DOI: 10.1109/ICCONS.2017.8250682
https://ieeexplore.ieee.org/document/8250682

Study of an effective way of Detecting Unexpected Permission Authorization to Mobile Apps

Manisha Patil*
Research Scholar - PhD (Computer Studies)
Symbiosis Centre for Research and Innovation, Symbiosis International University. Pune. 412 115, Maharashtra, India.
E-mail: manishashivani@gmail.com

Prof. Dr.Dhanya Pramod**
Director/Professor
Symbiosis Centre for Information Technology (SCIT)
Hinjawadi, Pune – 411057, Maharashtra, India.
E-mail: dhanya@scit.edu

***Abstract*: The recent boom in Android mobile device usage has caused a shift in the information technology and has affected the way how information and data are stored, shared among the mobile users. The advent of social networking applications also demands the availability of resources that can be shared among the authentic users. This paper reviews and compares the available techniques and solutions for detecting Unexpected Permission Authorization to Mobile Apps. It is observed that malware for the android system is also growing significantly, current solutions for detecting malware on smartphones are still ineffective.**

Keywords— Android Mobile App Security, User Privacy, Permission check, de-compilation, Static analysis .

I INTRODUCTION

Android Smartphone OS has captured more than 80 % of the total market share, leaving its competitors iOS, Windows mobile OS and Blackberry far behind [1]. Gartner Smartphone sale report 2014 reports 53.2% in Android devices compared to the previous year [1]. The overall market share increased to 80% from 66% compared to the past two years, a substantial rise of 14% among the users. Ubiquitous Internet connectivity and availability of personal information such as contacts, messages, social network access, browsing history , online shopping and banking credentials have attracted the attention of malware developers towards the mobile device in general and Android in particular.

Android malware such as premium rate SMS Trojans, spyware, botnets, aggressive adware and privilege escalation attack exploits reported exponential rise apart from being distributed from the secure Google Play store and well-known third-party marketplaces [2, 10, and 32].

Unlike the Apple app store, Google play does not verify the uploaded apps manually. Instead, official market depends on Bouncer [4] [19], a dynamic emulated environment to control and protect the marketplace from the malicious app threats.

Mobile Malware is any kind of, intrusive or annoying software or program code designed to use a device without owner's consent. Malware is often distributed as a spam within a malicious attachment or a link in an infected websites. Malware – virus, worm, Trojan, Rootkits, botnet.

Operating System	2013	2014	2015
Android	879,821	1,170,952	1,358,265
Windows	325,127	339,068	379,299
iOS/Mac OS	241,416	286,436	324,470
Others	873,194	683,519	565,186
Total	**2,319,559**	**2,479,976**	**2,627,221**

Table – I: Worldwide Device Shipment by Operating System (Thousands of Units). Gartner (March 2014). [1]

Mobile security stack shows four different layers where vulnerability issues can be studied such as application, operating system, hardware and infrastructure Layer.

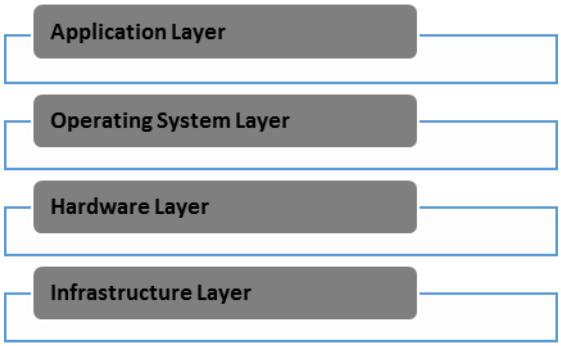

Fig 2: Mobile Security Stack

Application Layer security is considered for the study. It is important to underline that evolution of malware is a continuous race between attackers and defenders both use the same programming methods, tools and resources either to create a malware or to develop an intelligent malware detection mechanism.

Mobile environment threats may affect different assets like 1) Personal data 2) Corporate Intellectual property 3) Classified information 4) Financial assets 5) Device and service availability and functionality 6) Personal and political reputation.

Threats in mobile environment are reported such as Data leakage resulting from device loss or theft, an Unintentional disclosure of data, -attacks on decommissioned devices: phishing, spyware, network spoofing, surveillance, dialer ware, financial malware, network congestion

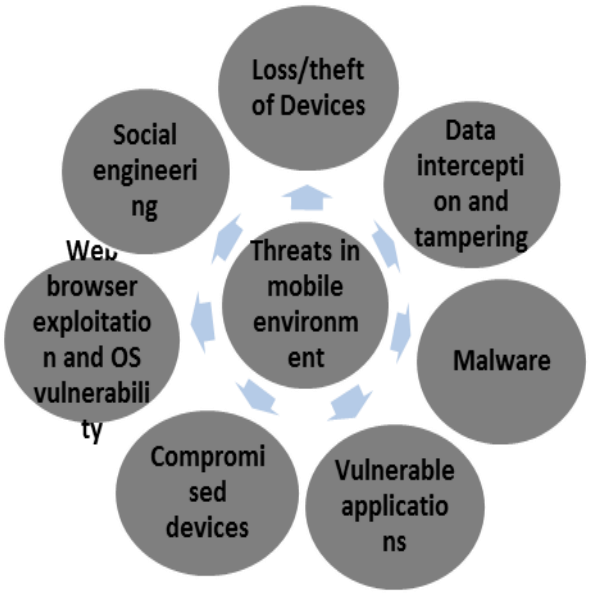

Fig 3: Threats in a mobile environment

.

Methodologies to perform attacks against smartphones are categorized using classes - Wireless, break- in, infrastructure based, worm based, bot-net, user based. Attackers use possible methodologies to perform an attack in a mobile environment to reach different goals like privacy, sniffing, denial of service, overbilling.

II LITERATURE REVIEW

Researchers have made some work on malware detection on the Smartphone. It laid the valid foundation for further research. Performing application analysis is an effective way of detecting malicious applications on smartphones. Application analysis can be performed statically before an application is running. Markus Miettinen [32] et al. analyzed how the malware performed malicious activities on Smartphone and pointed out that it is an effective protection by seeing the Smartphone system itself as a test object. They suggested that some information should be monitored for anomaly detection, such as operating system events, resource usage, application-level events, and so on. Finally, they came up with a unified intrusion detection model. But they did not propose a specific and feasible anomaly detection scheme.

Schmidt [8] et al. proposed a solution based on monitoring events occurring on Linux-kernel level. They use kernel system calls, network activity events and file system logs to detect anomalies in the system. At that time, there were no real Android devices available, so they failed to test their system properly.

Shabtai [7] et al. proposed Adnromaly — a framework for anomaly detection on Android smartphones. The framework continuously monitored the information obtained from the Smartphone. Then, it applied machine learning to classify the collected data as benign or malicious. Yet they could not find real malware to test their proposal. Enck et al [40] used de-compilation and static analysis techniques to study 1100 free applications from the official Android Market to understand a broad range of security-related metrics associated with these applications. They discovered that sensitive information is widely leaked in applications. For instance, more than half of the applications include at least one advertisement libraries that collect and send private information, e.g. the location of the phone.

Pridgen & Wallach [34] examined a sample of 114,000 apps and found that the number of permissions required by apps is increasing, and consequently, posing a privacy risk to Android users.

Felt et al. [8] and Kelley et al. [18] suggested that many users have a low comprehension of the Android permissions system – that is the permissions system may be insufficient in providing adequate user privacy in the hands of a novice user.

Kern & Sametinger [28] took a different approach and recommended the use of fine-grained individual permissions control on a per app basis. This means that each Android app would have each of their permissions explicitly listed and the user would either deny or allow the permission request.

Zhou et al. [43] designed a system that could control an app's access to sensitive permissions Berthome et al. [10] proposed a set of two apps, comprising (1) the Security Monitor, a third party app installed onto the device, and (2) the Security Reporter, which would be injected into a decompiled target app. The injected app is able to monitor the targeted app and can then report to the Security Monitor with details such as resource requests.

Juanru, Dawu & Yuhao [27] used a similar technique of decompiling Android apps to aid with their Android malware research.

Xu, Saïdi & Anderson [42] developed a solution called Aurasium that automatically repackages Android apps to have sandboxing and policy enforcement abilities in order to enhance user privacy.

Application analysis can also be performed dynamically by monitoring a running application. A representative example is TaintDroid [39]. It applies dynamic taint tracking and analysis on the usage of sensitive data on Android. Any information which comes from a trusted application is considered to be tainted. TaintDroid marks data coming from the "taint sources" and tracks the taint flow. If the data, in the end are used by an untrusted application, TaintDroid reports it as a sensitive data leak. However, TaintDroid cannot print alert messages for many of the malware samples that we have evaluated. Building on top of TaintDroid,

Kirin [41], an application certification for Android. Kirin performs a permission check on the application during installation. When a user 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 delete it or alert the user.

Crowdroid [25] is a lightweight client application that monitors system calls invoked by the target mobile application, preprocesses the calls, and sends them to the cloud, where a clustering technique helps determine whether the application is benign or malicious. Increased use of Crowdroid will result in improved malware detection, but using the approach initially might cause false positives as the sample size is still very small. Moreover, it isn't clear how users will react when they're asked to send application behavior to a third party, and total dependence on user behavior might not produce accurate results.

CloudAV [26] is a cloud-based antivirus file scanning mechanism, but it lacks the features required to detect zero-days attacks, remote exploits, and memory-resident attacks.

Paranoid Android (PA) [21], a cloud-based malware protection 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 Smartphone, records all the necessary information required to replay the mobile application's

execution. The tracer transmits the recorded information to the cloud-based replayer, which replays the execution in the emulator. The replayer can deploy several security checks, such as dynamic malware analysis, memory scanners, system call anomaly detection, and commercial antivirus scanning, from the cloud's ample resources. PA uses a proxy to temporarily store inbound network traffic information so that the phone can save energy by not sending this data back to the server. Instead, the server can directly contact the proxy to get the network traffic information needed to successfully replay the execution. However, PA incurs some significant overhead, increases the CPU load by 15 percent, and consumes 30 percent more energy during heavyweight tasks. Furthermore, because tracing systems implement the tracer module in the user

AppFence [30] implements two simple runtime mechanisms to protect users' privacy. The first one is data shadowing, a mechanism that returns fake or blank data when an untrusted application requests private data such as phone IDs and location information. The second idea is to block an application's communication from sending out private information at runtime. It prevents the exhilaration of sensitive data by intercepting calls to the network stacks to detect when such data is written to a socket. Such offending messages are dropped. These two approaches are phone-based solutions and can potentially add much CPU overhead

Bugiel et al. [37] Present a security framework named XManDroid which monitors the real-time communication between applications and verifies the inter-process communications against a set of pre-defined security policies. The aim is to prevent malicious applications from exploiting transitive permission properties to enable privilege escalation. In order to test their methodology, they included 7 types of attacks in their dataset which cover several possible scenarios whereby rogue applications request transitive permissions. For future work, the authors plan to integrate the methodology into the existing permission framework currently in use by Android.

In the work carried out by Portokalidis et al. [22], Paranoid Android is a security model implemented on remote servers where identical copies of smartphones are running in a virtual environment. A program, which resides on the device, collects all the necessary information needed to replay the execution and transmits it to the remote server. The information is re-executed on the virtual smartphones. The aim is to run constant security checks on applications while maintaining minimal computational and battery overhead.

III CONCLUSION

The study shows, the approaches proved valuable in protecting smart phones but they have restrictions. In particular, the Android system has been in a dominant position in the market of Smartphone operating system. Malware for the Android system is also growing significantly. Therefore, it is necessary to develop a security suite for the Android phones, such as signature-based anti-virus technology, Smartphone firewall, access control mechanisms and lightweight Intrusion Detection Technology.

Malware for smartphones shows the traditional features that the malware for personal computers has, adding new threats to privacy and security, as they leverage the peculiar characteristics of smartphones (GPS, sensitive data like contacts book, agenda ,SMS, microphone and camera) and limitations (small screen, reduced resources, power supply). Current solutions for detecting malware on smartphones are still ineffective: it urges to provide new and successful methods and tools for contrasting the rapid spreading of malware.

REFERENCES

[1] G. Inc., Android Smartphone Sales Report, 2014, http://www.gartner.com/newsroom/id/2996817 (Online; Last Accessed March 17 2015).

[2] Android Malware Genome Project, http://www.malgenomeproject.org/(Online; Last Accessed 11th February 2015).

[3] AppBrain, Number of applications available on Google Play, http://www.appbrain.com/stats /number-of-android-apps

[4] Android and security: Official mobile Google blog, http://googlemobile.blogspot.in/2012/02/android-and-security.html (Online; Last Accessed 15th October 2014).

[5] A.-D. Schmidt and S. Albayrak, "Malicious Software for Smartphones," Technische Universit¨at Berlin - DAI-Labor, Tech. Rep. TUBDAI 02/08-01, February 2008, http://www.dai-labor.de.

[6] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "Survey of Mobile Malware in the Wild," 2011. [Online]. Available:http://www.eecs.berkeley.edu/∼afelt/malware.html

[7] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "Andromaly: a behavioral malware detection framework for android devices". Journal of Intelligent Information Systems, pages 1–30, 2011. 10.1007/s10844-010-0148-x

[8] Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Jan Clausen, Kamer Ali Y¨uksel, Osman Kiraz, Ahmet Camtepe, and Sahin Albayrak. "Enhancing security of linux-based android devices". In in Proceedings of 15th International Linux Kongress. Lehmann, October.2008.

[9] Backdoor.AndroidOS.Obad.a, http://contagiominidump.blogspot.in/2013/06/ backdoorandroidosobada .html (Online; Last Accessed December 252014).

[10] P. Berthome, T. Fecherolle, N. Guilloteau & JF. Lalande, "Repackaging Android Applications for Auditing Access to Private Data", ARES 2012, pp. 388-396.

[11] C. A. Castillo, Android Malware Past, Present, and Future, Tech. rep., Mobile Working Security Group McAfee (2012).

[12] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, W. Lee, The Core of the Matter: Analyzing Malicious Traffic in Cellular Carriers, in: Proc. NDSS, Vol. 13, pp. 1–16.

[13] Carat: Collaborative Energy Diagnosis, http://carat.cs.berkeley.edu/ (Online; Last Accessed December 25 2014).

[14] E. Chin, A. P. Felt, K. Greenwood and D. Wagner, "Analyzing inter-application communication in Android," Analyzing inter-application communication in android., pp. 239-252, 2011.

[15] Fakedefender.B - Android Fake Antivirus, http://contagiominidump.blogspot.in/2013/11 /fakedefenderb-androidfake-antivirus.html (Online; Last Accessed December 25 2013).

[16] Fake Netxflix - Android trojan info stealer, http://contagiominidump.blogspot.in/2011/10/fake-netxflix-adtroidtrojan- info.html (Online; Last Accessed 11th February).

[17] F. Shahzad, M. A. Akbar, M. Farooq, A Survey on recent advances in malicious applications Analysis and Detection techniques for Smartphones.

[18] A.P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin & D. Wagner, "Android permissions: User attention, comprehension, and behavior", SOUPS 2012, p. 3.

[19] Google bouncer : Protecting the Google play market,http://blog.trendmicro.com/trendlabs-security-intelligence/a-lookat-google-bouncer/ (Online; Last Accessed 15th October 2014). 13

[20] G. Andre, P. Ramos, BOXER SMS Trojan, Tech. rep., ESET Latin American Lab (2013).

[21] G. Portokalidis et al., "Paranoid Android: Versatile Protection for Smartphones," Proc. Ann. Computer Security Applications Conf. (ACSAC 10) ACM, 2010, pp. 347-356.

[22] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: versatile protection for smartphones," in 'Proceedings of the

26th Annual Computer Security Applications Conference', Austin, Texas, 2010, pp. 347-356.

[23]H. Dediu, D. Schmidt, and R. Salle. (Visited March 2015) Asymco.[Online]. Available: http://www.asymco. Com/

[24] H. T. T. Truong, E. Lagerspetz, P. Nurmi, A. J. Oliner, S. Tarkoma, N. Asokan, S. Bhattacharya, The Company You Keep: Mobile Malware Infection Rates and Inexpensive Risk Indicators, arXiv preprintarXiv:1312.3245.

[25] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-Based Malware Detection System for Android," Proc. ACM Workshop Security and Privacy in Mobile Devices (SPMD 11), ACM, 2011, pp. 15-26.

[26] J. Oberheide, E. Cooke, and F. Jahanian, "CloudAV: N-Version Antivirus in the Network Cloud," Proc. 17th Conf. Security Symp. Usenix, 2008, pp. 91-106

[27] L. Juanru, G. Dawu & L. Yuhao, "Android Malware Forensics: Reconstruction of Malicious Events", ICDCSW 2012, pp. 552-558.

[28] M. Kern, & J. Sametinger, "Permission Tracking in Android", UBICOMM 2012, pp. 148-155.

[29] L. Inc., State of Mobile Security 2012, Tech. rep., Lookout Mobile Security (2012).

[30] L. Inc., Current World of Mobile Threats, Tech. rep., Lookout Mobile Security (2013).

[31] M. Hypponen, "Mobile Security Review September 2010," F-Secure Labs, HelsinkiFinland, Tech. Rep., September 2010.

[32]M. Miettinen and P. Halonen. "Host-based intrusion detection for advanced mobile devices". 2006.

[33] P. Hornyack, S. Han, J. Jung, S. Schechter and D. Wetherall, "These Aren't the Droids You're Looking For: Retrofitting Android to Protect Data from," the 18th ACM conference on Computer and communications security (CCS), pp. 639-652, 2011

[34] T. Book, A. Pridgen & DS. Wallach, "Longitudinal Analysis of Android Ad Library Permissions", arXiv preprint arXiv: 1303.0857, 2013.

[35] S. Shekhar, M. Dietz & D.S. Wallach, "Adsplit: Separating smartphone advertising from application", CoRR, abs/1202.4030, 2013.

[36] Spitmo vs Zitmo: Banking Trojans Target Android, https://blogs.mcafee.com/mcafee-labs/spitmo-vs-zitmo-bankingtrojans-target-android (Online; Last Accessed 11th February).

[37] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, and Ahmad-Reza Sadeghi, "XManDroid: A New Android Evolution to Mitigate Privilege Escalation Attacks," Technical Report, Technische Universit Darmstadt2011.

[38] W. Zhou, Y. Zhou, X. Jiang, P. Ning, Detecting Repackaged Smartphone Applications in Third-party Android Marketplaces, in: Proceedings of the second ACM conference on Data and Application Security and Privacy, CODASPY '12, ACM, New York, NY, USA, 2012, pp.317–326. doi:10.1145/2133601.2133640. URL http://doi.acm.org/10.1145/2133601.2133640

[39] W. Enck, P. Gilbert, B.-G. Chun, L. Cox, J. Jung, P. McDaniel and A. Sheth, "Taintdroid: an information-flow tracking system for real-time privacy monitoring on smartphones," In 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 1-6, 2010.

[40] W. Enck, D. Octeau, P. McDaniel and S. Chaudhuri, "A study of Android application security," 20th Usenix Security Symposium, 2011.

[41]W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," Proc. 16th ACM Conf. Computer and Communications Security (CCS 09), ACM, 2009, pp. 235-245.

[42] Xu R., H. Saïdi & R. Anderso, 'Aurasium: Practical policy enforcement for android applications', 21st USENIX conference on Security symposium, 2012, pp. 27-27.

[43] Y. Zhou, X. Zhang, X. Jiang & V. Freeh, "Taming information-stealing smartphone applications (on Android)", TRUST 2011, pp. 93-107

Table II: Comparative study of existing techniques and solutions for Mobile Apps

Tool / Author	Purpose	Method	Remark
Markus Miettinen [29] et al.	some information monitored for anomaly detection, such as operating system events, resource usage, application-level events	a unified intrusion detection model	did not propose a specific and feasible anomaly detection scheme
Schmidt [8] et al.	use kernel system calls, network activity events and file system logs to detect anomalies in the system	solution based on monitoring events	no real Android devices available so failed to test system
Shabtai [7] et al Adnromaly	machine learning to classify data	a framework for anomaly detection	Could not find real malware to test proposal.
Enck et al [34]	to understand a broad range of security-related metrics	de-compilation and static analysis techniques	discovered that sensitive information is widely leaked
TaintDroid [33]	sensitive data on Android	dynamic taint tracking and analysis	cannot print alert messages for many of the malware samples
Kirin[35],	Permission check on the application during installation.	application certification	Delete app or alert the user
Crowdroid[23]	monitors system calls preprocess the calls and sends them to the cloud	clustering technique	dependence on user behavior
CloudAV[24]	file scanning	cloud-based antivirus	It lacks the features required to detect zero-days attacks, remote exploits, and memory-resident attacks.
Paranoid Android (PA)[19]	moves security analysis and computations to a remote server	cloud-based malware protection	increases the CPU load by 15 percent, and consumes 30 percent more energy during heavyweight tasks
AppFence [30]	data shadowing, block an application's communication from sending out private information at runtime	protect users' privacy	phone-based solutions and can potentially add much CPU overhead
Bugiel et al. [31] XManDroid	monitors the real-time communication between applications	security framework	Cover several possible scenarios against a set of pre-defined security policies.
Portokalidis et al. [20]	remote servers	Paranoid Android is a security model	Constant security checks on applications while maintaining minimal computational and battery overhead.