Go-Detect Application Inspired by Apoptosis to Detect SMS Exploitation by Malwares

Madihah Mohd Saudi, Luqman Amran and Farida Ridzuan

Abstract Nowadays, malware attack mobile phone compared to the computer due to its mobility and extensive usage. The users are being exposed with sophisticated threats that lead to loss of money and confidential information. These threats are inferred by malwares that exploit the mobile applications (apps) vulnerabilities. Five surveillance features in a mobile phone commonly used by the malwares includes Short Message Service (SMS), call log, Global Positioning System (GPS), camera and audio. This paper focuses on the SMS feature and presents a mobile apps called as Go-Detect inspired by Apoptosis to detect SMS exploitation by malwares. There are 16 new SMS Android Package Index (API) classifications that have been developed and used as the input for this app. Apoptosis or known as cell-programmed-death is a concept borrowed from human immunology that has been integrated in this app. It will uninstall and delete the infected apps that matched with the proposed classifications. A total of 5560 Drebin dataset has been used as the training dataset and reverse engineered using static analysis in a controlled lab environment. This app is built by using JAVA. Based on the testing conducted with 50 anonymous mobile apps from the Google Play store, 36% matched with the proposed classification. This new classification and apps can be used as the reference and basis for other researchers to detect malware in a mobile phone.

M. Mohd Saudi (✉) · L. Amran · F. Ridzuan
Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), Bandar Baru Nilai, 71800 Nilai, Negeri Sembilan, Malaysia
e-mail: madihah@usim.edu.my

M. Mohd Saudi · F. Ridzuan
CyberSecurity and Systems (CSS), Institute Science Islam (ISI), Universiti Sains Islam Malaysia (USIM), 71800 Nilai, Negeri Sembilan, Malaysia

A. P. P. Abdul Majeed et al. (eds.), *RITA 2018*, Lecture Notes in Mechanical Engineering, https://doi.org/10.1007/978-981-13-8323-6_9

1 Introduction

The growing of the smartphone and tablet users has caught the attention of cybercriminals who see them as their gold mine. At the present time, there are higher opportunities for successful malware attackers as there are higher numbers of smartphone and tablet users. Mobile malware or malware is defined as malicious software that can exploit user's mobile phone without the user consent. The motivation behind the attack is financially driven, to obtain user's information or simply to cause harm and damage on the device. The mobile malware is designed to disable a mobile device and then allow an irresponsible attacker to remotely control the device or to steal personal information stored on the device. The mobile malwares attack can spread via SMS. Short Message Service or commonly known as SMS is one of the earliest features created for a mobile phone. This service allows mobile phone to send text message from one cell phone to another cell phone. The cybercriminals take the opportunity this service profit-based by sending malware. For example, in year 2018, AndroRAT Trojan is able to exploit root privilege at victim's smartphone and delete and send forged SMS from the exploited smartphone [1]. While Hummingbad virus uses SMS to spread click fraud where once user clicked on the link, she will be infected by the virus which collects personal information and exploits the mobile phone's root. This malware is designed to target a victim mobile device [2]. As for AndroidOS_Smszombie.A it exploits china mobile via SMS payment and camouflage itself as a wallpaper application [3]. Some of the malwares are designed to send unauthorized texts without the user's knowledge or consent. Existing works such as by Mohd Saudi et al. [4], Bose and Shin [5] and Hamandi et al. [6] had highlighted some of the challenges and techniques for SMS exploitation by malwares, which is the urge and the basis formation of this paper.

Therefore, this paper aims are to present the formation of mobile malware API classification for SMS and a mobile application developed to detect API exploitation for SMS. The patterns are based on covering algorithm and the effectiveness of the proposed patterns is then evaluated.

For this paper, Sect. 2 summarizes existing work on mobile malware detection techniques and architecture. Section 3 explains the research, while Sect. 4 describes the experiments results. The summary and potential future work of this paper is included in Sect. 5.

2 Related Work

Jang and Yun categorized and classified the mobile malwares based on similarity matching of malware behaviour profile using several algorithm methods such as K-means clustering, Cipher and Encoding algorithms [7]. Various algorithms were implemented to detect malware in the mobile phone applications [8–12]. There are four main features to complete these works which are application metadata, hybrid,

static and dynamic. These existing works act as our guidance for our experiment in this paper. We applied static analysis in our experiment due to the systematic feature and in Sect. 4, we present Android Package Index (API) that related with SMS exploitation. Work done by Hyun and Kim used hybrid analysis feature [12] which is the combination of two features which are static analysis and dynamic analysis. The static analysis has been used for our work for a better performance. Other than that, few research papers by Balaji [1] have given a lot of helps in the development of the mobile application. This paper provides the ideas and guidance to build the mobile application based on the techniques implemented in this paper. Nonetheless, a tool to detect mobile malwares based on API has been developed by Liu et al. [8]. This tool may be adopted in developing malware detection techniques based upon the API call patterns. Though there are few existing works that are related with malware detection, yet, the challenges for future work will be on how to detect and overcome SMS exploitation for smartphone.

3 Methodology

The research processes include static analysis and classifications of SMS exploitation for API are summarized in Fig. 1.

Prior the formation of the Go-detect mobile application, reverse engineering was conducted which inclusive of static analysis which can be referred in Fig. 1, step 1 until step 10. The 5560 Drebin dataset were being simulated in Genymotion emulator as the training dataset and analysed to extract the processes and API calls from the mobile apps coding. For step 2, as depicts in Fig. 2, the experiment was conducted in a lab environment with no outgoing network connection. Software used for this experiment can be referred in Fig. 2 and Table 1.

For this experiment, the software used as depicts in Table 1. 5560 files in Drebin dataset were used for training purpose and reverse engineered using static analysis. 500 mobile applications for testing were downloaded from Google Play Store randomly. The Drebin dataset is inclusive of Android Malware Genome dataset and has been used as benchmarked dataset by many researchers such as by Laura [2], Yusof et al. [13], Li et al. [14], Lindorfer et al. [15] and Talha et al. [16].

The APIs are captured via the static analysis as summarizes in Fig. 1. For static analysis, the features of API calls were extracted by using a reverse engineering tool which is Apktool. The extracted APIs were then classified by using covering algorithm. Figure 3 displays an example of a screen shot for extracted apk files which is then decompiled into folder. Figure 4 shows the API captured and extracted in Genymotion, where ShowJava is installed via static analysis.

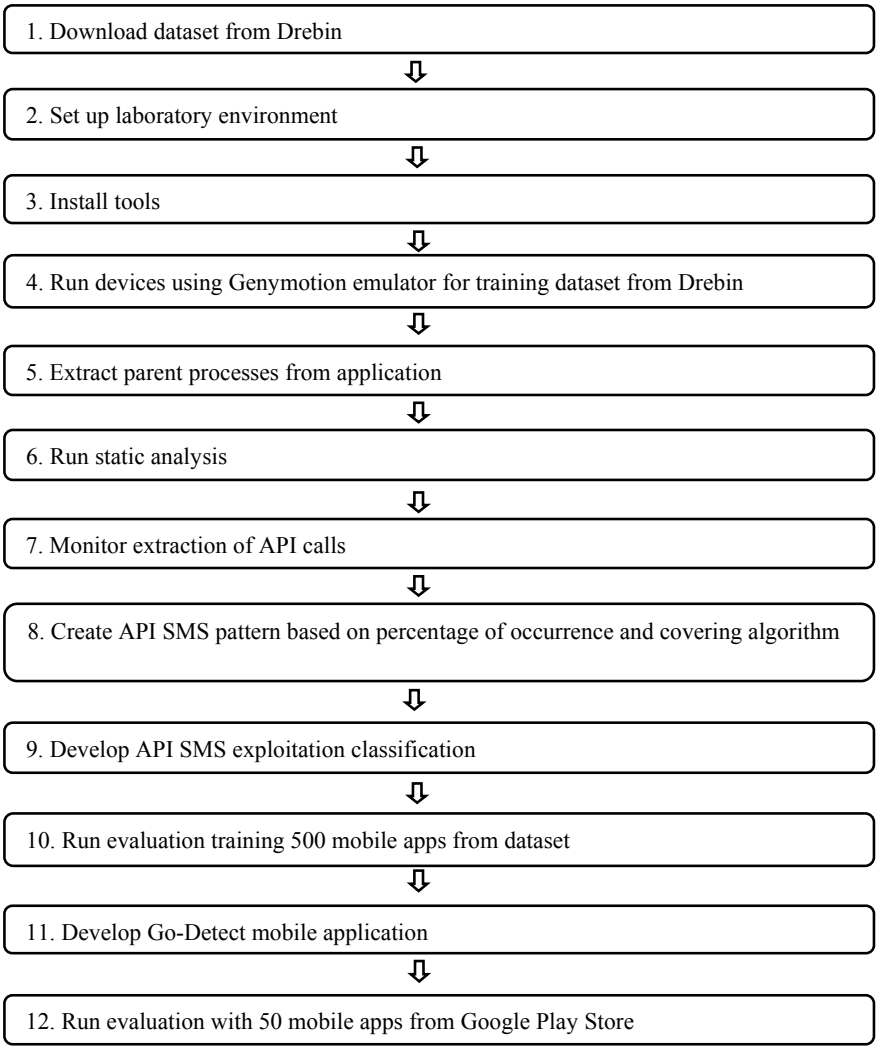

Fig. 1 Experiment processes

Then from the extracted API calls, the most relevant API calls with SMS exploitation were calculated based on frequency which can be referred in Fig. 5. Later pattern and classification of API SMS exploitation is developed by using covering algorithm. It is based on the most relevant function that could be used for exploitation. As a result, 16 new SMS API classification have been developed . Then 500 random

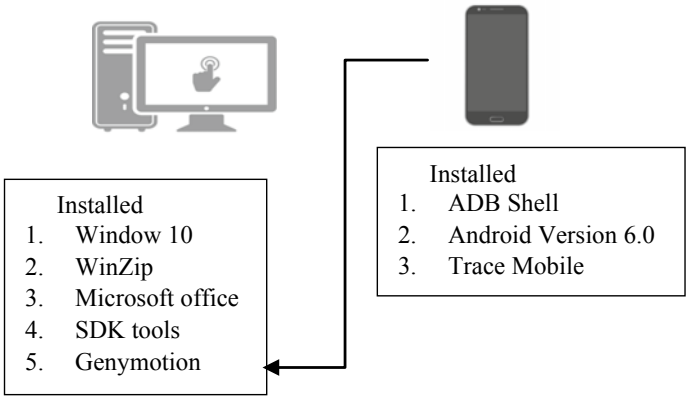

Fig. 2 Lab architecture

Table 1 Software used

Software	Function
Genymotion	Acts as emulator for Android
Microsoft Excel	To record and display log dataset for further pattern development
WinZip	Tool to unzip file
Apk tool	To decompile apk file
Android SDK	To conduct the static analysis
Android studio	To build application

mobile apps from Google Playstore were downloaded as the testing dataset. The proposed new classifications were tested with the testing dataset and the results can be referred in Table 5. Basically step 1 until step 10 in Fig. 1, were meant for the classification development which is the novelty and main contribution for this paper. Figure 5 displays the API frequency.

The covering algorithm and percentage of occurrence are applied to the extracted APIs prior the pattern formation and to avoid pattern redundancy. The presence of the API occurrence is written as 1 and 0 if it is absence. After that, the total calculation for the presence and absence of the APIs are counted and compared with the existing dataset. These values are the input for the covering algorithm, where it generates API pattern for each app. For covering algorithm , let M be a vector contains set of all

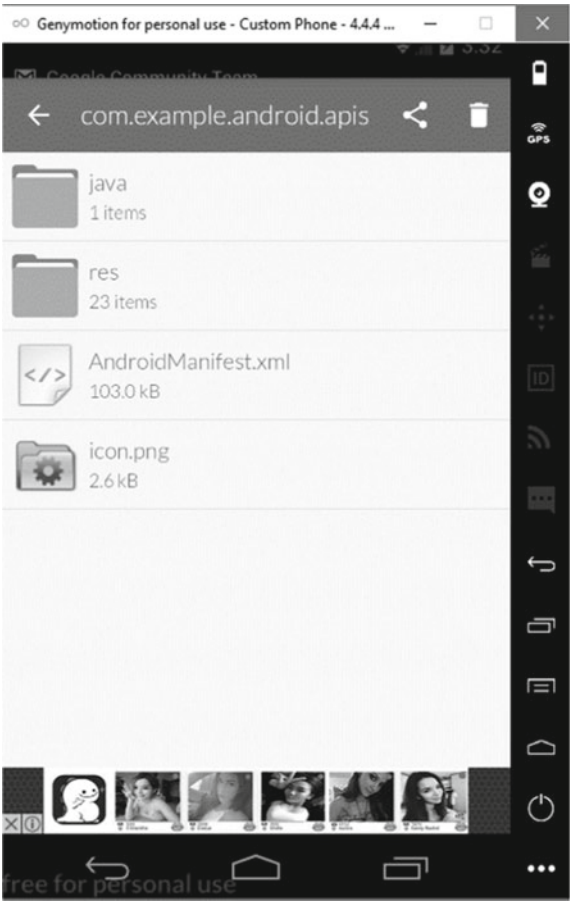

Fig. 3 Screenshot of apk file decompiled

API calls. For every ith application in the training dataset, Mi = {r1, r2, r3,…,rj}, where r can be referred in Formula 1.

$$r_j = \begin{cases} 1, \; if \; permission \; jth \; exist \\ 0, \; otherwise \end{cases} \tag{1}$$

Then for real-time implementation, the proposed 16 new SMS API classifications have been used as the database and input for Go-detect mobile app. For evaluation of the proposed mobile app, 50 random dataset from Google Play store were tested. The result for the evaluation can be referred in Table 6.

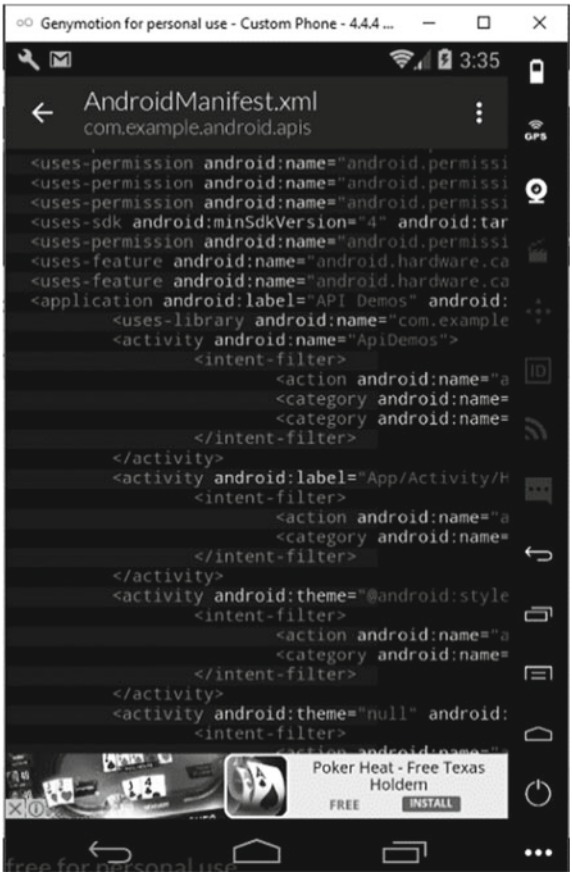

Fig. 4 Screenshot of API captured

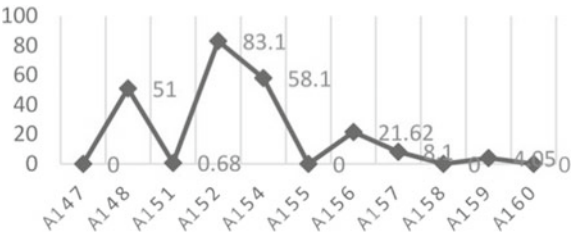

Fig. 5 API frequency

4 Findings and Discussion

The findings of this paper will be reported into three sections, which are (1) API pattern produced for potential SMS exploitation, (2) evaluation of mobile malware API classification for SMS, (3) development of Go-Detect Application and (4) application evaluation.

4.1 API Pattern for Potential SMS Exploitation

Based on the 5560 files in the Drebin dataset, 191 API from different types of mobile malware are detected as malicious. Table 2 shows the function of the API and its representation. As this paper focuses on SMS exploitation, Table 3 presents the APIs that is related to SMS.

Table 3 shows the classification of 11 APIs related with SMS extracted from the Drebin dataset. The representation above are used to develop the pattern based on percentage of occurrence and covering algorithm. Table 4 shows 16 patterns created based on the APIs frequently used for SMS exploitation.

Table 2 API representation

Nominal data	String	Nominal data	String
A1	addAcoount	A96	start
A2	addAccountExplicity	A97	stop
A3	blockingGetAuthToken	A98	setAudioSource
A4	getAccounts	A99	setVideoSource
A5	Get AccountsByType	A100	setActualDefaultRingtoneUri
A6	getAuthToken	A101	getActiveNetworkInfo
A7	getPassword	A102	getAllNetworkInfo
A8	invalidateAuthToken	A103	getMobileDataEnabled
A9	peekAuthToken	A104	getNetworkInfo
A10	removeAccount	A105	requestRouteToHost
A11	SetAuthToken	A106	setMobileDataEnabled
A12	setPassword	A107	startUsingNetworkFeature
A13	sendBroadcast	A108	stopUsingNetworkFeature
A14	setContentView	A109	isConnectedOrConnecting
A15	setPersistent	A110	acquire

(continued)

Table 2 (continued)

Nominal data	String	Nominal data	String
A16	startActivity	A111	release
A17	startActivityForResult	A112	addNetwork
A18	startActivityNeeded	A113	disableNetwork
A19	getRecentTasks	A114	disconnect
A20	GetRunningTasks	A115	enableNetwork
A21	killBackgroundProcess	A116	getConfiguredNetworks
A22	restartPackage	A117	getConnectionInfo
A23	reportFailedPasswordAttempt	A118	getDhcpInfo
A24	reportSuccessPasswordAttempt	A119	getScanResults
A25	setActivePasswordState	A120	getWifiState
A26	setTimeZone	A121	isWifiEnabled
A27	dataChanged	A122	reconnect
A28	senKeyDownUpSync	A123	removeNetwork
A29	disableKeyguard	A124	saveConfiguration
A30	reenableKeyguard	A125	setNumAllowedChannels
A31	exitKeyguardSecurely	A126	setWifiEnabled
A32	notify	A127	startScan
A33	sendBroadcast	A128	acquire
A34	startActivity	A129	release
A35	expand	A130	reboot
A36	setBitmap	A131	cancel
A37	setResource	A132	vibrate
A38	suggestDesiredDimensions	A133	clearHistory
A39	bindAppWidgetId	A134	clearSearches
A40	cancelDiscovery	A135	getAllBookmarks
A41	disable	A136	getAllVisitedUrls
A42	enable	A137	addToMyContactsGroup
A43	getAddress	A138	createPersonInMyContactGroup
A44	getBondedDevices	A139	setPhotoData
A45	getState	A140	getLookupUri
API46	isDiscovering	A141	openContactPhotoInputStream
A47	isEnabled	A142	putInt
A48	listenUsingRfcommWithServicesRecord	A143	putLong
A49	startDiscovery	A144	putString
A50	createRfcommSocketToServiceRecord	A145	putInt
A51	getBondState	A146	putString
A52	getName	A147	AddMessages

(continued)

Table 2 (continued)

Nominal data	String	Nominal data	String
A53	getBatteryUsageHint	A148	readYourTextMessages
A54	connect	A149	getOrCreateThreadId
A55	addPeriodicSync	A150	startListening
A56	getMasterSyncAutomatically	A151	sendMultipartTextMessageSMS
A57	getSyncAutomatically	A152	sendTextMessageSMS
A58	openFileDescriptor	A153	isVoiceMailNumber
A59	openInputStream	A154	receiveTextMessagesSMS
A60	openOutputStream	A155	deleteMessage
A61	query	A156	editTextMessage
A62	removePeriodicSync	A157	receiveTextMessageMMS
A63	setIsSyncable	A158	sendMultipartTextMessageMMS
A64	setMasterSyncAutomatically	A159	sendTextMessageMMS
A65	setSyncAutomatically	A160	updateMessage
A66	sendBroadcast	A161	getCellLocation
A67	sendOrderedBroadcast	A162	getDeviceID
A68	sendStickyBroadcast	A163	getDeviceSoftwareVerse
A69	setWallpaper	A164	getLine1Number
A70	startActivity	A165	getNeighboringCellInfo
A71	startServices	A166	getSimSerialNumber
A72	sendBroadcast	A167	getSubscribeID
A73	setWallpaper	A168	getVoiceMailAlphaTag
A74	startActivity	A169	getVoiceMailNumber
A75	addPreferredActivity	A170	Lisyen
A76	clearPackagePrefferedActivities	A171	getCallerInfo
A77	setComponentEnabledSetting	A172	markAsVoiceMail
A78	open	A173	exec
A79	addGpsStatusListener	A174	connect
A80	addNameListener	A175	bind
A81	getBestProvider	A176	getContent
A82	getLastKnownLocation	A177	openConnection
A83	getProvider	A178	openStream
A84	getProviders	A179	connect
A85	isProviderEnabled	A180	getInputStream
A86	requestLocationUpdates	A11	execute
A87	sendExtraCommand	A182	Cipher(AES)
A88	setTestProviderEnabled	A183	Cipher(AES/CBC/PKCS5Padding)

(continued)

Table 2 (continued)

Nominal data	String	Nominal data	String
A89	isBLuetoothScoOn	A184	Cipher(RSA/ECB/PKCS1Padding)
A90	isWiredHeadsetOn	A185	CryptoCipher
A91	setBluetoothScoOn	A186	getPackageInfo
A92	setMode	A187	getSystemService
A93	setSpeakerphoneOn	A188	HttpPost
A94	startBluetoothSco	A189	Obfuscation(Base64)
A95	stopBluetoothSco	A190	sendSMS
A191	system/bin/su		

Table 3 API related with SMS

Nominal data	String	Nominal data	String
A147	AddMessages	A156	editTextMessage
A148	readYourTextMessages	A157	receiveTextMessageMMS
A151	sendMultipartTextMessageSMS	A158	sendMultipartTextMessageMMS
A152	sendTextMessageSMS	A159	sendTextMessageMMS
A154	receiveTextMessagesSMS	A160	updateMessage
A155	deleteMessage		

Table 4 Pattern representation related with SMS

Pattern representation	Pattern
P1	A152
P2	A52 + A154
P3	A148 + A152 + A154
P4	A148 + A152 + A154 + A156
P5	A148 + A152 + A154 + A159
P6	A152 + A154 + A159
P7	A148 + A154
P8	A154 + A157
P9	A148
P10	A148 + A152
P11	A148 + A152 + A156
P12	A148 + A154 + A156 + A157
P13	A148 + A154 + A157
P14	A152 + A154 + A156 + A157
P15	A148 + A156
P16	A152 + A154 + A157

4.2 Evaluation of Mobile Malware API Classification for SMS

As for the testing from 500 mobile apps downloaded from Google Play Store, only 43 mobile apps are matched with the produced patterns. These mobile apps are categorized as high potential for SMS exploitation. This result can be referred in Table 5.

Table 5 Results for the APIs matched with SMS exploitation

Pattern	Number of applications	Application category	Percentage (%)
Pattern1	13	System (3)	2.6
		Wallpaper (1)	
		Medical (1)	
		Education (1)	
		Tool (4)	
		Game (1)	
		Entertainment (1)	
		Photo (1)	
Pattern2	5	Game (1)	1.0
		Social (2)	
		Tool (2)	
Pattern3	2	Network (1)	0.4
		Game (1)	
Pattern4	4	System (2)	0.8
		Game (1)	
		Social (1)	
Pattern5	0	0	0
Pattern6	1	System (1)	0.2
Pattern7	2	Tool (2)	0.4
Pattern8	3	System (1)	0.6
		Tool (1)	
		Wallpaper (1)	
Pattern9	8	Tool (4)	1.6
		Wallpaper (1)	
		System (2)	
		Life Style (1)	

(continued)

Table 5 (continued)

Pattern	Number of applications	Application category	Percentage (%)
Pattern10	0	0	0
Pattern11	2	Game (2)	0.4
Pattern12	0	0	0
Pattern13	1	Game (1)	0.2
Pattern14	1	Book (1)	0.2
Pattern15	1	System (1)	0.2
Pattern16	0	0	0

4.3 Go-Detect Application Development

Based on the pattern produced earlier, an application called Go-Detect is developed. This application will classify each of API in the sample apps installed in the mobile phone. The main function of the application is malware pattern match testing. The interfaces of the application are presented in Fig. 6.

Go-Detect application allows the user to test the vulnerability of an Android application based on the pattern produced earlier. If the generated API of the application matched by pattern produced, users will be prompt that their device may considered as harmful and confidential information might be leaked from this device. Therefore, users are given a choice to uninstall the application. The produced patterns for potential SMS exploitation is then evaluated and compared against 500 mobile apps downloaded randomly from Google Play. As a result, as displays in Table 5, only 43 out of 500 mobile apps matched with the proposed patterns.

Fig. 6 Screenshots of Go-Detect application

Table 6 Result for application evaluation

Pattern	No. of applications	Percentage (%)
Pattern1	4	8
Pattern2	2	4
Pattern3	2	4
Pattern4	1	2
Pattern5	0	0
Pattern6	0	0
Pattern7	0	0
Pattern8	3	6
Pattern9	6	12
Pattern10	0	0
Pattern11	0	0

4.4 Application Evaluation

The application is then evaluated to ensure that it fulfil its functionality to detect vulnerable application based on SMS feature. 50 samples of mobile applications from Google Play Store were downloaded randomly for the functional test. The results of the evaluation are presented in Table 6.

The result shows that 18 out of 50 samples or 36% of the applications were considered as potential application that can be exploited through SMS. Nonetheless, the application has successfully fulfilled its functionality.

5 Conclusion

16 possible patterns for SMS exploitation of API are produced in this paper, which is the main contribution and novelty of this paper. Based on the results presented in which 43 mobile apps are found to be matched with the generated patterns, it is proven that SMS features related with API could be exploited by malwares in the Android smartphone. Therefore, Go-Detect application is developed to allow a vulnerability test to be carried out in an Android application. The application managed to detect 36% of the mobile apps based on the proposed classification. It is concluded that the proposed classification and the developed Go-Detect application could detect malware attack based on the SMS feature.

This result from this paper could be used as reference for other researchers to further work extension in the same area. A new model of API classification for mobile phone based on the proposed classifications developed in this paper. The application promotes secure user environment for Android users.

Acknowledgements The author would like to express their gratitude to Universiti Sains Islam Malaysia (USIM) for the support and facilities provided. This research paper is funded under grant: [PPP/USG-0116/FST/30/11716].

References

1. Balaji N (2018) AndroRAT—a remote access trojan compromise android devices and inject root exploits. https://gbhackers.com/androrat-remote-access-trojan/. Last accessed 28 Aug 2018
2. Laura H (2016) How to tell if your Android phone has the HummingBad malware. https://www.cnet.com/how-to/hummingbad-how-to-tell-if-your-android-phone-has-a-bad-case-of-malware/. Last accessed 28 Aug 2018
3. Bob P (2012) Android malware exploits china mobile SMS payments. http://blog.trendmicro.com/trendlabs-security-intelligence/android-malware-exploits-china-mobile-sms-payments/. Last accessed 28 Aug 2018
4. Mohd Saudi M, Abd Rahman MZ, Mahmud AA, Basir N, Yusoff YS (2016) A new system call classification for android mobile malware surveillance exploitation via SMS message. In: Sulaiman H, Othman M, Othman M, Rahim Y, Pee N (eds) Advanced computer and communication engineering technology. Lecture notes in electrical engineering, vol 362. Springer, Cham
5. Bose A, Shin KG (2006) On mobile viruses exploiting messaging and bluetooth services. In: Securecomm and workshops. Baltimore, MD, pp 1–10. http://doi.org/10.1109/SECCOMW.2006.359562
6. Hamandi K, Chehab A, Elhajj IH, Kayssi A (2013) Android SMS malware: vulnerability and mitigation. In: 27th International conference on advanced information networking and applications workshops, Barcelona, pp 1004–1009. http://doi.org/10.1109/WAINA.2013.134
7. Jang J, Yun J, Mohaisen A, Woo J, Kim HK (2016) Detecting and classifying method based on similarity matching of android malware behavior with profile. SpringerPlus 5:273. https://doi.org/10.1186/s40064-016-1861-x
8. Liu C-H, Zhang Z-J, Wang S-D (2016) An android malware detection approach using Bayesian inference. In: Proceeding of IEEE international conference on computer and information technology, pp 476–483
9. Shamili AS, Bauckhage C, Alpcan T (2010) Malware detection on mobile devices using distributed machine learning. In 2010 20th International conference on pattern recognition (ICPR). IEEE, pp 4348–4351
10. Kaushik P, Jain A (2015) Malware detection techniques in android. Int J Comput Appl 122(17):22–26
11. Mohata VB, Dakhane DM, Pardhi RL (2013) Mobile malware detection techniques. Int J Comput Sci Eng Technol (IJCSET) 4(04), 2229–3345; Choi S, Bijou M, Sun K, Jung E (2015) Int J Inf Educ Technol 5(6), 460–465
12. Saudi MM, Ridzuan F, Basir N, Nabila NF, Pitchay SA, Ahmad IN (2015) Android mobile malware surveillance exploitation via call logs: proof of concept. In: 2015 17th UKSim-AMSS international conference on modelling and simulation (UKSim). IEEE, pp 176–181
13. Yusof M, Mohd Saudi M, Ridzuan F (2017) A new mobile Botnet classification based on permission and API calls In: Seventh international conference on emerging security technologies (EST), pp 122–127

14. Li Z, Sun L, Yan Q, Srisa-an W, Chen Z (2017) DroidClassifier: efficient adaptive mining of application-layer header for classifying android malware. In: Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, vol 198, pp 597–616
15. Lindorfer M, Neugschwandtner M, Weichselbaum L, Fratantonio Y, van der Venn V, Platzer C (2014) ANDRUBIS—1,000,000 apps later: a view on current android malware behaviors. In: Third international workshop on building analysis datasets and gathering experience returns for security pp 3–17
16. Talha KA, Alper DI, Aydin C (2015) apk auditor: permission-based Android malware detection system. Digit Investig 13:1–14