Android Mobile Malware Classification using Tokenization Approach based on System Call Sequence

Intan Nurfarahin Ahmad, Farida Ridzuan, Madihah Mohd Saudi, Sakinah Ali Pitchay, Nurlida Basir and N. F. Nabila

***Abstract*— The increasing number of smartphone over the last few years reflects an impressive growth in the number of advanced malicious applications targeting the smartphone users. Recently, Android has become the most popular operating system opted by users and the most targeted platform for smartphone malware attack. Besides, current mobile malware classification and detection approaches are relatively immature as the new advanced malware exploitation and threats are difficult to be detected. Therefore, an efficient approach is proposed to improve the performance of the mobile malware classification and detection. In this research, a new system call classification with call logs exploitation for mobile attacks has been developed using tokenization approach. The experiment was conducted using static and dynamic-based analysis approach in a controlled lab. System calls with call logs exploitation from 5560 Drebin samples were extracted and further examined. This research paper aims to find the best *n* value and classifier in classifying the dataset based on the new patterns produced. Naïve Bayes classifier has successfully achieved accuracy of 99.86% which gives the best result among other classifiers. This new system call classification can be used as a guidance and reference for other researchers in the same field for security against mobile malware attacks targeted to call logs exploitation.**

***Index Terms*— Android mobile malware, mobile malware classification, system call sequence, tokenization.**

I. Introduction

The invention of smartphone facilitates human daily life. Smartphone that offers multiple functionalities has now becomes a major device for communication. The smartphone also integrates multiple wireless networking technology to support other functionality and services such as social media, GPS, phone call, SMS, MMS, and game play with high graphic quality [1]. A research reported by e-marketer shows the number of smartphone users worldwide continues to increase every year as shown in Figure 1 [2].

Year 2016 shows the number of smartphone users forecasted to reach 2.1 billion. By year 2020, this number is expected to reach around 2.87 billion smartphone users. Moreover, the increasing number of smartphones gives a huge impact to the increasing number of malicious applications [1]. Therefore, protecting smartphone users from malware attack has become a major challenge.

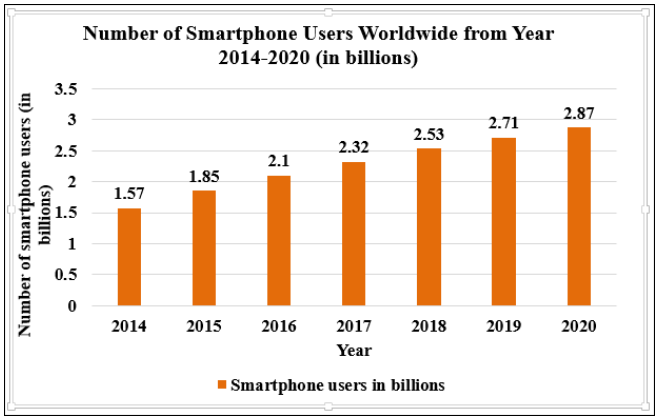

Fig. 1. Number of Smartphone Users Worldwide from Year 2014-2020 (in billions)**.**

Recently, Android is the most popular operating system opted by users [3]. Android users can easily download thousands of applications in markets and these applications allow users to freely conduct any customizations and extensions. Android allow users to install applications from various sources such as Google Playstore, third-party markets, Torrents and direct download [4]. This leaves an open door for attacks on user's security environment hence allows the attackers to embed malicious code into the applications. Ignorant or non-illiterate users could become victims of these attacks by unconsciously executing the malicious applications and finally their devices will be infected by the malware.

Large number of mobile malware attack are reported each day and most of the attack are motivated by profit [1]. On March 2016, a report was released by McAfee Lab which shows more than 12 billion mobile malware cases are reported and it is continuously increasing every year [5]. A recent case was reported on May 2017 where the world's biggest cyberattack (ransomware attack) has hit more than 150 countries and has paralyzed 300,000 machines from

Manuscript received July 16, 2017; revised August 8, 2017. This work was funded by Ministry of Higher Education (Malaysia), FRGS grant: [FRGS/FST/32/50114].

I. F. Ahmad is with the Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), Bandar Baru Nilai, 71800 Nilai, Negeri Sembilan, Malaysia. (e-mail: intan_lavender@yahoo.com).

F. Ridzuan is with the Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), Bandar Baru Nilai, 71800 Nilai, Negeri Sembilan, Malaysia. Phone: +(60)6-7986443; (e-mail: farida@usim.edu.my).

M. M. Saudi, Sakinah Ali Pitchay, Nurlida Basir and N.F. Nabila are with the Faculty of Science and Technology (FST), Universiti Sains Islam Malaysia (USIM), Bandar Baru Nilai, 71800 Nilai, Negeri Sembilan, Malaysia (e-mail: {madihah, sakinah.ali, nurlida, fatin}@usim.edu.my).

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

various global companies such as FedEx, Nissan, and Hitachi [6]. Other than that, Check Point's research team recently discovered an adware known as "Judy". It is a malware found in official Google Playstore which was released by the Korean company named as Kiniwini. Judy communicates using Command and Control (C&C) server. It generates a huge volume of fraudulent advertisement and forces victims to click it [7].

Malware analysis is a popular research area but with many unsolved problems. Many researchers have proposed methods for classification and detection of Android mobile malware such as identification of expert features, system call behaviour, permission, and API usage. Approaches for mobile malware classification and detection can be categorized into two techniques which are static and dynamic-based analysis approach. Generally, static-based approach involves extracting information from the application's manifest file without executing it [8]. Meanwhile, dynamic-based analysis extracts the malware behaviour during its execution in emulator environment. This approach allows researcher to obtain additional, detailed information from the suspected applications [8]. Previous works which conduct dynamic-based analysis approach managed to generate behaviour patterns from malicious applications [9], [10], [11]. However, most of them are still lacking in efficiency and accuracy [12]. Efficiency and accuracy are two important characteristics in performing mobile malware classification and detection, thus a suitable approach is needed in order to optimize the performance [13].

In this paper, a hybrid approach which combines both static-based (permissions features) and dynamic-based (system call sequences features) analysis are implemented. Hybrid-based analysis is used to extract permissions and system calls features related to Android call logs exploitation. Tokenization approach is used to build a new classification model which is expected to produce unique behaviour patterns based on system call sequence. Combination of hybrid-based analysis and implementation of tokenization approach for Android mobile malware classification is proposed to increase the performance of the malware classification and detection hence producing a higher result in accuracy.

This paper is organized as follows. Section 2 presents the previous works related to Android mobile malware classification and detection techniques. Section 3 presents the research methodology and the proposed mechanism, including the processing of permissions and system call sequences extraction. Section 4 presents the overall experiment results and finally, Section 5 presents the summary and potential future works for this research.

II. Related Work

Malware analysis is a very time-consuming activity and one should have an in-depth knowledge and intelligence to handle it. A good structure of malware analysis approach is a great weapon to fight against the dark side of the information society [14]. Malware analysis usually involves static and dynamic-based analysis or combination of both, known as hybrid analysis [4].

Static-based analysis performs observation based on source codes or binaries without actually running the program. The results usually show the suspicious patterns and behaviours that lies in the program [8]. For Android malware detection, the researcher usually extracts features such as Requested Permission, API calls, Operation Code and system call [15]. These features are used to detect malicious payload and profile malware threats [15]. MAMA (Manifest Analysis for Malware Detection in Android) was a technique proposed by the authors to extract several features by analysing the Manifest file of the Android applications. They have reported to successfully achieve a high true positive rate which is 94.83% [16]. Other than that, Droid Mat was developed to detect Android malware with different intentions. They analyse 238 Android malware and focus on extracting API call and manifest file. Other researcher uses K-means and K-Nearest Neighbour as the classifier and the result shows 97.87% of classification accuracy [17]. An experiment by PUMA defines user permissions as their feature and managed to produce 86% accuracy rate [18]. Meanwhile, an experiment by Drebin uses several features such as hardware components, requested permissions, application components, filtered intents, API calls and network address. They managed to produce 94% accuracy based on different malware families [19].

Static-based analysis, however, is not effective against malware that employ cover-up techniques, such as code polymorphism and obfuscation. Moreover, this approach is slightly unstructured and rely heavily on experience and personal skills [8]. Therefore, dynamic-based analysis also known as behavioural-based analysis is the alternative approach to counter the weaknesses of the static-based analysis. Dynamic-based analysis observes the suspicious behaviour in a running application. A CopperDroid uses system call and binder information to see the bad behavior on suspected application [10]. An experiment known as CrowDroid has created four artificial malwares and obtained 100% classification accuracy based on its system call features. However this research focuses only on author-created malware and the result shows high false positive for real world malware [9]. Other than that, MALINE tools were used to detect malicious system call patterns on 4289 android samples. This experiment resulted 93% rate of accuracy. The result for their experiment were evaluated using histogram and Markov chain approach [20].

To enhance the accuracy of the detection, researchers nowadays use the hybrid-based features analysis approach. AASandbox performed a static and dynamic analysis to extract and analyse Android features such as permission and java code from the APK file [21]. Droid-Sec implemented hybrid-based analysis and the result shows 96.5% true positive based on 250 benign and 200 malicious samples [22]. A ProfileDroid has successfully discovered new unknown behaviours of mobile malware characteristics based on hybrid-base analysis [23]. Although these researches produced high positive rates of classification and detection, but most of the behavioural-based patterns produced by these researchers are inconsistent in terms of its string size and data flexibility. Besides, they also suffer from

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

lack of ability in handling huge amounts of features collected from the applications. These weaknesses can lead to low classification and detection accuracy rate. Therefore, a suitable approach is needed to increase the performance of the mobile malware classification and detection rate.

III. Research Methodology

This experiment proposes an Android mobile malware classification based on system call that is expected to exploit call logs. The tokenization approach is used to produce a unique system call sequence patterns. This research implements a hybrid-based analysis approach where permission feature is extracted during static analysis and system call feature is extracted during dynamic analysis. An initial experiment was conducted using 5560 Android malware samples from Drebin dataset [19]. A controlled laboratory environment is developed as illustrated in Figure 2. In this experiment, an emulator from Genymotion [24] was used with Android version of 4.1.1 and API level 16. This emulator runs in Windows 8 with 8GB of RAM.

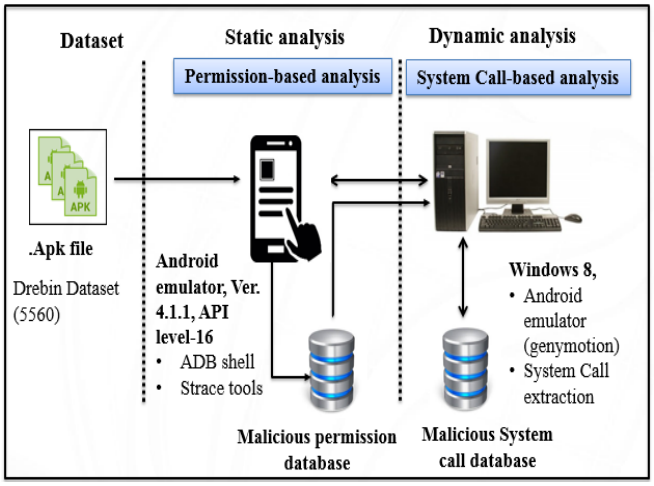

Fig. 2. Lab Architecture

A. Permission-Based Analysis Phase

The permission-based analysis is a static analysis approach. In this phase, all permissions related to call logs exploitation were extracted as shown in Figure 3.

Fig. 3. Permission-Based Analysis

First, 5560 malicious samples from Drebin [19] are used for static-based analysis. In this stage, "Virustotal" is an online tool used to retrieve requested built-in Android permissions from each sample. Application with at least one specific permission related to user call logs exploitation will then be used for further analysis. Table 1 shows the list of permissions expected to trigger user call logs exploitation. These permissions are chosen based on its function and ability to perform suspicious activities in user call logs. After going through permission-based analysis, malicious applications that are suspected to exploit call log will then go for the dynamic-based analysis.

TABLE I
Permissions that expected to Exploit Call Logs

Permissions	Function	Category
CALL_PHONE	Malicious apps may place unnecessary and illegal calls	Service that cost user's money
CALL_PRIVILEGED	Malicious apps may place unnecessary and illegal calls to emergency services	Service that cost user's money
PROCESS_OUTGOING_CALLS	Malicious apps can automatically make a phone call	Service that cost user's money
READ_CALL_LOG	Malicious apps able to read, save and share call log data without user notice	Service that gather user's personal information
READ_CONTACTS	Malicious apps able to read user's contact	Service that gather user's personal information
READ_PHONE_STATE	Malicious apps able to access features such as phone number and serial number of the phone	Service that gather user's personal information

B. System Call-Based Analysis Phase

The system call–based analysis phase consists of two main processes which are system call recorder and system call analyzer as shown in figure 4. In the first process, system call sequences of all suspected applications are extracted. This process is conducted in Genymotion VM environment [24]. The system call can only be triggered through user interaction with the running application. Therefore, monkey tool is used to generate pseudo-random gestures such as keystrokes, touches, and gestures on a devices or on an emulator [20]. With this tool, researcher can obtain consistent result as it allows user to manipulate the command based on requirement. Next, each application will go through 1000 random events or gestures generated at a time. The process of the system call recorder involved the following steps:

- Suspected application is installed in the emulator
- ADB shell monkey is used to trigger system call event
- Strace tools is used to record the system call
- The extracted system call is stored as Strace log for further process

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Finally, all recorded system calls for each suspected malicious application are stored as Strace log file for further analysis.

The system call analyzer consists of two main sub-processes. Firstly, the Strace log files are transferred to system calls sequenced patterns database. Each of the recorded system call is noted as 1 to indicate the presence of the system call while 0 indicates the absence of the system call. This step will produce string of binary number to represent the initial system call patterns for each application. Next, each of the binary patterns will be compared individually to avoid redundancy. Only unique patterns of system call sequences expected to exploit call logs are produced.

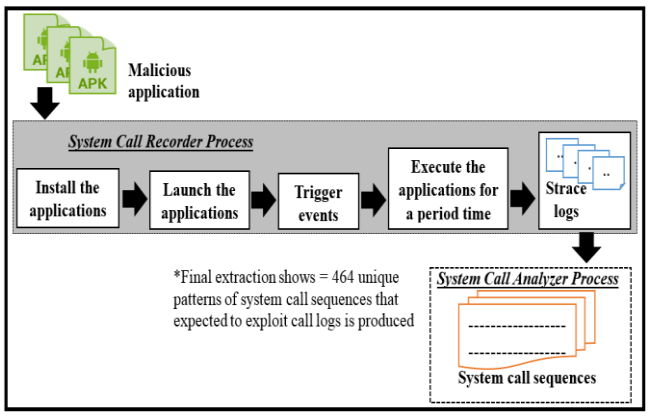

Fig. 4. System Call-Based Analysis

C. Tokenization of System Call Sequence Phase

In this phase, the extracted malicious system call sequence is converted into new unique patterns using tokenization approach. Unlike previous research, the covering algorithm successfully produced 60 patterns from malicious system call sequence but shows inconsistencies for the pattern's string size [1]. The system calls trigger was different for each application hence various system call string length were produced. Therefore, tokenization approach is implemented in this research with the aim to produce a consistent pattern's string size with high data processing flexibility. Figure 5 shows the working flow of the system call sequence classification that uses tokenization approach.

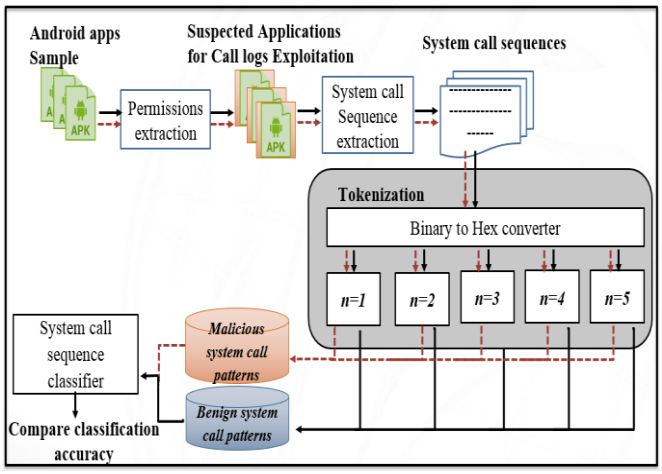

Fig. 5. System Call Sequence Classification using Tokenization approach

Tokenization approach consists of three main processes. First, the raw system call sequence extracted during the dynamic analysis are converted into binary value. Next, the binary patterns are then converted into hexadecimal value. This process will reduce the string size and produce consistent patterns string. Finally, tokenization approach is implemented to the new hex-value patterns to classify or break them up into a smaller pieces of input string [25]. Figure 6 shows the processes of the tokenization approach implemented in this experiment. The hex-value patterns are divided into a five *n* different dataset includes *n=1, n=2, n=3, n=4* and *n=5* where *n* indicates the number of tokenization group.

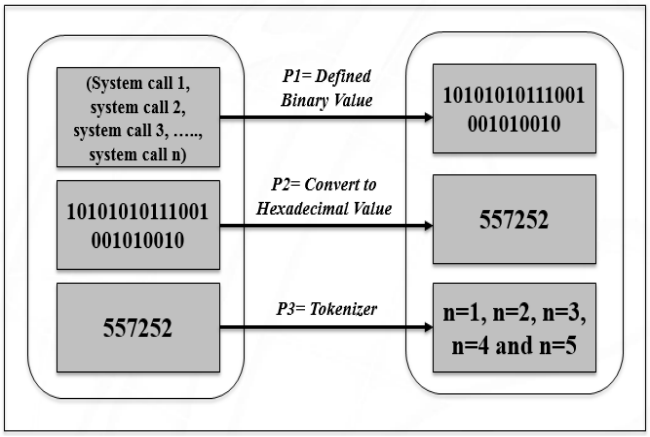

Fig. 6. Tokenization Process for System Call Sequence Classification

D. Behavior-Based Classification Evaluation Phase

This phase will evaluate the new malicious system call patterns expected to exploit call logs. The unique malicious system call patterns are further classified as malicious type or benign type malware to find the optimum *n*-value, classifier and the accuracy of the classification performance. Four classifiers which are SVM, Random Forest, Naïve Bayes, and J48 were run using WEKA 3.8.10 [26]. These classifiers are widely used by previous researchers [11] [19] [27] as they are able to deal with large instances and features such as in text classification, patterns analysis and bioinformatics [28].

The new system call patterns are evaluated based on its classification accuracy. This classification accuracy is determined based on the number patterns that are correctly classified as malicious patterns. The best *n* value and classifier are chosen based on the number of system call patterns that generates the highest classification accuracy.

IV. Result and Discussion

In this section, results related to samples extraction analysis and system call classification are presented and discussed.

A. New Unique Patterns

For this experiment, static and dynamic-based analysis are performed on 5560 samples of malicious application collected from Drebin [19]. Specific features were extracted in different phases. The first phase is the permission-based analysis where any application with the permissions features related to call logs exploitation as shown in Table 1 are categorized as malicious application. Next, a deep analysis

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

or dynamic analysis is carried out on the suspected malicious applications. In this phase, system calls from running applications are extracted. This phase will identify the behavior of each application based on user interaction. The extracted raw system calls will then be transferred to the system call sequence database. Next, each system call sequence log is compared to one another to eliminate redundancy, thus only unique system call patterns remained. From 5560 malicious application samples used in this experiment, 464 patterns of malicious system call sequence expected to exploit call logs were generated.

A tokenization approach is implemented in this experiment to reduce the pattern's string size and produce a consistent system call string length for each pattern. Furthermore, this approach increases the data flexibility by implementing a unique hex-value for each pattern hence optimizing the performance of mobile malware classification and detection. Figure 7 shows the examples of system call patterns in binary value which have been converted to hex-values.

Samples	Binary-patterns	Hexadecimal-patterns (N=1)
1	11101111100110000000010000000011111001100000100010100000000000000	03BE600803E608A000
2	10101000011100000001000000000001111101001000000000000011100001	02A1C04003E90000E1
3	11011011100111000110000100000000000000011000000001010000000000000	036E7184000600A000
4	10000111000010000000000000000000000000101100100100010100000000000000	021C2000001648A000
5	11111111100110000000010000100001111100110000010001010000000000000	03FE600843E608A000
6	11101111100110000000010000000000000000000000000000000000000000000	03BE60080000000000
7	11111111110111011100000000000000000001011001001000101000000000000	03FF7780001648A000

Fig. 7. Examples of 464 of malicious system call patterns conversion of binary value to hexadecimal value

B. Behavior-Based Classification Accuracy

In this experiment, four popular classifiers which are SVM, Random Forest, Naïve Bayes, and J48 were used to classify n=5 hex-value to different dataset produced during tokenization phase including the patterns in binary value. The classifier is validated using cross-validation. The default value is set as 10 where the cross validation is divided into 10 subsets and the holdout method is repeated for 10 times, where 9 subsets are used for training and the last piece is used for testing. This can estimate how well the learned model generalizes. 464 malicious patterns and the classification performance are shown in Figure 8.

This experiment aims to classify the system call patterns into malicious or benign application. Based on Figure 8, hex-value of the system call patterns using Naïve Bayes classifier where n=2 resulted the highest classification accuracy which is 99.86%. Interestingly, when patterns are classified higher than n=2, the accuracy shows all classifiers are decreased. This is due to the increasing number of groups for each pattern produced in tokenization phase generates a sparse vector element [28]. This element mostly held zero values, resulting in lower classification accuracy for each pattern. Other than that, the classification accuracy drops drastically once the patterns are converted from binary patterns to n=1 hex value patterns. The conversion of binary value to hexadecimal value had shorten the pattern's string length. To make the patterns size consistent for each application, java programming was set up with default one-dimensional array value based on number of system calls features extracted. The dataset n=1 hex value patterns was generated based on its default one-dimensional array value. Due to this, a high spare vector element had been generated in the dataset thus produce low classification accuracy for dataset n=1 hex value patterns [28].

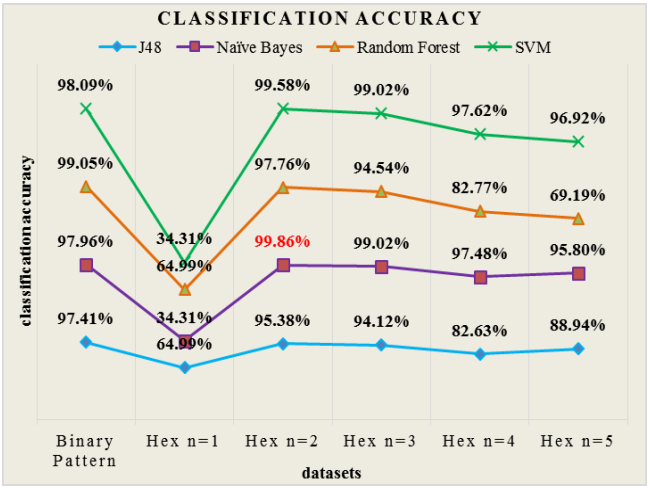

Fig. 8. Classification Accuracy

In terms of the classifier used, Naïve Bayes had successfully produced the highest classification accuracy compared to another classifier. This is due to the nature of Naïve Bayes which is able to handle huge amount of dataset and it is not sensitive to irrelevant features [29]. In this experiment, the binary patterns can produce high classification accuracy, but by implementing tokenization approach, the result is seemed to improve and increased up to 99.86% of accuracy rate with n=2 hex value is chosen as the best dataset.

C. Additional Discussion

During the system call-based analysis phase, the dynamic analysis was conducted in control lab environment. To sustain the consistencies of the result from each suspected sample, the emulator is set up as fix variable where it uses 4.1.1 Android version and 16 API level. However, from the experiment, not all samples can be executed. Some of them can be installed and run, while some of them can only be installed but unable to be executed (run in the background). On the other hand, some of them cannot be installed or executed in the emulator. The number of samples that having these conditions can be refer in Table 2.

This condition might occurs due to the period of samples collection from Drebin which is from August 2010 to October 2012 [19]. At that point, most of the Android devices are using lower than 4.1.1 Android version and API level 16. Therefore, some of the samples collected might only be compatible with the older and lowest version of Android and API level, thus making it unsuitable with this experiment environment. Therefore, only compatible samples were used for deep analysis.

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE II
List of Samples Condition

Condition	No. of Samples	Explanation
Installed and Executed	5122	Compatible with proposed experiment environment
Installed and run in background	357	Lower API level or Android Version
Unable to be installed	81	Lower API level or Android Version

V. Conclusion

This paper presents an Android mobile malware classification based on system call sequence patterns expected to exploit user call logs using tokenization approach. This experiment is successfully produce a new system call sequence unique patterns that shows specific behavior of the malicious application that exploit call log. The utilization of tokenization approach is seemed as a new approach to increase the mobile malware classification and detection performance. Besides, this approach has successfully produced a unique and consistent string length for each pattern. The size of patterns stored for data processing is also reduced, thus higher efficient performance of the classification and detection can be achieved. The experiment resulted 2-n hex-value patterns using Naïve Bayes managed to produce the highest accuracy with 99.86%. For future work, the proposed approach could be implemented in developing a powerful tool for Android malware detection. This approach can also be used as a basis research for other researchers to implement it with different Android application features.

Acknowledgment

The authors would like to express their gratitude to Ministry of Higher Education Malaysia and Universiti Sains Islam Malaysia (USIM) for the support and facilities provided.

References

[1] M. M. Saudi, F. Ridzuan, N. Basir, N. F. Nabila, S. A. Pitchay, and I. N. Ahmad, "Android Mobile Malware Surveillance Exploitation Via Call Logs : Proof of Concept," *UKSIM '15 Proc. 2015 17th UKSIM-AMSS Int. Conf. Model. Simul.*, pp. 176–181, 2015.

[2] e-marketer, "Slowing Growth Ahead for Worldwide Internet Audience," 2016.

[3] M. Dimjasevic, S. Atzeni, I. Ugrina, and Z. Rakamaric, "Android Malware Detection Based on System Calls," *Uucs*, 2015.

[4] X. Wang, Y. Yang, and Y. Zeng, "Accurate mobile malware detection and classification in the cloud," *Springerplus*, vol. 4, no. 1, p. 583, 2015.

[5] McAfee Labs, "McAfee Labs Threats Predictions Report," 2016.

[6] J. Wattles and J. D. Cnnmoney, "Ransomware attack : Who â€™s been hit," *CNNtech*, 2017. [Online]. Available: http://money.cnn.com/2017/05/15/technology/ransomware-whos-been-hit/index.html.

[7] T. S. Dutta, "Warning ! Millions Of Android Smartphones Hit By This Malware Warning ! Millions Of Android Smartphones Hit By ' Judy ' Malware," *techviral*, 2017. [Online]. Available: https://techviral.net/android-smartphones-hit-malware/.

[8] Y. Lin, C. Huang, Y. Chang, and Y. Lai, "Three-Phase Detection and Classification for Android Malware Based on Common Behaviors," vol. 12, no. 3, pp. 157–165, 2016.

[9] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-Based Malware Detection System for Android," *Proc. 1st ACM Work. Secur. Priv. smartphones Mob. devices - SPSM '11*, p. 15, 2011.

[10] A. Reina, a Fattori, and L. Cavallaro, "A system call-centric analysis and stimulation technique to automatically reconstruct android malware behaviors," *ACM Eur. Work. Syst. Secur. (EuroSec).*, pp. 1–6, 2013.

[11] W. Yu, H. Zhang, L. Ge, and R. Hardy, "On behavior-based detection of malware on Android platform," *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pp. 814–819, 2013.

[12] G. Suarez-tangil, J. E. Tapiador, P. Peris-lopez, and J. Blasco, "Expert Systems with Applications D ENDROID : A text mining approach to analyzing and classifying code structures in Android malware families," *Expert Syst. Appl.*, 2013.

[13] I. Santos, X. Ugarte-pedrero, F. Brezo, and P. G. Bringas, "NOA : AN INFORMATION RETRIEVAL BASED MALWARE DETECTION SYSTEM Jos ´ e Mar ´ ıa G o," vol. 32, pp. 1001–1030, 2013.

[14] H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, "Automatic Intelligent Analysis of Malware Behaviour," vol. 8, no. 4, pp. 1225–1229, 2015.

[15] S. Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," *2013 IEEE 27th Int. Conf. Adv. Inf. Netw. Appl.*, pp. 121–128, 2013.

[16] B. Sanz, I. Santos, X. Ugarte-pedrero, C. Laorden, J. Nieves, and P. Garc, "International Joint Conference SOCO'13-CISIS'13-ICEUTE'13," vol. 239, pp. 469–478, 2014.

[17] Y. Wang, J. Zheng, C. Sun, and S. Mukkamala, "Quantitative security risk assessment of Android permissions and applications," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7964 LNCS, no. September 2012, pp. 226–241, 2013.

[18] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Álvarez, "PUMA: Permission usage to detect malware in android," in *Advances in Intelligent Systems and Computing*, 2013, vol. 189 AISC, pp. 289–298.

[19] 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.*, pp. 23–26, 2014.

[20] M. Dimjaˇ, S. Atzeni, I. Ugrina, Z. Rakamari, and M. Dimjaˇ, "Android Malware Detection Based on System Calls Android Malware Detection Based on System Calls," 2015.

[21] T. Bläsing, L. Batyuk, A. D. Schmidt, S. A. Camtepe, and S. Albayrak, "An android application sandbox system for suspicious software detection," *Proc. 5th IEEE Int. Conf. Malicious Unwanted Software, Malware 2010*, pp. 55–62, 2010.

[22] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-Sec," in *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, 2014, pp. 371–372.

[23] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "ProfileDroid: Multi-layer Profiling of Android Applications," *Proc. 18th Annu. Int. Conf. Mob. Comput. Netw.*, pp. 137–148, 2012.

[24] "Genymotion," 2014. [Online]. Available: https://www.genymotion.com/.

[25] C. D. Manning, P. Raghavan, and H. Schütze, *An introduction to information retrieval*, vol. 21. 2009.

[26] Machine Learning Group at University of Waikato, "Weka 3: Data Mining Software in Java," 2014. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/indehtml.

[27] B. Wolfe, K. O. Elish, and D. Yao, "Comprehensive Behavior Profiling for Proactive Android Malware Detection," pp. 328–344, 2014.

[28] M. Z. Mas'ud, S. Sahib, M. F. Abdollah, S. R. Selamat, and R. Yusof, "An evaluation of n-gram system call sequence in mobile malware detection," *ARPN J. Eng. Appl. Sci.*, vol. 11, no. 5, pp. 3122–3126, 2016.

[29] E. Keogh, "Naïve Bayes Classifier," 2006.