A New Mobile Malware Classification for Call Log Exploitation

Madihah Mohd Saudi, Nur Fadhilah Mohd and Nurlida Basir

***Abstract*—The increase of smartphone usage has led to the tremendous growth of the mobile application (apps) development. However, some of the developers overlook security aspects that may leave apps to be infected with mobile malware. An effective way to encounter mobile malware infection is important to avoid the victim of the infected smartphone to lose any confidential, contacts, messages, pictures and any other important information in their smartphone. Therefore, this paper presents a new mobile malware classification based on call log exploitation to encounter the mobile malware infection using the covering algorithm. The system call has been used as the main feature for the classification formation. The experiment was conducted on 5560 Drebin dataset as for training dataset and 500 mobile applications from Google Play Store as the testing dataset in a controlled lab environment. Open source tools and dynamic analysis were used in this experiment. 206 patterns have been produced based on the proposed mobile malware classification. Furthermore, we have evaluated our patterns with the mobile apps from the Google Play Store, where 4% from the mobile apps in Google Play Store matched with our mobile malware patterns. This new mobile malware classification is beneficial to distinguish between the benign and malicious mobile apps and can be used as guidance for other researchers with the same interest in mobile malware analysis field.**

***Index Terms*—mobile malware, call log exploitation, system call.**

I. Introduction

The high demand for mobile apps has led to the boom in the mobile apps development. Apps can be easily downloaded either from the Android, Apple, Google Play store or any third party stores and there is no guaranteed that the apps are virus free. Mobile malware is specifically built to attack apps in the smartphone systems. It relies on exploits of particular operating system and mobile phone software technology and majority of mobile malware targets the Android platform. Most mobile apps asked the user for permission to access information such as contacts, cameras, messages and location and user just accepts this term and condition and assumes the apps are secured and the data will not be misused. Surprisingly, an article written by Tara stated that Android app stores especially third-party ones are flooded with mobile malwares [1]. Furthermore, 10 million Android phones worldwide have been infected by auto-rooting apps especially by the HummingBad, which is known as one of the famous auto-rooting malware [2]. Nevertheless, several threats have been found where they can seriously affect Android applications. One of the threats is by repackaging the mobile malware with benign apps and republish into application store [5]. Stagefright is an example of this threat, where it embedded itself in audio file [3]. As for call exploitation among the mobile malware examples are DroidJack/SandoRAT and Android.Hehe [12,13].

One of the common ways to infect the apps is via system call. System call is used to call native functionalities of the kernel in the smartphone system [4]. System calls supply useful functions to application like network, file, or process related with operations. There are several works that have been carried out for mobile malware detection summarized in Table 1 [6-10]. Based on Table 1, apart from system call, API and permission are another 2 ways to exploit the smartphone system.

TABLE 1
SUMMARY OF RELATED WORKS.

Related Work	Key Feature For Analysis	Advantage	Challenges
Burguera *et al.* 2011) [6]	System Call	Detection of genuine app based on system call	False-positive more likely to occur if the app make use less system call
Ham & Lee (2014)[7]	System Call	Improved version of Crowdroid with better detection of malware	The existing weakness in Crowdroid is not addressed, no improvement on similar issue. Have a small misclassification error
Gonzalez et al. (2015) [8]	Meta-information features and n-grams	Can incrementally process apps without training period or predefined detection patterns	

Manuscript received July 18, 2016; revised August 9, 2016. This work was supported by Ministry of Higher Education (Malaysia), FRGS grant: [FRGS/1/2014/I CT04/USIM/02 /1].

Madihah Mohd Saudi is with the Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), as an associate professor and a research fellow in Islamic Science Institute, Universiti Sains Islam Malaysia (USIM), Bandar Baru Nilai, Nilai, 71800, Malaysia. (e-mail: madihah@usim.edu.my).

Nurlida Basir is a senior lecturer and with the Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), Bandar Baru Nilai, Nilai, 71800 (email: nurlida@usim.edu.my).

Nur Fadhilah Mohd is with the Information Security and Assurance (ISA) programme Faculty of Science & Technology (FST), Universiti Sains Islam Malaysia (USIM), 71800 Nilai, Negeri Sembilan, Malaysia.

ISBN: 978-988-14047-4-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Almin et al (2015) [9]	API calls	Have a system to help user identify benign and malign apps during installation	Limited to K-means and Naïve Bayesian
Sahs & Khan (2012) [10]	Permission	Have 2 groups of permissions to detect the malware	Not considering system call and API.

Therefore, this paper aims to investigate and evaluate how mobile malware exploits system call for call log. Based on the experiment conducted, a new mobile malware classification based on system call for call log exploitation for Android smartphone has been developed in this paper. 206 patterns have been developed based on the proposed mobile malware classification. The proposed classification is further evaluated with 500 mobile apps from Google Play Store and 4% of the apps matched with patterns proposed.

This paper is organized into four sections. Section 2 presents the methodology used in this research. Section 3 presents the results discussion. Section 4 concludes and summarizes the potential future work of this research paper.

II. METHODOLOGY

In order to investigate and evaluate how mobile malware exploit the system call and to develop a system call classification based on the surveillance of the call log on a smartphone; this research follows the framework as illustrated in Figure 1 for data collection and analysis processes. Two types of dataset used in this research are training dataset and testing dataset. The training dataset was downloaded from Drebin project where this dataset was collected from the Play Store, different alternative Chinese Markets, alternative Russian Markets and other Android websites, malware forums and security blogs during August 2010 to October 2012. Additionally, it includes all samples from the Android Malware Genome Project. After the adware samples have been removed, the final dataset contains 5,560 malware samples. Drebin dataset is one of the largest malware datasets that has been used to evaluate malware detection in Android [11]. Several top 24 families of malware that contains in Drebin dataset are FakeInstaller, DroidKungFu, Plankton, Opfake, GinMaster, BaseBridge and DroidDream. On the other hand, the testing dataset was downloaded from Google Apps Store. The testing dataset was used to test the accuracy of the result produced by this research. There were a total of 500 anonymous samples downloaded from the most popular applications used among the Android users.

A controlled laboratory environment is used as illustrated in Figure 2. All the tools and software used in this experiment are almost 80% open source application and available on a free basis. To prevent any leakage of the mobile malware to the public, outgoing connection is strictly prohibited. Any antivirus that installed in the personal computer also disabled to forbid the immediate elimination of the application. Then, the Drebin dataset that mentioned above were tested in this lab.

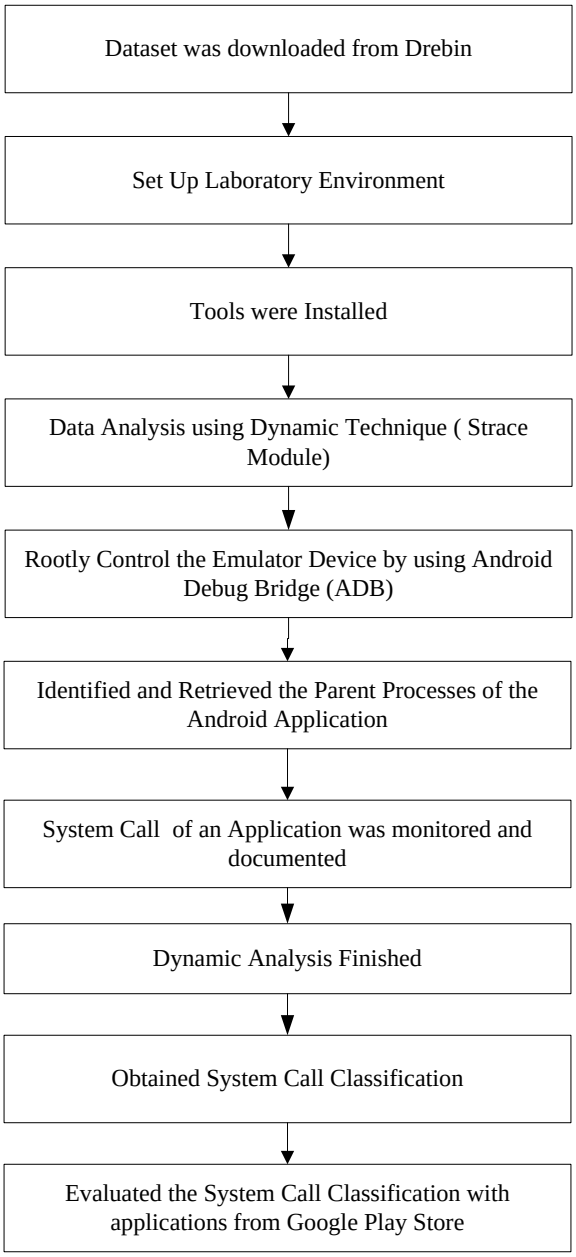

Fig. 1. Overall Research Processes

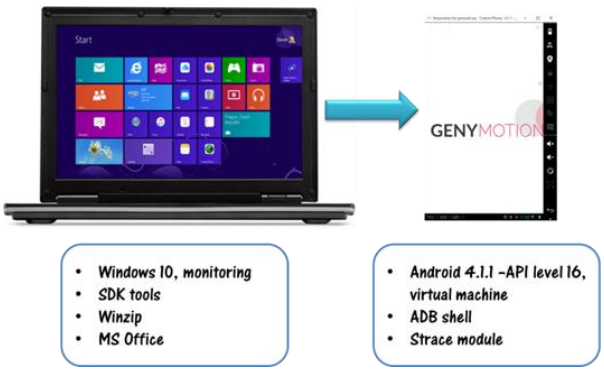

Fig. 2. Tools

ISBN: 978-988-14047-4-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Dynamic analysis approach was used in this research to accomplish the purpose of this research. This approach is where the mobile malware sample is executed within a controlled lab and its action and behavior is observed. The behaviors of the application were monitored through system calls. Figure 3 shows the example of system call that captured in a running application. The steps of analysis processes that are involved:

- Start Android Virtual Device from Genymotion
- Run Android Debug Bridge
- Install the application (adb install xxx.apk)
- Emulate the device (adb shell)
- Identify and retrieve the parent process of the Android application (ps)
- Entry the point to trace the running application's system call (strace –p id –c)

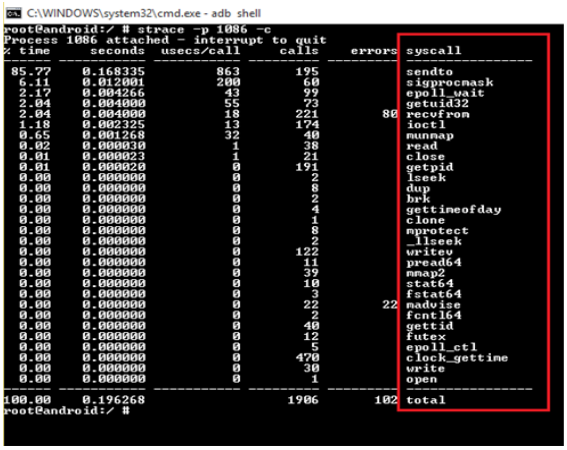

Fig. 3. Examples of System Calls Executed

Method covering algorithm was applied to verify the results. Covering Algorithm operates by adding tests into the rules that is under construction to create a rule with maximum accuracy. It involves finding an attribute to split on but the criterion for the best attribute is an attribute-value pair to maximize the probability of the desired classification.

III. RESULTS AND DISCUSSION

A new system call classification is produced after reviewing the full pattern of the system calls that generated by the application based on the classification of system call for call logs financial exploitation that produced from the previous study by [11]. Total patterns of system call that produced from 5560 samples are 206 patterns. Table II indicates examples of 20 patterns that were generated based on the system calls for the applications and it represents by number and alphabet to create a pattern. These system calls are related with call logs. These numbers and alphabets representation are used in Table III. Among these 68 system calls, a few system calls have been identified to use some of the calls for exploitation and these identified system calls are used as the input to generate patterns in Table III.

A new classification system call for call logs exploitation is developed based on 206 identified patterns. To validate these patterns, a testing was carried out by using 500 anonymous dataset gathered from Google Play Store from different game, tools, personalisation and antivirus applications. The accuracy of the system call classification for call logs was evaluated and the result of the testing dataset is presented in Table IV. There are 20 mobile applications (apps) that matched with the proposed patterns which consist of 5 different categories which are 6 apps for games, 4 apps for tools, 8 apps for games, 1 app for communication and 1 app for personalization. The names for the matched apps are written as anonymous and sanitized to avoid conflict of interest with any parties.

TABLE II
SYSTEM CALLS FOR MOBILE APPLICATIONS

No	System Call	No	System Call
z1	clock_gettime	z35	sendmsg
z2	epoll_wait	z36	socket
z3	Recvfrom	z37	bind
z4	Sendto	z38	getsockname
z5	Futex	z39	unlinkat
z6	Gettimeofday	z40	renameat
z7	Writev	z41	madvise
z8	getuid32	z42	pwrite
z9	Read	z43	fdatasync
z10	Ioctl	z44	getdents64
z11	Write	z45	setsockopt
z12	Close	z46	getpriority
z13	Openat	z47	lseek
z14	mmap2	z48	_llseek
z15	Newfstatat	z49	setpriority
z16	Mprotect	z50	pipe2
z17	Dup	z51	socketpair
z18	fcntl64	z52	readlinkat
z19	epoll_ctl	z53	nanosleep
z20	Munmap	z54	getrlimit
z21	Pread	z55	wait4
z22	rt_sigprocmask	z56	brk
z23	sched_yield	z57	fchown32
z24	Getsockopt	z58	getpid
Z25	Ppoll	z59	gettid
z26	Clone	z60	lstat64
z27	rt_sigreturn	z61	recv
z28	Faccessat	z62	stat64
z29	fstat64	z63	sigprocmask
z30	Prctl	z64	cacheflush
z31	Fchmodat	z65	select
z32	Fsync	z66	umask
z33	Mkdirat	z67	pipe
z34	Connect	z68	fork

TABLE III.
20 PATTERNS GENERATED

No	*Pattern*
R1	z4+z5+z8+z10+z11+z12+z14+z17+z20+z56+z58+z59
R2	z3+z27+z59
R3	z3+z4+z5+z21
R4	z3+z4+z5+z6+z7+z8+z10+z11+z12+z13+z14+z16+z17 +z18+z19+z20+z21+z26+z28+z41+z56+z58+z59+z62
R5	z3+z4+z5+z6+z7+z8+z10+z11+z12+z13+z14+z16+z17 +z18+z20+z21+z28+z32+z56+z58+z59+z62+z66
R6	z3+z4+z5+z6+z7+z8+z10+z11+z12+z13+z14+z16+z17 +z18+z20+z21+z28+z39+z40+z42+z56+z57+z58+z59+ z62+z66
R7	z3+z4+z5+z6+z7+z8+z10+z11+z12+z13+z14+z16+z17 +z18+z20+z21+z28+z42+z56+z58+z59+z62+z66
R8	z3+z4+z5+z6+z7+z8+z10+z11+z12+z13+z14+z16+z17 +z18+z20+z21+z28+z56+z58+z59+z62+z66
R9	z3+z4+z5+z6+z7+z8+z10+z11+z12+z13+z14+z16+z18 +z20+z26+z34+z36+z37+z41+z45+z56+z58+z59+z62
R10	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+

ISBN: 978-988-14047-4-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

	z17+z18+z19+z20+z21+z23+z24+z26+z27+z28+z31+z32+z33+z34+z36+z37+z38+z39+z40+z41+z42+z43+z44+z45+z46+z47+z48+z56+z57+z58+z59+z60+z62+z66
R11	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z24+z26+z28+z31+z32+z33+z34+z36+z39+z40+z41+z44+z45+z47+z56+z58+z59+z60+z62
R12	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z24+z26+z28+z31+z32+z34+z36+z37+z38+z39+z40+z42+z43+z45+z47+z56+z57+z58+z59+z60+z62+z66
R13	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z24+z26+z28+z36+z37+z38+z41+z44+z45+z47+z56+z58+z59+z60+z62
R14	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z27+z28+z31+z32+z33+z34+z36+z37+z39+z40+z41+z42+z43+z44+z45+z47+z48+z56+z57+z58+z59+z60+z62+z66
R15	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z27+z28+z31+z32+z33+z34+z36+z37+z39+z40+z41+z45+z56+z58+z59+z60+z62
R16	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z27+z28+z34+z36+z37+z41+z45+z47+z56+z58+z59+z62
R17	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z28+z31+z32+z33+z39+z40+z41+z42+z43+z56+z57+z58+z59+z60+z62+z66
R18	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z28+z31+z32+z33+z39+z40+z41+z47+z56+z58+z59+z60+z62
R19	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z28+z31+z32+z33+z39+z40+z41+z56+z58+z59+z60+z62
R20	z3+z4+z5+z6+z7+z8+z9+z10+z11+z12+z13+z14+z16+z17+z18+z19+z20+z21+z23+z26+z28+z31+z32+z33+z41+z47+z56+z58+z59+z60+z62

TABLE IV
20 APPLICATIONS MATCHED WITH THE PROPOSED PATTERNS

Testing Dataset	Category Type
A1	Games
A2	Games
A3	Games
A4	Games
A5	Games
A6	Games
A7	Tools
A8	Tools
A9	Tools
A10	Games
A11	Games
A12	Games
A13	Games
A14	Games
A15	Games
A16	Games
A17	Games
A18	Tools
A19	Communication
A20	Personalisation

IV. CONCLUSION

This research has successfully developed a new system call classification for exploiting call logs and very beneficial in identifying mobile malware exploitation in mobile applications. In future, this new system calls classification and patterns can be used as an input to develop a new model to detect mobile attacks exploitation via call logs. Furthermore, this research can be used as guidance for other researcher to extend their work in mobile malware analysis.

REFERENCES

[1] Tara Seals. (2016, 22 Jan). Android App Stores Drenched with Malware, [Online]. Available: http://www.infosecurity-magazine.com/news/android-app-stores-drenched-with/

[2] Dan Goodin. (2016, 8 Jul). 10 million Android phones infected by all-powerful auto-rooting apps. [Online]. Available: http://arstechnica.com/security/2016/07/virulent-auto-rooting-malware-takes-control-of-10-million-android-devices/ (Accessed 9 August 2016).

[3] Andrew Tarantola (2015, 10 Jan), Stagefright by now spreads using maliciouc audio files. [Online]. Available: https://www.engadget.com/2015/10/01/stagefright-bug-now-spreads-through-malicious-audio-files/(Accessed 9 August 2016).

[4] F. Tchakounté, P. Dayang. (2013). System Calls Analysis of Malwares on Android. *International Journal of Science and Technology*, Volume 2 No. 9, September, [Online]. Available: www.journalofsciencestechnology.org/archive/2013/sep_vol/5513797455.pdf(Accessed 9 August 2016).

[5] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. Proceedings of the Second ACM Conference on Data and Application Security and Privacy - CODASKY '12.

[6] I. Burguera, U. Zurutuza & S.N. Tehrani.(2011). Crowdroid : Behavior-Based Malware Detection System for Android, SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices pp. 15–25]

[7] Y. J. Ham and H.-W. Lee. (2014). "Detection of Malicious Android Mobile Applications Based on Aggregated System Call Events," Int. J. Comput. Commun. Eng., vol. 3, no. 2, pp. 149–154.

[8] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, (2015). "DroidKin : Lightweight Detection of Android Apps Similarity." Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Volume 152, pp 436-453.

[9] Almin, S. B., & Chatterjee, M. (2015). A Novel Approach to Detect Android Malware. Procedia Computer Science, 45, 407-417. [Online].Available: http://www.sciencedirect.com/science/article/pii/S1877050915004135 (Accessed 9 August 2016).

[10] Sahs, J., & Khan, L. (2012). A Machine Learning Approach to Android Malware Detection. 2012 European Intelligence and Security Informatics Conference, pp 141-147.[Online] Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6298824&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6298824 (Accessed 9 August 2016).

[11] H. Albanna, M. M. Saudi, N. Basir (2015). A systematic Review Analysis of Root Exploitation for Mobile Botnet Detection. Advanced Computer and Communication Engineering Technology, Lecture Notes in Electrical Engineering. Volume (362). pp 113-122.

[12] Hitesh Dharmdasani (2014). Android.Hehe: Malware Now Disconnects Phone Calls.[Online] Available: https://www.fireeye.com/blog/threat-research/2014/01/android-hehe-malware-now-disconnects-phone-calls.html

[13] Peter Coogan (2014). DroidJack RAT: A Tale of How Budding Enterpreneurism Can Turn to Cybercrime. [Online] Available: http://www.symantec.com/connect/blogs/droidjack-rat-tale-how-budding-entrepreneurism-can-turn-cybercrime (Accessed 9 August 2016).

ISBN: 978-988-14047-4-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)