BEHAVIOR-BASED ANDROID MALWARE DETECTION AND PREVENTION

Jalaj Pachouly[1], Prof. Varsha Dange[2]
[1,2]Department of Computer Engineering, Dhole Patil College of Engineering Pune

Abstract
As seen in last five years use of mobile devices and tablets grown to manifold and ratio between the mobile computing device to human being already cross the 100 percent, it means that their are more mobile computing device than the human being, which essentially means that per person , their are multiple devices. As we know that Android share the 98 percent stack as a computing platform specially for mobile computing, in general when talked about the smart phones and tablets, hence at the same time, android platform attract the more than 90 percent cyber crime. Android do have security mechanism specifically designed for controlling the permission and for App isolation, but that is not enough to prevent and detect the Nobel security attacks carried out on the mobile devices, which creates serious consequence like identity theft, data leak, ransom-ware and compromise in privacy etc. Hence to provide the viable solution for this situation, it is required to deeply analyze the misbehavior's by deployed application to conclude them as Malicious or genuine. MADAM (MultiLevel Anomaly Detector for Android Malware) is an attempt to provide the prevention and detection to ensure the end user safety, which basically inspect the deployed app for all possible misbehavior's at various level of application execution on android platform, mainly kernel, application ,user, package and classify the application as malicious or genuine using the user interface where user is informed about the Malicious app, and user can uninstall and revoke the permission of the application which it should not have.

Index Terms: Android security, intrusion detection system, malware, classification.

I. INTRODUCTION

Android being the popular platform, is the preferred target for security attacks,which theratens the confidence while moving business over mobile computing. Their are various category of malwares based on the technology and kind of their working and activation. Such malware can subcribe a user for high premium unwanted services, costly phone calls and sms, leaking the user location, contact information, transaction information, privacy even without noticed by the user Or when it is quite late and damage is already done in terms of financial, socail or in other terms.Following are the few malware categories of malware mostly found responsible for most of the security attacks- :

- Botnet- a network of private computers infected with malicious software and controlled as a group without the owners' knowledge, e.g. to send spam
- Rootkit- a set of software tools that enable an unauthorized user to gain control of a computer system without being detected
- SMS Trojan-These Trojans use the SMS (text) messaging services of a mobile device to send and intercept messages. The user is usually unaware of the behavior.
- Spyware-software that enables a user to obtain covert information about another computer activities by transmitting data covertly from their hard drive.
- Installer-a piece of software that installs a program on a device
- Ransom-ware-a type of malicious software designed to block access to a computer system until a sum of money is paid.
- Trojan-It is a type of computer software that is camouflaged in the form of regular software such as utilities, games and sometimes even anti virus programs

There are a set of behaviors which can classify an application to be malicious or genuine, can be leveraged to detect and classify an application. More than 90 percent malicious software found exposes one or more of such behaviors.

A. *Text messages sent by a non-default message app.*

Every smart phone has one default sms sending program, if some other application are sending the SMS, then such app can be suspicious..

B. *Text messages sent to numbers not in the user contact list.*

This is the case where the Malware hijacks the default SMS program and send premium SMS to numbers which are not in the user contact list, and result in loss of money, or register user to unwanted services, and charges money. MADAM notices this behavior and put such apps in suspicious app list.

C. *High number of outgoing message per period of time.*

This kind of behavior creates SPAM, where it sends same message to all the contacts, usually with objectionable, or material with having promotions for programs without the users' interventions.MADAM here checks the ratio, something like message per period, which is configurable, and if it reaches the thresh-hold, misbehavior is detected.

D. *High number of process per app.*

This is one of the trick most malware uses to crash the platform, while spanning many number of process, which will finally result in stack buffer overflow situation, with this information malicious program tries to modify the OS behavior and launch malicious act, which changing the instruction pointer to move to malicious program execution.

E. *Excessive foreground time for non interacting and administrator app.*

There are malware which tries to take control of the device, by keeping them in foreground, without user's interaction, and hence not allowing other genuine program to run in foreground, which needs user's intervention. If any program tries to take control more than 30 seconds with admin privileges (Configured in MADAM), it will be kept in suspicious app list.

F. *Unauthorized installation of new apps.*

These malware tries to install other app either another kind of malware or advertisement app, without user's notice and authorization, and generally known as Installers.

G. *Unsolicited kernel level activity of background app.*

There are few malwares like Botnet, Spyware and some generic Trojans, which tries to generates an open, write or sendmsg system call, in background state, hence considered a misbehavior and detected as suspicious app.

II. REVIEW OF LITERATURE

Identifying malicious applications directly on the smart phone using static analysis, gathering as many features of an application as possible at run-time.[1]

Android Security Framework (ASF), a generic, extensible security framework for Android that enables the development and integration of a wide spec- trum of security models in form of code-based security modules.[2]

AppGuard, a powerful and flexible security system that overcomes these deficiencies. It enforces user defined security policies on untrusted Android applications without requiring any changes to a smart phones firmware, root access, or the
like.[3]

A policy-based framework for enforcing software isolation of applications and data on the Android platform. In MOSES, it is possible to define distinct security profiles within a single smart-phone.[4]

Alterdroid is a dynamic analysis tool that compares the behavioral differences between an original app and numerous automatically generated versions of it containing carefully injected modifications to detect Malware.[5]

An approach built to automatically perform out-of-the-box dynamic behavioral analysis of An- droid malware. This paper presents a unified analysis to characterize low-level OS-specific and high-level Android-specific behaviors.[6]

Proposed MADAM, a multilevel host-based malware detector for humanoid devices. By analyzing and correlating many options at four completely different levels.[7]

Mitigating Android malware installation through providing robust and lightweight classifiers. A thorough analysis to extract relevant features to malware behavior captured at API level,and evaluated different classifiers using the generated feature set.[8]

Studies done indicates that current Android permission warnings do not help most users make correct security decisions and present

recommendations for improving user attention and comprehension, as well as identify open challenges.[9]

There are experiments done to see the effectiveness of Malware detector and found that the best case detects 796202 , and still needs better develop next-generation anti-mobilemalware solutions.[10]

Comparison of MADAM Results with Existing Frameworks

System	Dynamic/Static	Rooting	Detection Rate	Overhead	Attack
MADAM	Both	Yes	96.9%	1.4%	Several Classes of Attacks
TaintDroid	Dynamic	Custom-ROM	N.A.	14%	Privacy Leak
Patronus	Dynamic	Yes	87%	7.1%	SMS - Spyware
DroidAnalyzer	Static	Offline	N.A.	N.A.	Rootkits
DroidSIFT	Static	Offline (Server)	93%	N.A.	General
AlterDroid	Static	Offline	97%	N.A.	Obfuscated Malware
ASF	Dynamic	Custom ROM	N.A.	2-3 %	Access Control

Fig. 1. Comparing with existing systems[11]

III. SYSTEM ARCHITECTURE / SYSTEM OVERVIEW

Below is an architecture diagram depicting the components of MADAM framework, which are used to detect and prevent app misbehavior's.

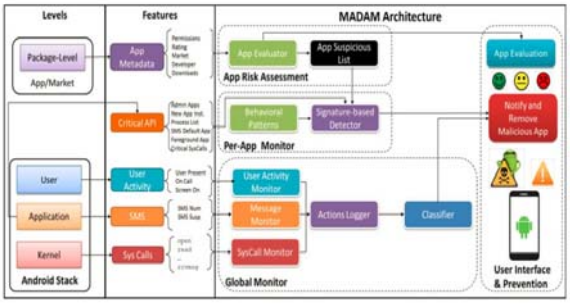

Fig. 2. MADAM Architecture[11]

To derive the features at the four system levels, and to detect and prevent a misbehavior, MADAM can be logically decomposed into following main architectural blocks, which are.

1) App Risk Assessment
2) Global Monitor.
3) Per-App Monitor.
4) User Interface and Prevention

Below are the few important modules used in the Live Update System.

A. App Risk Assessment

When a new app is installed on the device (deploy-time), the App Evaluator component intercepts and hijacks the installation event. This component analyzes the metadata of the new app to assess its risk, by retrieving features from the app package, related to critical operations, and from the market, related to app reputation. In detail, these features are:

1) The permissions declared in the manifest
2) market of provenance
3) The total number of downloads, developer reputation and 4) the user rating.

B. Global Monitor

Global Monitor : The Global Monitor is at the core of the MADAM framework, since it is responsible of collecting the run-time device behaviors and classifying them as genuine or malicious. Below are the important features-

1) Behavior is represented through a vector of features.
2) The features are extracted from different kinds of dynamic events like -User Activity, Critical API (in particular, SMS, i.e., text messages) and System Call (Sys Calls).
3) The Actions Logger is the component that records all these features into a vector, which is then fed to the Classifier.
4) Actions Logger is trained to recognize genuine behaviors versus malicious behavioral patterns.

C. Per-App Monitor

The Per-App monitor is based on a set of known malicious behavioral patterns which considers the Suspicious App List created by the App Risk Assessment module, the alerts raised by the Classifier and a set of features at application-level not considered by the Classifier.

D. User Interface and Prevention

User Interface and Prevention: Prevention module that acts as a security enforcement mechanism by blocking the detected misbehavior's like -

1) SMS being sent without the user authorization.
2) The User Interface (UI) module handles the process for removing the responsible app.
3) The UI conveys to the user all the events which require an active interaction, such as for removing malicious apps
4) Provide choices to user to select which behaviors should be blocked or allowed.

5) UI is exploited by the App Evaluator to communicate to the user the risk score of a new app at deploy-time. In this case, the user can then decide whether to continue the installation (or not) of the app

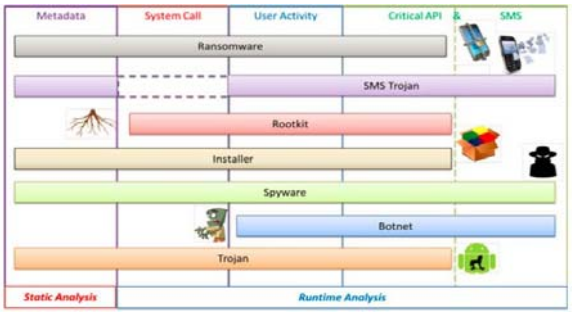

Fig. 3. Relevant features for the detection of the seven malware behavioral classes.[11]

IV. SYSTEM ANALYSIS

System analysis of MADAM covers the various aspects of MADAM framework like Algorithm, mathematical modelling, overhead of running MADAM in device, usage of battery and performance related aspects of MADAM. It is also important to analyze the accuracy of the detection and keep it to as low as possible to keep good usage confidence of the detector.

A. Algorithm

1) Launch the App Evaluator in background waiting for new apps to be installed.
2) Populate the App Suspicious List 3) Launch Global Monitor in background, to retrieve the 14 features and classify the app behaviors.
4) Per-App Monitor is launched to monitor kernel and API features to detect and stop known behavioral patterns,using Signature-based Detector.
5) The User Interface Prevention module kills the app deemed as responsible, and proposes the user to remove it.
6) Misbehaving app and the class of malware are communicated to the user, who takes the final decision on the app removal.

B. Mathematical Model

The first eleven features concern the system calls related to file modification and inter-component communication (i.e., open, ioctl, brk, read, write, exit, close, sendto, sendmsg, recvfrom, and recvmsg).Set of system call can be represented as -

Set of System Call = $[f_1.....,f_{11}]$ *each* f_j *is the number of occurrences of system call* f_j The KNN classifier exploits this geometric representation to classify behaviors closer to-

1- Genuine ones as belonging to class ω_0
2- Behaviors closer to the malicious ones as belonging to class ω_1

A K-NN (K-Nearest Neighbor) classifier, which is a similarity-based classifier, i.e., it classifies two similar elements as belonging to the same class. The similarity measure used by the K-NN classifier is the euclidean distance measured in the features space, i.e., two elements are considered similar if geometrically close in the features space .

This is computed as:

$$\text{Similarity}(x,y) = -\sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

where x_i *and* y_i *are the features of the vectors x and y.*

C. False Positives

Considering the reliability of the malware detector, it is important to keep False Positive rate as low as possible. Basically False Positive is the case when app is genuine , where as the detector is classifying it as Malcious. Although this is not a dangerous situation, but it will create the impact on the usability and reliability of the detector, and unnecessarily bother with incorrect information. Hence it is important to ensure that False Positive should not decrease the reliability of the Malware Detector. Here is the analysis for False Positive for MADAM.

Test	FPs	FPR	FPs/day
Light	3	$1 \cdot 10^{-5}$	0.5
Medium	8	$2.8 \cdot 10^{-5}$	1.1
Heavy	75	$2.6 \cdot 10^{-4}$	10.7

Fig. 4. False Alarms Experimental Results[11]

D. Performance Overhead and Energy Consumption

Performance of the MADAM framework is measured when the MADAM is running on the device versus when the MADAM is not running on the device. It is found that overhead of 1.4 percent when the MADAM is running, which is very much inline with other anti malware tools.

Battery depletion is monitored over two period in 24 hours, and found that it is only 4 percent.

V. ADVANTAGES/DISADVANTAGES

A. Advantages

1) MADAM uses behavior based approach and based on the classifiers which is based on 2800 real life Malware.

2) MADAM app can be run in the Learning mode, where it keeps updating it classifiers without UI interaction and alerts.

3) False positive rate is low , and prevent up to 96 percent Malware of real life.

4) Multilevel approach makes it possible to dynamically detect most of current Android malware, right on the device.

B. Disadvantages

1) MADAM also consumes the resources like processor time and memory for preventing and detecting the Malware, hence will reduce the throughput of the system globally.

2) As MADAM is based on the run time behavior of an app, so if any app is not exhibiting any behavior, it can hide itself for long time being dormant.

3) As MADAM is continuously observing the overall run time system, it consumes battery power, although its quite less as 4 percent.

VI. RESULTS

To ensure the correctness and consistency for malware detection, tests are conducted on testbsed of 2800 malware samples, which are tested on MADAM as well as VirusTotal.VirusTotal [12] is a web-service which performs the malware static analysis for malware files and URLs. When we submit a file to VirusTotal, then it submits the file to very well known anti-virus software (around 50 to 60) , and returns the results confirming if it is a clean file or it is a malware.

Malware	Class	Year	Samples	MADAM	VirusTotal
Lemon	Botnet	2012	6	0	6
MmarketPay	Botnet	2012	1	0	1
Basebridge	Installer	2011	330	330	330
CrWind	Installer	2015	1	1	1
FakeFlash	Installer	2012	3	3	3
Gamex	Installer	2012	7	7	7
Gapev	Installer	2012	7	7	7
Gappusin	Installer	2012	58	58	*0*
Ansca	Trojan	2011	1	0	1
Antares	Trojan	2012	2	0	2
Faketimer	Trojan	2012	16	16	16
Pujacks	Trojan	2013	1	0	0
Moghava	Trojan	2012	3	3	3
Copycat	Ransomware	2014	10	10	10
FoCober	Ransomware	2015	13	13	13
Koler.C	Ransowmare	2014	7	7	7
AsRoot	Rootkit	2011	8	8	8
Coogos	Rootkit	2014	8	8	8
Droidrooter	Rootkit	2011	3	3	3
DroidCoupon	Rootkit	2011	1	1	1
DroidKungFu	Rootkit	2011	402	402	402
ExploitLinuxLootor	Rootkit	2012	70	70	70
ExploitRageCage	Rootkit	2012	1	*1*	*0*
Oldboot.b	Rootkit	2012	1	1	1
Placms	Rootkit	2012	12	12	12
RATC	Rootkit	2012	1	*1*	*0*
Spyhasb	Rootkit	2012	13	13	13
Stiniter	Rootkit	2012	1	1	1
YZHC	Rootkit	2011	22	22	22
Adsms	SMS Trojan	2011	3	3	3
BaseBridge	SMS Trojan	2011	122	122	122
BgServ	SMS Trojan	2011	9	*9*	*0*
BeanBot	SMS Trojan	2011	8	8	8
DogoWars	SMS Trojan	2011	7	7	7
Dialer	SMS Trojan	2012	1	1	1
FakeInstaller	SMS Trojan	2013	925	925	925
FakeLogo	SMS Trojan	2012	20	20	20
FakeMart	SMS Trojan	2013	1	1	1
FakeNotify.B	SMS Trojan	2013	1	1	1
FakePlayer	SMS Trojan	2010	17	17	17
FakeRegSMS.B	SMS Trojan	2013	41	41	37
Foncy	SMS Trojan	2012	2	2	2
GGTrack	SMS Trojan	2013	5	5	5
GoldenEagle	SMS Trojan	2011	1	1	1
Hispo	SMS Trojan	2014	3	3	3
Jifake	SMS Trojan	2012	29	29	29
Jamshider	SMS Trojan	2012	2	2	2
Mania	SMS Trojan	2012	6	6	6
Opfake	SMS Trojan	2012	14	14	14
Poder	SMS Trojan	2015	*1*	*1*	0
Qiscom	SMS Trojan	2012	1	1	1
Raden	SMS Trojan	2012	10	10	10
Saiva	SMS Trojan	2014	2	2	2
Samsapo	SMS Trojan	2013	1	1	1
Scavir	SMS Trojan	2012	1	1	1
Seaweth	SMS Trojan	2012	6	6	6
Selfmite.B	SMS Trojan	2013	1	1	1
SerBG	SMS Trojan	2013	14	14	14
SingaporeSMSWorm	SMS Trojan	2014	1	1	1
SmsSend	SMS Trojan	2013	1	1	1
Ssmsp	Sms Trojan	2014	1	1	1
Stealer	SMS Trojan	2014	14	14	14

Fig. 5. Detection Results and Comparisons (Part 1)[11]

VII. CONCLUSION

We can conclude MADAM, as a Multi-level host-based malware detector for Android devices. From results we can see, MADAM is able to detect misbehavior's from malware behavioral classes that consider 125 existing malware families, which encompass most of the known malware. MADAM is the first system which aims at detecting and stopping at run-time any kind of malware, without focusing on a specific security threat, using a behavior-based and multi-level approach. Great detection and prevention capabilities at the cost of 1.4 percent performance overhead and a 4 percent battery depletion.

Malware	Class	Year	Samples	MADAM	Virus Total
ACNetDoor	Spyware	2015	1	0	1
Aks	Spyware	2012	5	5	5
Arspam	Spyware	2011	1	1	1
Booster	Spyware	2014	1	1	1
Cellshark	Spyware	2011	1	0	1
Dougalek	Spyware	2012	9	0	9
DynSrc	Spyware	2013	1	*1*	*0*
Fidall	Spyware	2012	1	0	1
Flexispy	Spyware	2011	2	2	2
GamblerSms	Spyware	2011	1	0	1
Gone60	Spyware	2011	9	9	9
GPSSMSSpy	Spyware	2011	6	0	6
Kidlogger	Spyware	2011	6	6	6
Kmin	Spyware	2011	52	52	52
Kiser	Spyware	2013	9	9	0
Maistealer	Spyware	2012	1	1	1
MobileSpy	Spyware	2012	14	0	14
Mobinauten	Spyware	2011	8	8	8
Mtracker	Spyware	2014	1	*1*	*0*
Netisend	Spyware	2011	1	0	1
NickySpy	Spyware	2011	2	0	2
Plankton	Spyware	2011	11	0	11
SndApps	Spyware	2011	10	10	10
Spitmo	Spyware	2011	11	11	11
Tapsnake	Spyware	2011	2	0	2
SMS Replicator	Spyware	2013	4	4	4
Sheridroid	Spyware	2012	2	0	2
SmForw	Spyware	2011	2	2	2
SmsSpy	Spyware	2013	1	1	1
SmsZombie	Spyware	2012	10	0	10
Spybubble	Spyware	2011	3	0	3
Spy.Imlog	Spyware	2012	1	1	1
Spyoo	Spyware	2012	3	3	3
Tesbo	Spyware	2012	1	0	1
Trackplus	Spyware	2014	6	0	6
Typstu	Spyware	2011	14	14	14
Vdloader	Spyware	2011	16	16	16
Walkiwat	Spyware	2011	1	1	1
Ycchar	Spyware	2012	2	0	2
Ksapp	Spyware + Installer	2012	6	6	6
DroidDream	Spyware + Rootkit	2011	16	16	16
Gmuse	Spyware + Rootkit	2014	3	3	3
zHash	Spyware + Rootkit	2011	11	11	11
Dabom	SMS Trojan + Installer	2014	2	2	2
Updtkiller	SMS Trojan + Installer	2012	1	1	1
Bosm.C	SMS Trojan + Installer	2015	1	1	1
Boxer	SMS Trojan + Installer	2011	27	27	27
Cawitt	SMS Trojan + Spyware	2012	1	1	1
Cosha	SMS Trojan + Spyware	2012	10	10	10
Fjcon	SMS Trojan + Spyware	2012	4	4	4
MobileTx	SMS Trojan + Spyware	2012	*69*	69	*69*
Nandrobox	SMS Trojan + Spyware	2012	13	13	13
Geinimi	SMS Trojan + Botnet	2011	*69*	69	*69*
Total	-	-	**2,784**	**2,700**	**2,709**
Accuracy	-	-	-	**96.9%**	**97.3%**

Fig. 6. Detection Results and Comparisons (Part 2)[11]

TrojanSMS.BoxerAQ	SMS Trojan	2012	1	1	1
TrojanSMS.Denofow	SMS Trojan	2011	5	*5*	*0*
TrojanSMS.Hippo	SMS Trojan	2012	13	13	3
TrojanSMS.Stealer	SMS Trojan	2014	1	1	1
Updtbot	SMS Trojan	2012	1	1	1
Vidro	SMS Trojan	2014	5	5	5
XXShenqi	SMS Trojan	2014	1	1	1
Zsone	SMS Trojan	2011	12	12	12

Fig. 7. Detection Results and Comparisons (Part 3)[11]

REFERENCES

[1] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, Drebin: Eective and explainable detection of android malware in your pocket, in Proc. Netw. Distrib. Syst. Security Symp., 2014, pp. 115.

[2] M. Backes, S. Bugiel, S. Gerling, and P. von Styp-Rekowsky, Android security framework: Extensible multi-layered access control on android, in Proc. 30th Annu. Comput. Security Appl. Conf., 2014, pp. 4655. [Online]. Available: http://doi.acm.org/10.1145/2664243.2664265.

[3] M. Backes, S. Gerling, C. Hammer, M. Maei, and P. von Styp-Rekowsky, Appguard ne-grained policy enforcement for untrusted android applications, in Proc. Data Privacy Manag. Auton. Spontaneous Security, 2014, pp. 213231.

[4] Y. Zhauniarovich, G. Russello, M. Conti, B. Crispo, and E. Fernandes, Moses: Supporting and enforcing security pro les on smartphones, IEEE Trans. Dependable Secure Comput., vol. 11 , no. 3, pp. 211223, May 2014.

[5] G. Suarez-Tangil, J. Tapiador, F. Lombardi, and R. Di Pietro, Thwarting obfuscated malware via dierential fault analysis, Computer, vol. 47, no. 6, pp. 2431, Jun. 2014.

[6] A. Reina, A. Fattori, and L. Cavallaro, A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors, in Proc. ACM Eur. Workshop Syst Security, Apr. 2013, pp. 115.

[7] V. Misra. (2013). What are the exact mechanisms/ aws exploited by the rage against the cage and z4root android exploits?. [Online]. Available: http://www.quora.com/What-are-the-exactmechanismsaws-exploited-by-the-rage-against-the-cage-andz4root- Android-exploits.

[8] Y. Aafer, W. Du, and H. Yin, Droidapiminer: Mining api-level features for robust malware detection in android, in Proc. Security Privacy Commun. Netw., 2013, vol. 127, pp. 86103.

[9] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, Android permissions: User attention, comprehension, and behavior, in Proc. Symp. Usable Privacy Security, 2012, p. 3.

[10] Y. Zhou and X. Jiang, Dissecting android malware: Characterization and evolution, in Proc. IEEE Symp. Security Privacy, 2012, pp.

95109. [Online]. Available: http://dx.doi.org/10.1109/ SP.2012.16.

[11] Andrea Saracino, Daniele Sgandurra, Gianluca Dini, and Fabio Martinelli, "MADAM: Eective and Ecient Behavior-based Android Malware Detection and Prevention," in Proc. IEEE Symp. Security Privacy, 2018 , IEEE Transactions on dependable and secure computing, VOL. 15, NO. 1, January/February 2018.

[12] Google Groups. (2015). Virustotal. [Online]. Available: https:// www.virustotal.com/.