Mobile Malware Classification via System Calls and Permission for GPS Exploitation

Madihah Mohd Saudi
Faculty of Science and Technology (FST),
Universiti Sains Islam Malaysia (USIM),
Bandar Baru Nilai, 71800 Nilai,
Negeri Sembilan, Malaysia

Muhammad 'Afif b. Husainiamer
Faculty of Science and Technology (FST),
Universiti Sains Islam Malaysia (USIM),
Bandar Baru Nilai, 71800 Nilai,
Negeri Sembilan, Malaysia

***Abstract*—Now-a-days smartphones have been used worldwide for an effective communication which makes our life easier. Unfortunately, currently most of the cyber threats such as identity theft and mobile malwares are targeting smartphone users and based on profit gain. They spread faster among the users especially via the Android smartphones. They exploit the smartphones through many ways such as through Global Positioning System (GPS), SMS, call log, audio or image. Therefore to detect the mobile malwares, this paper presents 32 patterns of permissions and system calls for GPS exploitation by using covering algorithm. The experiment was conducted in a controlled lab environment, by using static and dynamic analyses, with 5560 of Drebin malware datasets were used as the training dataset and 500 mobile apps from Google Play Store for testing. As a result, 21 out of 500 matched with these 32 patterns. These new patterns can be used as guidance for all researchers in the same field in identifying mobile malwares and can be used as the input for a formation of a new mobile malware detection model.**

Keywords—Mobile malware; Global Positioning System (GPS) exploitation; system call; permission; covering algorithm; static and dynamic analyses

I. INTRODUCTION

Currently, Android smartphone is the most and widely used worldwide and many new mobile malwares are designed to attack it. Mobile malwares is defined as malicious software that is built to attack mobile phone or smartphone system without the owner consent. Examples of the mobile malwares are Slembunk and Santa Claus, where they are able to collect sensitive and confidential information and control smartphone with root exploitation. They tarnish the infected victim reputation and have caused loss of money, productivity and confidential information. Furthermore, McAffee has also reported that 37 million of malwares have been detected in apps stores in year 2016.[1] Apart from SMS, call log, audio and picture exploitation, Global Positioning System (GPS) has been used by many attackers to exploit smartphones. Through GPS, attackers know the victims' details such as satellite information and every movement can be monitored by them. In early year 2017, Google has released a patch (CVW-2016-8467) to overcome security vulnerabilities related with GPS exploitation in Nexus 6 and 6P phones.[2] Currently, not much work has been done to detect GPS exploitation in smartphone. Therefore, this paper objective is to detect mobile malware attacks for GPS exploitation based on system call and permission patterns. A covering algorithm is used as a basis for the proposed patterns. Then the proposed patterns are evaluated to prove its effectiveness.

This paper is organized as: Section 2 presents related work on mobile malware architecture, features and detection techniques. Section 3 describes the methodology used in this research. Section 4 presents the results of experiment carried out in this research. Section 5 includes the summary and potential future work of this paper.

II. RELATED WORK

There are many ways how mobile malwares can be categorized. Work done by Altaher classified android malware based on weighted bipartite graph [1]. He used API and permission for the classification but the dataset used for the experiment only limited to 500 dataset. A bigger and more recent dataset would be a good improvement for this work. As for Feizollah and colleagues, they used feature selection for mobile malwares features extraction [2]. These are based on four main features which are static, dynamic, hybrid and application metadata features. The paper provides a comprehensive review on feature selection for mobile malwares and it is used as guidance for our experiment in this paper. Hybrid feature which combines static and dynamic analyses has been applied due to its comprehensive and systematic feature. System call and permission that are related with GPS exploitation have been extracted and categorized in different patterns and details explained in Section 4 in this paper. Work by Manuel and colleagues also used hybrid feature in their experiment [3]. While works by [4]-[6] used static analysis only, which would give a better a performance if dynamic analysis is integrated in future (hybrid technique).

Apart from that, few research papers by [7]-[9], they discussed about Location Based Services (LBS) or GPS usage for Android smartphone. As for Singhal and Sungkla, they discussed about the implementation of LBS to give multiple

[1] B. Snell, "Mobile threat report what's on the horizon for 2016", 2016. [Online]. Available: https://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2016.pdf. [Accessed: 30- May- 2017]

[2] Tom Mendelsohn, "Google plugs severe android vulnerability that exposed devices to spying | Ars Technica," 2017. [Online]. Available: https://arstechnica.com/security/2017/01/google-plugs-severe-android-bootmode-vulnerability/. [Accessed: 30-May-2017].

services to the user based on their location through Google Web Services and Walk Score Transit APIs on Android. While Ma and colleagues, have developed a tool called as Brox to detect location information leakage in Android by integrating static analysis and Vanjire and colleagues have developed an Android application to locate nearest friends and family members location. There are also many works related to Android malwares analysis such as by [2], [6], [10]-[13]. However, none of the existing works discuss in detailed on how to detect and overcome GPS exploitation for smartphone. This is among the challenges for future work.

III. METHODOLOGY

The dynamic and static analyses and classification of GPS exploitation for system call and permission are summarized as in Fig. 1. The experiment was conducted in a controlled lab environment as illustrated in Fig. 2. No outgoing network connection is allowed to avoid any spread of the mobile malwares.

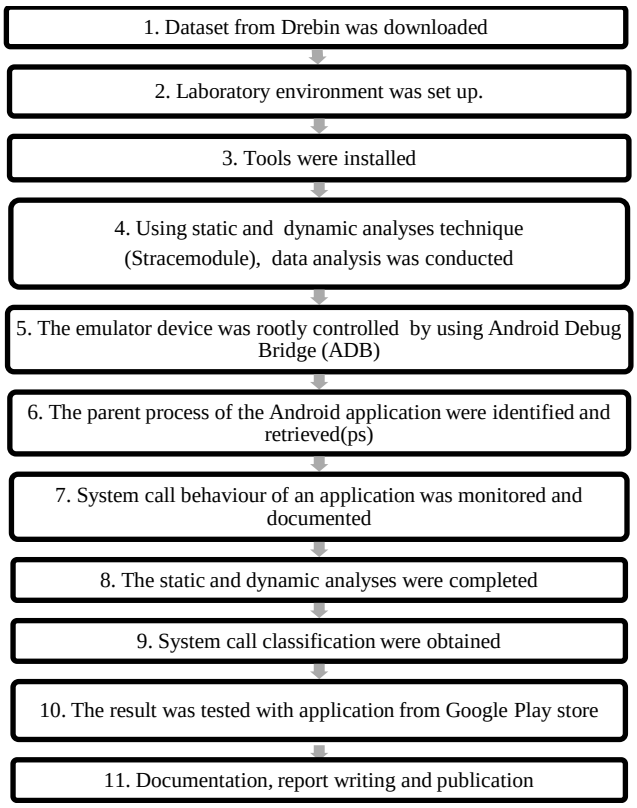

Fig. 1. Research processes.

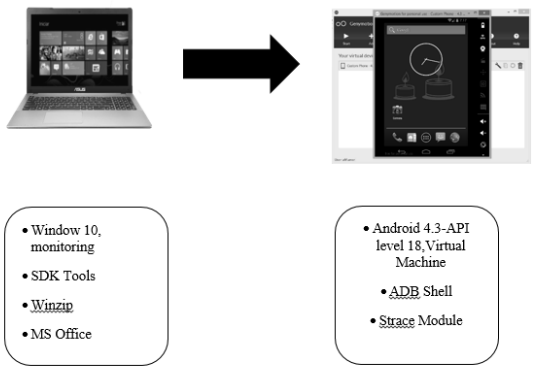

Fig. 2. Lab architecture.

TABLE. I. SOFTWARES USED

Software	Function
Genymotion	Android emulator
Microsoft Excel	Display log dataset in xlsx format Tabulate result recorded
WinZip	Unzip compressed file
ApkTool	Decompile apk resource file into a folder
Strace	Learn application behavior effectively through system calls
Android SDK	Conduct the dynamic analysis
Android Studio	Build application

Table 1 displays the softwares used for the experiment. For this research, the training dataset consists of 179 different types of mobile malwares from 5560 Drebin dataset [4]. While for the testing, 500 mobile applications (apps) have been randomly selected from Google Play Store. The Drebin dataset includes all dataset from the Android Malware Genome Project. It is among the largest malware dataset, free and widely used by many researchers such as by [2], [6], [10]-[13].

The dynamic analysis was used to capture the system call while static analysis was used to capture permission. Then all the extracted system calls and permissions were classified by using covering algorithm. For the dynamic analysis, the apk was installed in Genymotion and being controlled by Android Debug Bridge (ADB). Then the running processes and system calls of the apk were identified and extracted. Fig. 3 displays an example of a screen shot for the system calls captured and Fig. 4 displays an example of a screen shot for the permissions captured. As for the static analysis, the permissions were extracted in the Genymotion where Dexplorer being installed inside it.

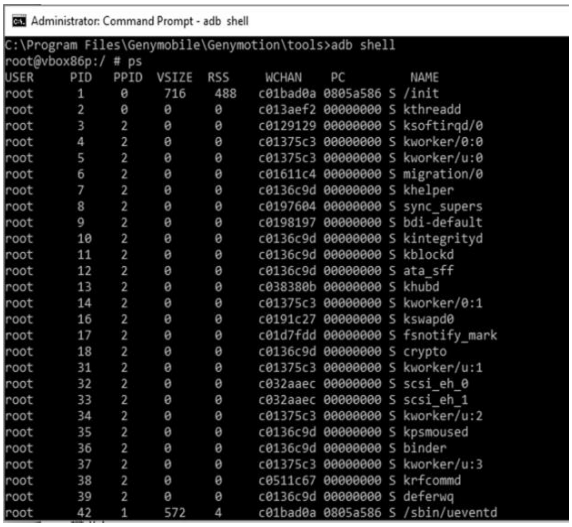

Fig. 3. Screenshot of system call captured.

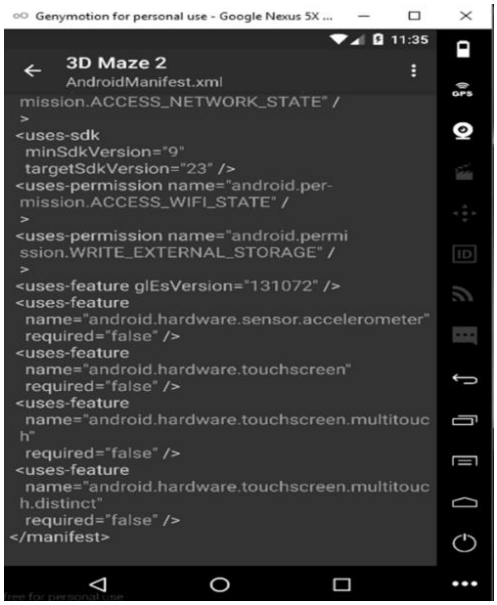

Fig. 4. Screenshot of permissions captured.

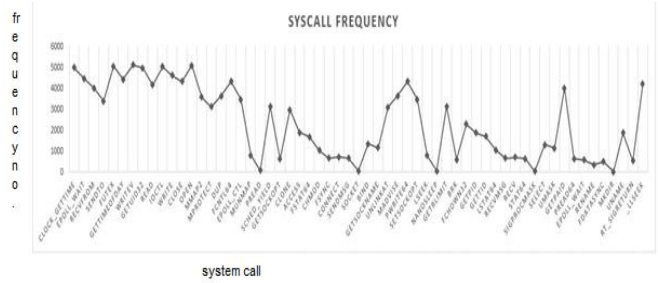

Fig. 5. System calls frequency.

Fig. 5 displays an example of the system calls frequency. Once all the permissions and system calls have been extracted, percentage of occurrence and covering algorithm were applied. These are crucial to verify the extracted dataset and to produce pattern. The percentage of occurrence is developed to compare the similarity between the extracted system calls and permissions. This is useful to avoid redundancy. Each of the system call occurrence is written as 1 to indicate the presence of the system call and 0 for vice versa. Then, the total of the presence and absence of the system calls and permissions were calculated and being compared with the existing dataset.

Once above steps are completed, the output became the input for the covering algorithm. The covering algorithm is used to generate system call and permission pattern for each apk. It identifies rules that have been set by the researchers. In this experiment, specific to general rule induction for covering algorithm has been applied as the following:

1) The extracted system calls and permissions are being picked up and generalized by repeatedly dropping condition.

2) If all the system calls and permissions covered by the set rule, then removed it and continue until all the system calls and permissions are covered.

3) When dropping the condition, make sure to choose the maximize rule coverage.

IV. FINDINGS

Thousands of system calls and permissions have been extracted, but the focus of this paper is on GPS exploitation. There are 58 system calls and 41 permissions out of 5560 samples that have been discovered that could be used together with genuine system calls for GPS exploitation. These system calls representation are shown in Table 2 and permissions representations are shown in Table 3.

TABLE. II. SYSTEM CALLS REPRESENTATION

Nominal Data	System call	Nominal Data	Systemcall
m1	clock_gettime()	m32	getsockname()
m2	epoll_wait()	m33	unlinkat()
m3	recvfrom()	m34	madvise()
m4	sendto()	m35	pwrite64()
m5	futex()	m36	setsockopt()
m6	gettimeofday()	m37	lseek()
m7	writev()	m38	nanosleep()
m8	getuid32()	m39	getrlimit()
m9	read()	m40	brk()
m10	ioctl()	m41	fchown32()
m11	write()	m42	getpid()
m12	close()	m43	gettid()
m13	open()	m44	lstat64()
m14	mmap2()	m45	recvmsg()
m15	mprotect()	m46	recv()
m16	dup()	m47	stat64()
m17	fcntl64()	m48	sigprocmask()
m18	epoll_ctl()	m49	select()

Nominal Data	System call	Nominal Data	Systemcall
m19	munmap()	m50	umask()
m20	pread()	m51	getpaid()
m21	sched_yield()	m52	pread64()
m22	getsockopt()	m53	rename()
m23	clone()	m54	fdatasync()
m24	access()	m55	mkdir()
m25	fstat64()	m56	uname()
m26	chmod()	m57	rt_sigreturn()
m27	fsync()	m58	_llseek()
m28	connect()		
m29	sendmsg()		
m30	socket()		
m31	bind()		

TABLE. III. FORTY-ONE PERMISSIONS

Nominal Data	Permission	Nominal Data	Permission
p1	access_course_location	p22	install_packages
p2	access_fine_location	p23	install_shortcut
p3	access_gps	p24	internet
p4	access_location_extra_commands	p25	kill_background_process
p5	access_network_state	p26	modify_audio_setting
p6	access_wifi_state	p27	read_calendar
p7	battery_stat	p28	read_call_log
p8	bluetooth	p29	read_contact
p9	bluetooth_admin	p30	read_external_storage
p10	call_phone	p31	read_logs
p11	camera	p32	read_phone_state
p12	change_network_state	p33	read_settings
p13	change_wifi_multicast_state	p34	read_sms
p14	change_wifi_state	p35	receive_boot_complete
p15	clear_app_cache	p36	receive_mms
p16	control_location_updates	p37	receive_sms
p17	delete_packages	p38	record_audio
p18	disable_keyguard	p39	restart_packages
p19	expand_status_bar	p40	write_external_storage
p20	get_accounts	p41	write_settings
p21	get_tasks		

TABLE. IV. SIX TOP PERMISSIONS USED TO EXPLOIT GPS

ACCESS_COURSE_LOCATION
ACCESS_FINE_LOCATION
GET_ACCOUNTS
READ_EXTERNAL_STORAGE
READ_PHONE STATE
WRITE_EXTERNAL_STORAGE

Table 4 shows permission classification that mostly used together with system call to exploit GPS that have been extracted from the Drebin dataset. Through dynamic analysis, numerous system calls per application have been encountered until the execution was stopped. Based on system calls presence during dynamic analysis, logs of dataset were recorded.

Table 5 shows the top 10 system calls classification that widely used with permission and system call to exploit GPS that have been extracted from Drebin dataset.

Table 6 shows list of patterns which have been produced based on mostly used for GPS exploitation.

TABLE. V. TOP TEN SYSTEM CALLS USED FOR GPS CONNECTION

chmod() epoll_wait() ioctl() read() access()	socket() bind() connect() recv() writev()

TABLE. VI. THIRTY-TWO PATTERNS FOR POSSIBLE GPS EXPLOITATION

Pattern Representation	Pattern
GPS1	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26
GPS2	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7
GPS3	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28
GPS4	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28+m30
GPS5	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28+m30+m31
GPS6	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28+m30+m31+m46
GPS7	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28+m30+m46
GPS8	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28+m31
GPS9	p1+p2 +p20+p30+p32+p40+

	m2+m9+m10+m24+m26+m7+m28+m31+m46
GPS10	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m28+m46
GPS11	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m30
GPS12	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m30+m31
GPS13	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m30+m31+m46
GPS14	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m30+m46
GPS15	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m31
GPS16	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m31+m46
GPS17	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m7+m46
GPS18	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28
GPS19	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m30
GPS20	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m30+m31
GPS21	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m30+m31+m46
GPS22	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m30+m46
GPS23	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m31
GPS24	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m31+m46
GPS25	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m28+m46
GPS26	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m30
GPS27	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m30+m31
GPS28	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m30+m31+m46
GPS29	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m30+m46
GPS30	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m31
GPS31	p1+p2 +p20+p30+p32+p40+ m2+m9+m10+m24+m26+m31+m46
GPS32	p1+p2+p20+p30+p32+p40+ m2+m9+m10+m24+m26+m46

TABLE. VII. PERCENTAGE OF APPLICATIONS THAT MATCH WITH SYSTEM CALLS AND PERMISSION BASED ON GPS EXPLOITATION

Pattern	**Google play applications**	**Application name**	**Application types**	**Percentage**
GPS 1	21	A1	Game	4.2%
		A2	Downloader	
		A3	Game	
		A4	Game	
		A5	Entertainment	
		A6	Game	
		A7	Music	
		A8	Location	
		A9	Launcher	
		A10	Game	
		A11	Education	
		A12	Game	
		A13	Entertainment	
		A14	Communication	
		A16	Map	
		A17	Weather	
		A18	Travel	
		A19	Browser	
		A20	Map	
		A21	Map	
		A22	Comic	
GPS 2	21	A23	Game	4.2%
		A24	Downloader	
		A25	Game	
		A26	Game	
		A27	Entertainment	

		A28	Game	
		A29	Music	
		A30	Location	
		A31	Launcher	
		A32	Game	
		A33	Education	
		A34	Game	
		A35	Entertainment	
		A36	Communicatio n	
		A37	Map	
		A38	Weather	
		A39	Travel	
		A40	Browser	
		A41	Map	
		A42	Map	
		A43	Comic	
GPS 3	1	A45	Downloader	0.2%
GPS 4	1	A46	Downloader	0.2%
GPS 5	0	None	0	0%
GPS 6	0	None	0	0%
GPS 7	0	None	0	0%
GPS 8	0	None	0	0%
GPS 9	0	None	0	0%
GPS 10	0	None	0	0%
GPS 11	2	A47	Games	0.4%
		A48	Downloader	
GPS 12	0	None	0	0%
GPS 13	0	None	0	0%
GPS 14	0	None	0	0%
GPS 15	0	None	0	0%
GPS 16	0	None	0	0%
GPS 17	0	None	0	0%
GPS 18	1	A49	Downloader	0.2%
GPS 19	2	A50	Game	0.4%
		A51	Downloader	
GPS 20	0	None	0	0%
GPS 21	0	None	0	0%
GPS 22	0	None	0	0%
GPS 23	0	None	0	0%
GPS 24	0	None	0	0%
GPS 25	0	None	0	0%
GPS 26	2	A52	Game	0.4%
		A53	Downloader	
GPS 27	0	None	0	0%
GPS 28	0	None	0	0%
GPS 29	0	None	0	0%
GPS 30	0	None	0	0%
GPS 31	0	None	0	0%
GPS 32	0	None	0	0%

From 32 proposed patterns for potential GPS exploitation, only 21 of them which were downloaded from Google Play Store matched with our proposed patterns as summarized in Table 7.

Then from Table 7, the categories of these 21 apps are summarized in Table 8.

TABLE. VIII. CATEGORIES OF THE MATCHED MALICIOUS APPLICATIONS

No	Malicious Application	Type
1	Apps1	Game
2	Apps2	Downloader
3	Apps3	Games
4	Apps4	Game
5	Apps5	Entertainment
6	Apps6	Game
7	Apps7	Music
8	Apps8	Location
9	Apps9	Launcher
10	Apps10	Game
11	Apps11	Education
12	Apps12	Game
13	Apps13	Entertainment
14	Apps14	Communication
15	Apps15	Map
16	Apps16	Weather
17	Apps17	Travel
18	Apps18	Browser
19	Apps19	Map
20	Apps20	Map
21	Apps21	Comic

V. CONCLUSION

Based on the analysis results in this paper, it can be concluded that each of the executed mobile application has its own system call and permission. GPS has been identified as one of the features and has been used for different purposes. Thirty-two possible patterns for GPS exploitation of system calls and permissions combination are presented in this paper. Without users' consent, their confidential information especially that is related with their location or GPS can be easily exploited by the attackers. Thus based on 21 mobile apps that matched with our patterns, it is proven that GPS feature in the Android smartphone can be exploited by android mobile malware through permission and system call. For future work, this research can be used as guidance for other researchers to extend their work with the same interest and domain. These 32 patterns can be used as a database and input for the formation of a new model to detect mobile attacks exploitation via GPS. Furthermore, automatic for system call and permission extraction is another challenge to be tackled in future.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Ministry of Higher Education (MOHE), Malaysia and Universiti Sains Islam Malaysia (USIM) for the support and facilities provided. This research paper is funded under grant: [FRGS/1/2014/ICT04/USIM/02/1].

REFERENCES

[1] A. Altaher, "Using Weighted Bipartite Graph for android malware classification", International Journal of Advanced Computer Science and Applications, vol. 8, no. 4, 2017.

[2] A. Feizollah, N. Anuar, R. Salleh and A. Wahab, "A review on feature selection in mobile malware detection", Digital Investigation, vol. 13, pp. 22-37, 2015.

[3] M. Egele, T. Scholte, E. Kirda and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools", ACM Computing Surveys, vol. 44, no. 2, pp. 1-42, 2012.

[4] D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, "Drebin: effective and explainable detection of android malware in your pocket," Symp. Netw. Distrib. Syst. Secur., no. February, pp. 23–26, 2014. http://doi.org/10.14722/ndss.2014.23247.

[5] H. Kang, J. Jang, A. Mohaisen, and H. K. Kim, "Detecting and classifying android malware using static analysis along with creator information," Int. J. Distrib. Sens. Networks, vol. 11, no. 6, p. 479174, Jun. 2015. http://doi.org/10.1155/2015/479174.

[6] S. Wu, P. Wang, X. Li, and Y. Zhang, "Effective detection of android malware based on the usage of data flow APIs and machine learning," Inf. Softw. Technol., vol. 75, pp. 17–25, Jul. 2016. http://doi.org/10.1016/j.infsof.2016.03.004

[7] M. Singhal and A. Shukla, "Implementation of location based services in android using GPS and web services," Int. J. Comput. Sci. Issues, vol. 9, no. 1, pp. 237–242, 2012.

[8] Ma, S., Tang, Z., Xiao, Q., Liu, J., Duong, T. T., Lin, X., & Zhu, H. (2013). Detecting GPS information leakage in android applications. Global Communications Conference (GLOBECOM), 2013 IEEE, 826–831. http://doi.org/10.1109/GLOCOM.2013.6831175.

[9] S. Vanjire, U. Kanchan, G. Shitole, and P. Patil, "Location based services on smart phone through the android application," Int. J. Adv. Res. Comput. Commun. Eng., vol. 3, no. 1, 2014. Retrieved from http://www.ijarcce.com/upload/2014/january/IJARCCE3B__A_unmesh_Location.pdf.

[10] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. van der Veen, and C. Platzer, "ANDRUBIS -- 1,000,000 apps later: A view on current android malware behaviors," in 2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014, pp. 3–17. http://doi.org/10.1109/BADGERS.2014.7.

[11] Hashim, H. A.-B., M. Saudi, M., & Basir, N. (2015). A systematic review analysis of root exploitation for mobile botnet detection. Lecture Notes in Electrical Engineering, 315, 925–938. http://doi.org/10.1007/978-3-319-07674-4.

[12] A. Karim, R. Salleh, M. K. Khan, T. Schreck, J. Hoffmann, and I. Witten, "SMARTbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications," PLoS One, vol. 11, no. 3, p. e0150077, Mar. 2016. http://doi.org/10.1371/journal.pone.0150077.

[13] Bhatt, M. S., Patel, H., & Kariya, S. (2015). A survey permission based mobile malware detection, 6(5), 852–856.