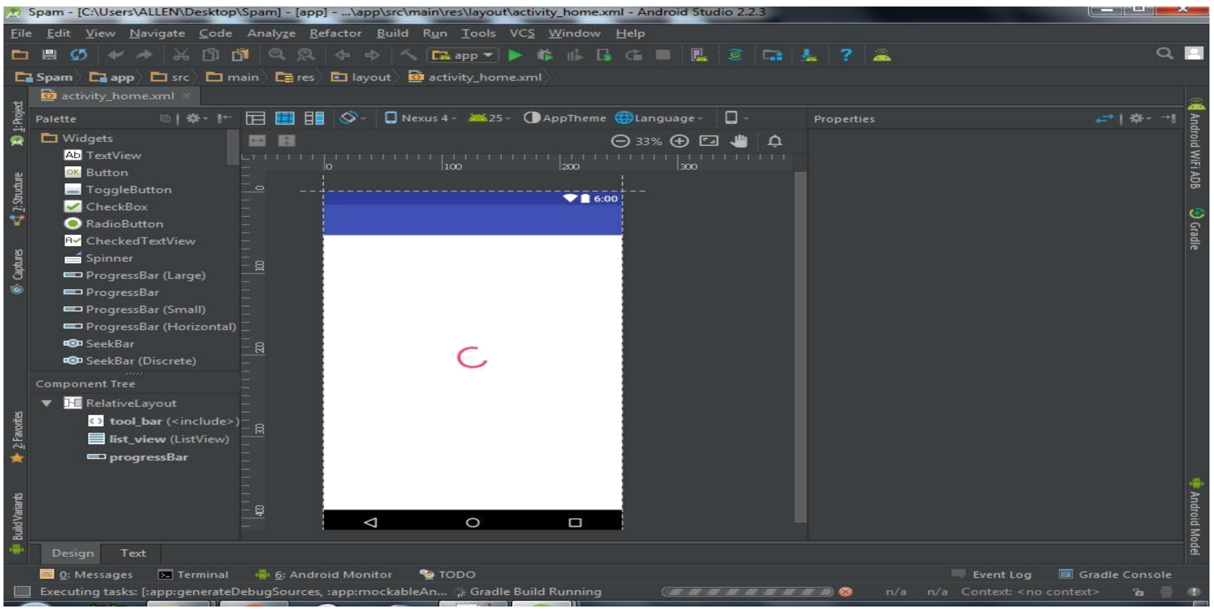
- Another one is people is not using all apps in mobile so that mobile memory will be wasted for unwanted application. So we are showing that infrequent application to the user to uninstall it. If they give permission it will be uninstalled and through it to recycle bin whenever we want app we can install it from recycle bin.

So, from the implementation work all the function specific components have been demonstrated with the suitable system requirements and obtained the result. The virus modeling and scanning operations are developed and determined the functional process of creating the spam application and the unused application.

V. EXPERIMENTAL RESULTS

Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing, a set of steps into which we an place specific test case design techniques and testing methods should be defined for software process. Testing often accounts for more effort than any other software engineering activity. If it is conducted haphazardly, time is wasted, unnecessary effort is expanded, and even worse, errors sneak through undetected. It would therefore seem reasonable to establish a systematic strategy for testing software.

Experimental results are iterated and executed in the cloud storage access platform to examine the service generic based application to check the occurrences of viruses and to determine the app misbehavior check in the android app development platform.



Spam - [C:\Users\ALLEN\Desktop\Spam] - [app] - ...\app\src\main\res\layout\activity_home.xml - Android Studio 2.2.3

Executing tasks: [:app:generateDebugSources, :app:mockableAn... Gradle Build Running

Fig:2 Process of Creating Spam Application

From the resultant value, the service has been give to the user location to freely use the virus free resources and also creating the application specific environment. The virus scanning and detection process is used to check the misbehaviour activities of the client applications and process the unconventional elements to find out the occurrences of defining the security and vulnerable patches over the client service access platform.

The following graph illustrates the functional process of how the virus scanning and modelling has been developed and the results are well optimized with the android app development environment and analyzing the process of misbehaving applications and services in the client level service entry need to effectively produce the QoS value to improve the agility of business users applications.

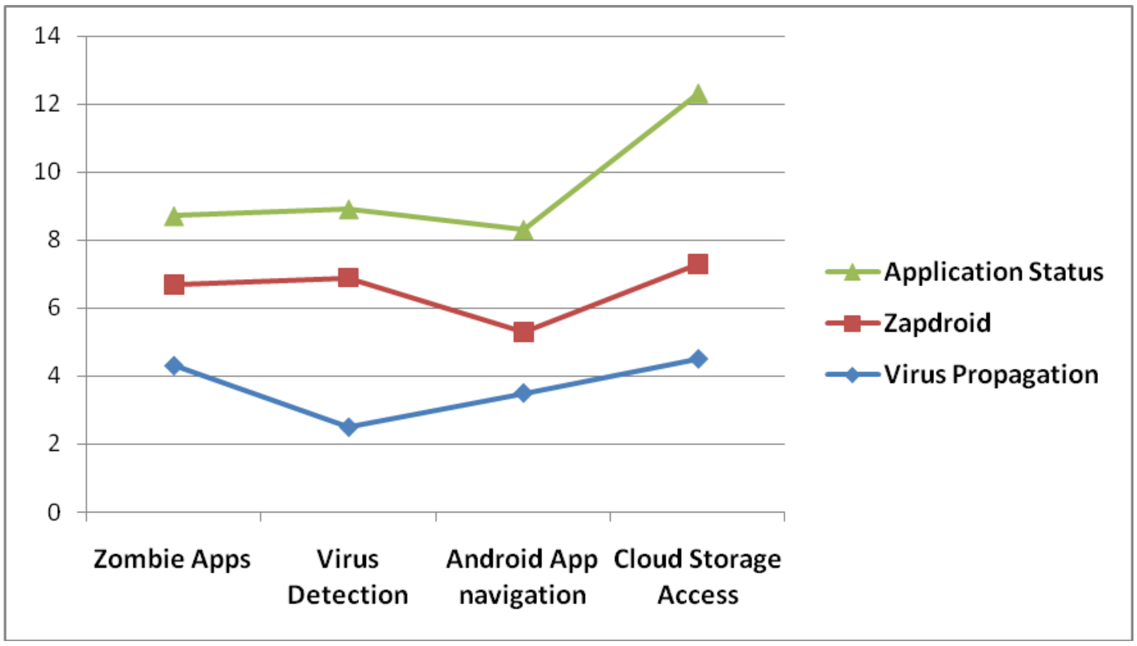


Fig: 3 Depicting the Process Outcome for Virus Detection Approach

The schematic process will illustrates the operational sequences of how to optimize the process based outcomes from figure 3. This mechanism is used to detect the outcome based functional segments and also analyzing the effective viruses and checking the desired solution in the virus detection platforms to specify the operations

VI.CONCLUSION

The malicious apps threaten the user data privacy, money and device integrity, and are difficult to detect since they apparently behave as genuine apps bringing no harm. This paper has implemented a multi-level host-based malware detector for Android devices. Once we detect mobile abnormal activity means automatically distributing the patches.

REFERENCES

[1] Milang et.al, “Global mobile statistics 2014 part a: Mobile subscribers; handset market share;mobile operators,” http://mobiforge.com/ research-analysis/global-mobile-statistics-2014- part-a-mobilesubscribers- handset-market-share-mobile-operators, 2014.

[2] A. Reina, A. Fattori, and L. Cavallaro, "A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors," EuroSec,April, 2013.

[3] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. Sadeghi, and B. Shastry,"Towards taming privilege-escalation attacks on android," in 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012, 2012.

[4] M. Backes, S. Gerling, C. Hammer, M. Maffei, and P. von Styp- Rekowsky, "Appguard fine-grained policy enforcement for untrusted android applications," in Data Privacy Management and Autonomous Spontaneous Security, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 213–231.

[5] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming information-stealing smartphone applications (on android)," in Proceedings of the 4th International Conference on Trust and Trustworthy Computing, ser. TRUST'11. Berlin, Heidelberg: Springer-Verlag, 2011,pp. 93–107. [Online]. Available: http: /dl.acm.org/citation.cfm?id=2022245.2022255.

[6] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924971

[7] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry, "Practical and lightweight domain isolation on android," in Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, ser. SPSM '11. New York, NY,

[8] Jang, "User Access permissions: user attention, comprehension, and behavior," in Symposium On Usable Privacy and Security, SOUPS '12, Washington, DC, USA - July 11 - 13, 2012, 2012, p. 3.

[9] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in Proceedings of the 2012 IEEE Symposium on Security and Privacy, ser. SP '12. Washington,DC, USA: IEEE Computer Society, 2012, pp. 95–109. [Online].Available: http://dx.doi.org/10.1109/SP.2012.16

[10] K. S. Labs, "Kindsight security labs malware report h1 2014,[Online]. Available: http://resources.alcatel-lucent.com/ ?cid=180437 1545-5971 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.