MalAnalytics: A Signature based Analytic System to Collect, Extract, Analyze and Associate Android Malware

PRATIK JUNNARKAR[1], AYUSHI SORTE[2]

***

Abstract: Smartphones are quickly turning out to be key gadgets for some clients. Lamentably, they likewise become fruitful justification for programmers to send malware. There is a critical need to have a "security logical and measurable framework" which can encourage investigators to analyze, dismember, partner and connect enormous number of portable applications. A powerful explanatory framework needs to address the accompanying inquiries:

How to naturally gather and deal with a high volume of portable malware?

How to break down a zero-day dubious application, and contrast or partner it and existing malware families in the database?

In this paper, we present the plan and execution of Mal Analytics, a mark based scientific framework to consequently gather, oversee, examine and remove android malware.

Introduction:

Cell phones are turning out to be winning gadgets for some individuals. Tragically, malware on cell phones is likewise expanding at an exceptional rate. Android OS-based frameworks, being the most famous stage for cell phones, have been a famous objective for malware engineers.Inspiteofthefactthatthereare number of works which center around Android malware location by means of authorization spillage, it is similarly critical to plan a framework that can perform thorough malware examination: breakdown and dismember dubious applications at the opcode level (rather at the consent level), and connect the application with existing malware in the database to decide if it is changed malware or zero-day malware, just as which real applications are tainted.

Challenges:

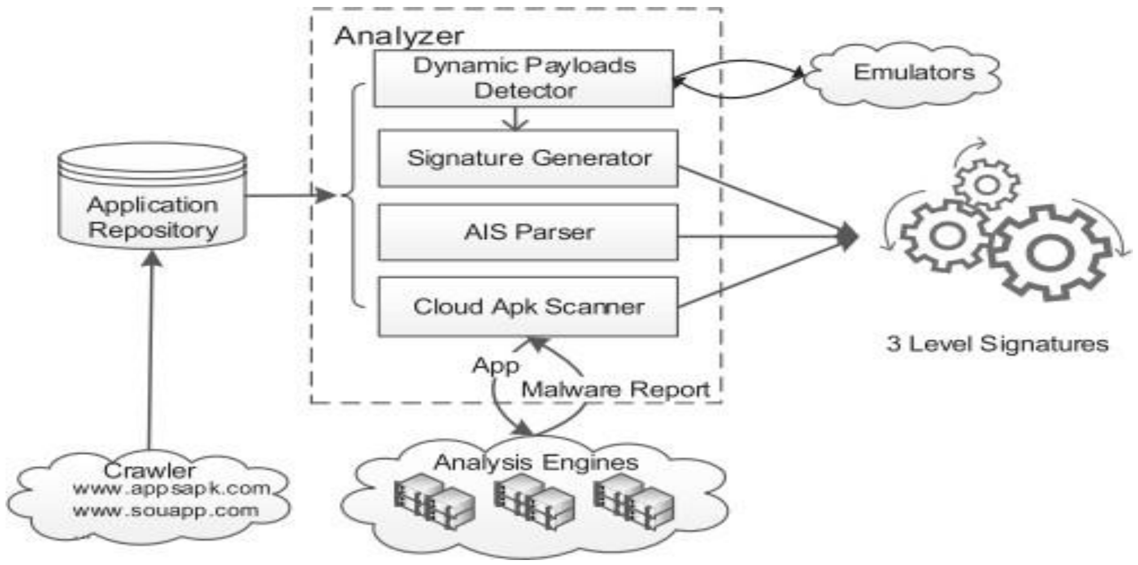
To understand a compelling investigative framework for Android portable applications, we have to beat a few specialized obstacles.

In the first place, how to efficiently gather malware from nature.

The subsequent obstacle is the manner by which to distinguish repackaged applications (or changed malware) from the huge expanse of uses and malware.

The third obstacle is the way to relate a malware with existing malware (or application) to encourage security examination.

Design & Implementation of MalAnalytics :



TheArchitectureofMalAnalytics Building Blocks of MalAnalytics :

Extensible Crawler: Users can indicate official or outsider commercial centers, just as blog destinations and the crawler will perform ordinary portable application download. The crawler empowers us to efficiently develop the versatile applications database for malware investigation andaffiliation.

Dynamic Payloads Detector: To manage malware which powerfully downloads noxious codes bymeansoftheInternetorconnection records,wehaveexecutedthedynamicpayloads findersegment,which decides malignant trigger code inside malware bundles and tracksthe downloaded application and its conduct in virtual machine.

Android App Information (AIS) Parser: AIS is an information structure inside MalAnalytics and it is utilized to speak to .apk data structure. Utilizing the AIS parser, experts can uncover the cryptographic hash (or other essential mark) of an .apk record, its bundle name, authorization data,communicaterecipientdataanddismantledcode…andsoon.

SignatureGenerator:Anti-Virusorganizationsasaruleutilizecryptographichash,e.g.,MD5, to create a mark for an application. This has two significant downsides.

Rightoffthebat,programmers can undoubtedly transform an application and change its cryptographic hash.

Also, the cryptographic hash doesn't give adequate adaptability to security investigation. In MalAnalytics, we utilize a three-level markageplantorecognizeeveryapplication.

(a) Android API calls table: Our framework utilizes the API calls table of the Android SDK. The android.jar document is the system bundlegavebytheAndroidSDK.WeutilizetheJava reflection to acquire all portrayals of the API calls.

(b) Disassembling process: Each Android application is made out of various classes and each class is made out of various strategies. To create marks for each class or strategy, MalAnalytics first dismantles an .apk document, at that point takes the Dalvik opcodes ofthe

.dexrecordandchangesthemtostrategiesandclasses.AtthatpointMalAnalyticsutilizesthe Android API calls table to create marks.

(c) Generate Lev3 signature (or strategy signature): The framework initially creates a signature for every technique and we consider this the Lev3 signature. In view of the Android API calls table, the framework separates the API call ID arrangement as a stringinevery strategy,atthatpointhashesthisstringanincentivetodeliverthetechnique'ssignature.

(d) Generate Lev2 signature (either class signature or dynamic payload signature): Next, MalAnalytics continues to create the Lev2 signature for each class, and it depends on the Lev3signatureoftechniquesinsidethatclass.Malwarejournalistsmayutilize different obscurityorrepackagingprocedurestochangethecallingrequestofthetechniquestablein a.dexrecord.Tobeatthisissue, ourmarkagecalculationwillinitiallysortthelevel3 signatureinsidethatclass,andafterwardconnectalltheselevel3signatureto shapethe level 2signature.

(e) GenerateLev1signature(orapplicationsignature): Malware scholars may utilize some repackaging or confusion strategies to change the request for the classes table of the .dex document, our signature calculation will initially sort all Lev2 signature, at that point link these Lev2 signature to create the Lev1 signature.

Conclusion:

We present MalAnalytics, an Android malware expository framework which can naturally gather malware, create marks for applications, distinguish vindictive code section (even at the opcode level), and simultaneously, partner the malware under investigation with different malware and applications in the database. Our mark philosophy gives critical points of interest over conventional cryptographic hash like MD5-based mark. We tell the best way to utilize MalAnalytics to rapidly recover, relate and uncover vindictive rationales. Utilizing the consent recursion procedure and class affiliation, we tell the best way to recover the authorizations of strategies, classes and application (as opposed to essential bundle data), and partner all applications in the opcode level. Utilizing MalAnalytics, one can without much of a stretch find repackaged applications by means of the closeness score.

REFERENCES:

1. McAfee, "McAfee Threats Report: Fourth Quarter 2011," Tech. Rep., 2012.
2. Contagio Mobile. [Online]. Available: http://contagiominidump.blogspot.com/
3. M. Zheng, P. P. C. Lee, and J. C. S. Lui, "ADAM: An AutomaticandExtensiblePlatformtoStressTestAndroidAnti-VirusSystems," inProceedings of the 9th Conference on DIMVA, 2012.
4. Scrapy. [Online]. Available:http://www.scrapy.org
5. P. Felt, H. Wang, A. Moshchuk, S. Hanna, and E. Chin, "Permission Re-Delegation: Attacks and Defenses," in Pro- ceedings of the 20thUSENIXconferenceonSecurity,2011.
6. M.Zheng, M.Sun, and J.C.S.Lui, "Droidanalytics: A signature based analytic system to collect, extract, analyze and associate android malware," Tech. Rep. [Online]. Available: http://arxiv.org/abs/1302.7212
7. G. McCluskey. (1998) Using Java Reflection. [Online]. Available: http://docs.oracle.com/javase/tutorial/reflect/index
8. GoogleInc.GooglePlayStore.[Online].Available: https://play.google.com/store
9. AndroidPIT. [Online]. Available:http://www.androidpit.ru