International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 5
ISSN: 2088-5334

# CallDetect: Detection of Call Log Exploitation Inspired by Apoptosis

Madihah Mohd Saudi[a,1], Amirul Adli Che Ismail[b,1], Azuan Ahmad[a,2], Muhammad 'Afif Husainiamer[b,2]

[a] *Cybersecurity and System Research Unit, Islamic Science Institute, Universiti Sains Islam Malaysia (USIM), Nilai, 71800, Malaysia*
*E-mail: [1]madihah@usim.edu.my, [2]azuan@usim.edu.my*

[b] *Faculty of Science & Technology (FST), Universiti Sains Islam Malaysia (USIM), Nilai, 71800, Malaysia*
*E-mail: [1]maddymms@gmail.com, [2]afif@raudah.usim.edu.my*

*Abstract*— **Currently, we saw the increment trend of mobile application(app) exploitation that leads to loss of confidential information and money. Many malware camouflages itself as a genuine mobile app or exploits vulnerabilities inside mobile apps. Hence, this paper presents a mobile app called CallDetect that detects Android Application Interface (API) exploitation for call logs inspired by apoptosis. Apoptosis is known as cell-programmed death, and it is part of the human immunology system. Once it suspects any danger that might cause any harm to the human body, it will kill the suspected danger and itself. In the case of CallDetect, it will scan and uninstall the potentially malicious mobile application on a mobile phone. CallDetect consists of 13 new classifications of API call log, which are used as the database for CallDetect. These classifications were built by using static analysis and open source tools in a controlled lab environment. There were 5560 training datasets from Drebin and 550 anonymous testing dataset from Google Playstore. Our finding showed that 39 mobile apps, or 7%, were identified with possible call log exploitation. This paper can be used as a reference for call log API exploitation and can be further enhanced by integrating it with permission and system call exploitation.**

*Keywords*— **call log exploitation; API; mobile malware; static analysis; apoptosis.**

## I. Introduction

There are different categories of malware, such as worm, virus, Trojan horse, and mobile botnet. This malware can infect and exploit PC, notebook, tablet, or mobile phone without the owner's consent. In a mobile phone, surveillance features such as call log, Short Message System (SMS), camera, geolocation (GPS), and audio could be exploited by malware. The Android platform is the one most commonly targeted by malware due to its open-source distribution and being used by many users across the world [1]. Mouawad and CallJam virus are examples of worms that exploited Android smartphone by charging the victim with some amount of money through different applications [2], [3]. Examples of the exploitations are extra charges on phone bills, invasive advertising, and extra charges messages and phone calls. According to Gonzales, since 2013, this exploitation has skyrocketed and caused the loss of money to the telcos [4]. Therefore, we need mitigation for such exploitation. In the Android architecture, the Android Programming Interface (API), permission, and system calls are the aspects mostly used for exploitation [5].

In this paper, we present a CallDetect app that can detect call log exploitation. It is inspired by apoptosis. Apoptosis or also known as cell-programmed death and is part of the human immunology system. It has been integrated with the CallDetect app. Once the CallDetect app detects any potential harm related to the call log, it will uninstall the application to avoid any harm to the user's mobile phone. Compared to other human immunology system mechanisms, apoptosis is selected due to its role in terms of detection and response. Once the system detects any harm, if it cannot destroy the harmful element by itself, it will kill the element together with itself. This process is called as cell-programmed death. Few existing works have applied the apoptosis concept [6]-[8]. These show the practicality of the apoptosis concept. Inside the CallDetect, there are thirteen (13) new call log exploitation classifications have been developed. The details of the results are explained in Section III of this paper. This paper is organized as follows: Section II discusses the material and method used; section III presents results and discussion, and selection IV concludes this paper and discusses future work.

## II. Materials and Method

Lately, many scams have been conducted via phone calls, and various software packages have been developed to detect and block the scam phone numbers. Yet, currently, we are still lack of a solution to detect malicious mobile apps, especially those related to call log Application Programming

Interface (API) exploitations [9]. For example, MalDozer has been developed to detect malware in different IoT devices with API as the input, but not focusing on call logs [10]. Furthermore, other existing works by [11]-[21] showed that API could be exploited by malware. Nevertheless, performance improvement is needed for the experiment [19], [20]. Existing works [16], [22], [23], combined API and permission as part of their techniques to detect malware, yet improvement is needed in terms of the feature selection and classifier concerning optimizing results. The summarization of these works can be seen in Table 1. Based on the existing works presented in this paper, each of these works has its technique to overcome malware attacks. However, none of the studies focus on mobile phone surveillance features that can easily be further exploited by malware. As for the current scenario, this paper focuses on the call log as it is one of the leading surveillance features in a mobile phone, and due to its significant impact concerning losing money and invading privacy.

TABLE I
SUMMARISATION OF EXISTING RELATED WORKS

| **Author** | **Feature** | **Suggestion for Performance Improvement** |
|---|---|---|
| [16] | API and permission | In term of feature selection and permission list |
| [22] | API and permission | In term of feature selection and classifier. |
| [24] | API and permission | In term of malware classification based on binary format. |

Fig. 1 displays the apoptosis cycle in the human body [24]. Table II shows how the apoptosis is being mapped to a mobile phone. For malware detection, there are other bio-inspired computing algorithms such as Particle Swarm Optimisation (PSO), Negative Selection Algorithm (NSA), Artificial Bee Colony, and Bat algorithm that have been applied in existing works. Nonetheless, based on the comparison between these algorithms and apoptosis (refer Table III), apoptosis is selected due to its performance capability in terms of faster detection and response, to its being easier for real-time implementation, and its higher detection rate.

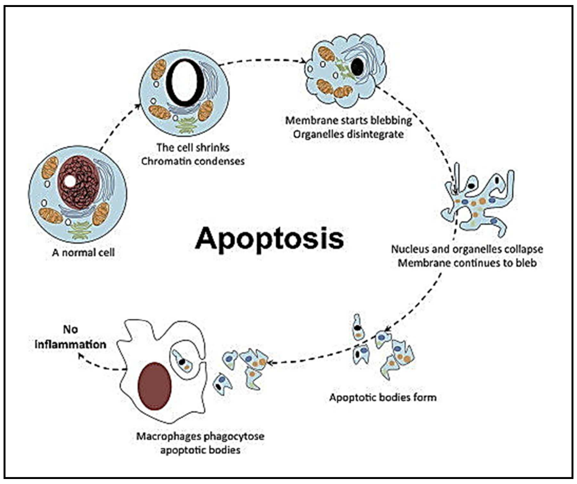


Fig. 1 Apoptosis Cycle

TABLE II
MAPPING APOPTOSIS TO MOBILE PHONE

| **Apoptosis** | **Mapping Apoptosis to Mobile Phone Surveillance Features** |
|---|---|
| The system must be able to readjust itself automatically, either to support a change in circumstances or to assist in meeting other system objectives. | The mobile phone will terminate any detected malicious process before it can cause any harm. It is based on a mobile phone surveillance feature: the call log. Mobile phones will uninstall the potentially harmful process or application. |

TABLE III
COMPARISON WITH OTHER BIO-INSPIRED ALGORITHMS

| **Name of Algorithm** | **Weakness** |
|---|---|
| PSO | Performance issue in term of optimisation. |
| NSO | Continuous learning ability issue and performance issue in terms of practicality concerning implementation in the real world. |
| Artificial bee colony | Performance issues in terms of accuracy. |
| Bat algorithm | Performance issues in terms of accuracy and optimisation. |
| Genetic algorithm | Performance issues in terms of optimisation |
| Apoptosis | Unique technique with better accuracy, faster and easier detection and response mechanisms. |

Before the development of the CallDetect application, 5,560 of the Drebin dataset were reverse-engineered by using static analysis [25]. The Drebin dataset is one of the biggest datasets for mobile malware and has been used by a small number of other studies [11], [26]-[28]. Five hundred fifty mobile apps have been selected randomly from the Google Play Store to evaluate the proposed classification and the app proposed in this paper. The experiment was conducted in a controlled lab environment and as per displays in Fig. 2 and Table 2.

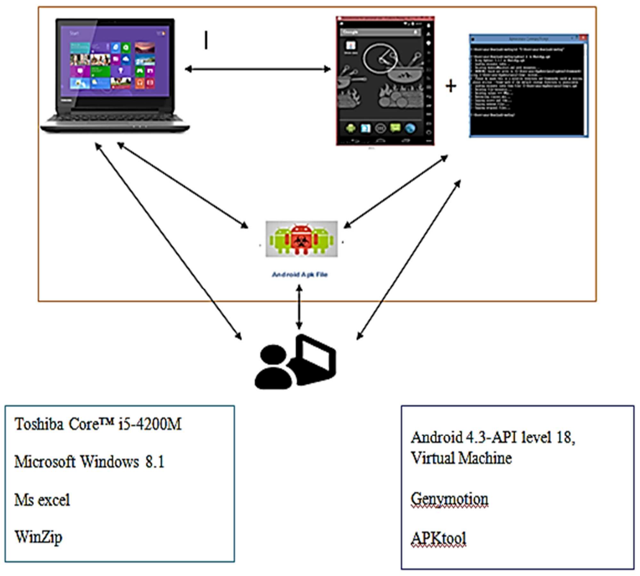


Fig. 2 Lab experiment architecture

TABLE IV
SOFTWARE USED FOR THE EXPERIMENT

| Software | Function |
|---|---|
| JAVA/Android Studio | To develop mobile application |
| Genymotion | It acts as Android emulator. |
| Show Java Application /APKtool | For APK resource file decompiling and permission extraction. |
| Java Decompiler | For API extraction. |

In contrast with dynamic analysis, static analysis is an analysis where the codes will not be executed. Existing work [29] summarises the value of static analysis, dynamic analysis, and hybrid analysis for malware analysis and detection techniques. Each of the techniques has it owns strength. In this paper, static analysis is used to extract the manifest file and APIs, due to its suitability for depth analysis. Based on the APIs' extraction from the training dataset, those that are related to possible call log exploitation were retrieved. These were then compared with the testing dataset from the Google Play Store.

Fig. 3 represents the static analysis conducted, and Fig. 4 depicts the overall research processes involved in this experiment.

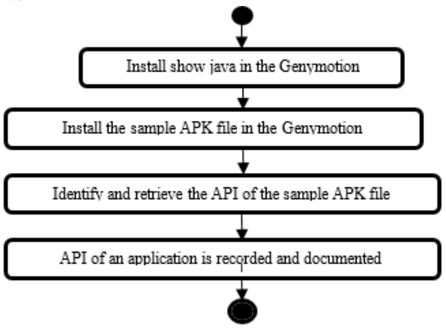
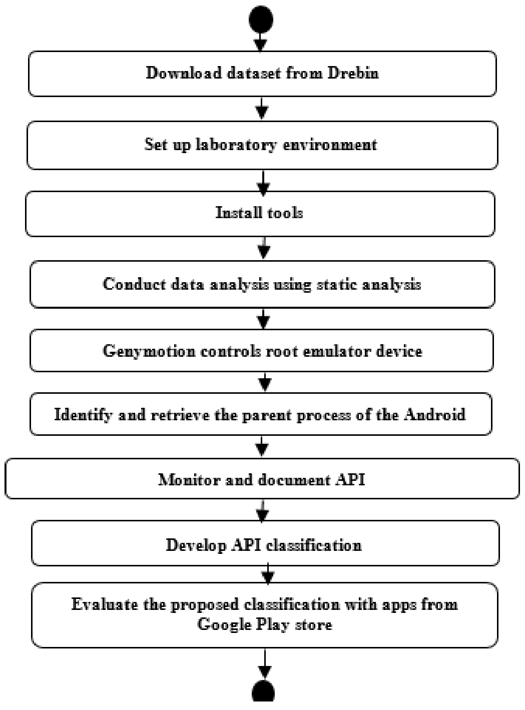

Install show java in the Genymotion
Install the sample APK file in the Genymotion
Identify and retrieve the API of the sample APK file
API of an application is recorded and documented

Fig. 3 Static analysis


Download dataset from Drebin
Set up laboratory environment
Install tools
Conduct data analysis using static analysis
Genymotion controls root emulator device
Identify and retrieve the parent process of the Android
Monitor and document API
Develop API classification
Evaluate the proposed classification with apps from Google Play store

.

Fig. 4 Overall research process

```
Given:
- Set characteristic's value: {API, not API}
- Set CharFlag = 0
- Set CharFlag = 0
- Dataset A
Output:
- API classification for Dataset A
Algorithms:
While (case ≤ 5560)
{
  -  get the worm attributes

     While (worm_attributes != null)
     {
          While (Dataset A != empty)
            {
 - Determine characteristic value for each type of worm
attribute  from the Dataset A
 - Dataset B = Dataset A
            (worm_attributes[case,type])
                        If (characteristic_value = API)
                          {
                           CharFlag = 1
                     break
                          }
                        If (characteristic_value = notAPI)
                           CharFlag = 1
          }
                     else
-  get the next worm attributes from Dataset A
            }
}
```

Fig. 5 Covering algorithm for the call log classification formation

Fig. 5 depicts the method by which the covering algorithm is used for the formation of API call log classification. The covering algorithm has the rule to be followed during each phase of the existing attributes. It consists of a positive instance and a total of the dataset. Positive instances are represented as PA, while the dataset total is D. The accuracy of the approach is based on the formulation of PA/D. Then it develops rules with a 100% accuracy rate. It is simplified as pseudocodes in Fig. 5, and Fig. 6. The concept of covering algorithm excludes many instances of other classes and include as many instances of the desired class as possible.

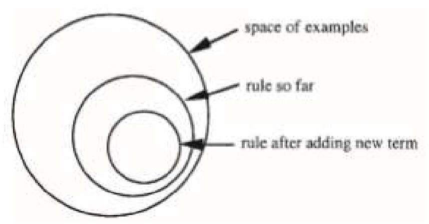


Fig. 6 Covering algorithm for instance space

## III. RESULT AND DISCUSSION

The following are the interfaces for the CallDetect app mobile application. Fig. 7 displays the main interface of the developed apps, while Fig. 8 shows the extracted manifest file from the selected app. Once the user selects a mobile app, the CallDetect app will run the scanning process. If it is matched with the database of the potential call log exploitation, it will ask the user if the user would like to uninstall the mobile app (refer Fig. 9). CallDetect is better compared to emulator-based work because it is real-time and based on the real scenario of a smartphone user.

Before the development of CallDetect, analysis and reverse engineered were carried out. The database inside the CallDetect app is based on the classification formed during the experimental work. The database is based on the highest number of potential features that could be exploited related to the call log, as displayed in Table V.

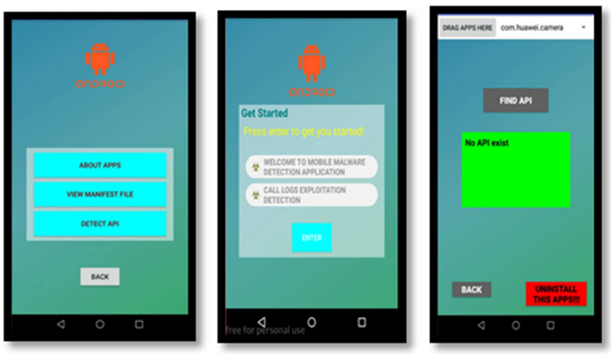
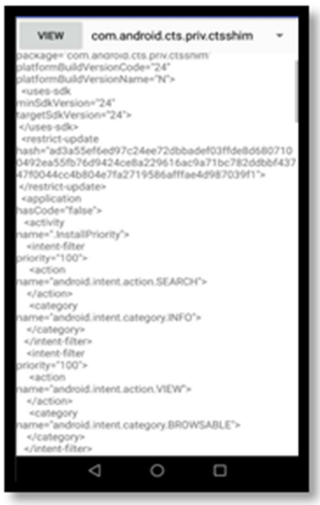


Fig. 7 CallDetect Main Interfaces



```
package="com.android.cts.priv.ctsshim"
platformBuildVersionCode="24"
platformBuildVersionName="N">
 <uses-sdk
minSdkVersion="24"
targetSdkVersion="24">
 </uses-sdk>
 <restrict-update
hash="ad3a55ef6ed97c24ee72dbbadef03ffde8d680710
0492ea55fb76d9424ce8a229616ac9a71bc782ddbbf437
47f0044cc4b804e7fa2719586afffae4d987039f1">
 </restrict-update>
 <application
hasCode="false">
  <activity
name=".InstallPriority">
   <intent-filter
priority="100">
    <action
name="android.intent.action.SEARCH">
   </action>
    <category
name="android.intent.category.INFO">
   </category>
  </intent-filter>
   <intent-filter
priority="100">
    <action
name="android.intent.action.VIEW">
   </action>
    <category
name="android.intent.category.BROWSABLE">
   </category>
  </intent-filter>
```

Fig. 8 CallDetect for manisfest file extraction

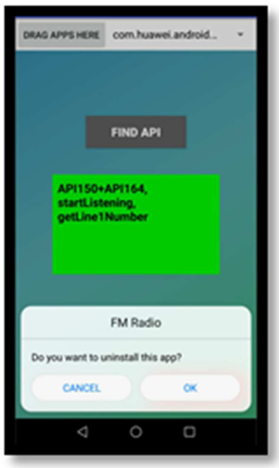

DRAG APPS HERE
com.huawei.android...
FIND API
API150+API164, startListening, getLine1Number
FM Radio
Do you want to uninstall this app?
CANCEL
OK

Fig. 9 CallDetect displays uninstall message for potential harm apps

Before these 13 classifications are developed, all the APIs related to the call log are extracted, and the most closely related could be exploited by malware. From the testing dataset, all APIs related to call logs have been extracted, and the ten (10) most related are listed in Table VI. Each is assigned with different values either as normal or dangerous. The normal is a default value, and it could be executed without asking the owner's consent. While dangerous has a higher risk related to privacy, and it could be executed with or without asking the owner's consent. Based on these normal and dangerous values, the classification is created, as displayed in Table V.

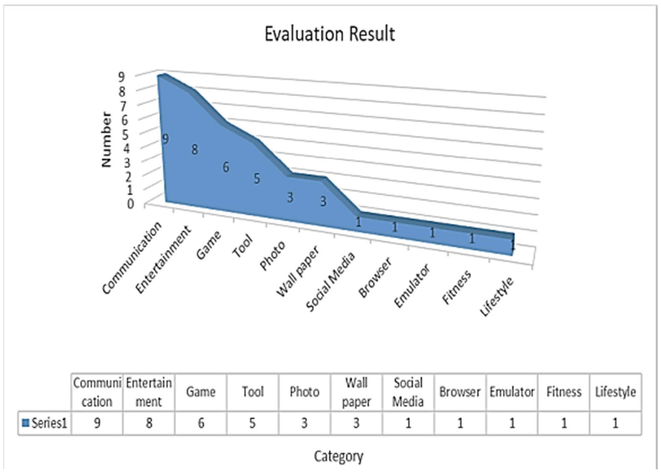
TABLE V
NEW CALL LOG EXPLOITATION CLASSIFICATION

| Classification | Content |
|---|---|
| 1 | A1+A2+A3+A4+A5+A6+A7+A8+A9+A10 |
| 2 | A4+A6+A9 |
| 3 | A6+A9 |
| 4 | A5+A6+A9 |
| 5 | A4+A5+A6+A7+A8 |
| 6 | A8+A8+A9 |
| 7 | A4+A6 |
| 8 | A4+A9 |
| 9 | A4+A5 |
| 10 | A4+A8 |
| 11 | A5+A9 |
| 12 | A2+ A4 |
| 13 | A2+A5 |

TABLE VI
TEN API MOST RELATED WITH CALL LOG EXPLOITATION

| Data Representation | API | Function | Value |
|---|---|---|---|
| A1 | addToMyContactsGroup | To add contact. | Normal |
| A2 | startListening | It starts to listen to audio speech. | Dangerous |
| A3 | isVoiceMailNumber | It checks the given number with the voicemail number inside the SIM card. | Normal |
| A4 | getLine1Number | It returns the phone number string. | Dangerous |
| A5 | getNeighboringCellInfo | It gets and informs the neighbouring device information. | Normal |
| A6 | getSimSerialNum-ber | It gets the Simcard serial number. | Dangerous |
| A7 | getVoiceMailAlphaTag | It retrieves the voice mail number based on the alphabetic identifier. | Dangerous |
| A8 | getVoiceMailNumber | It retrieves the voice mail number | Normal |
| A9 | listen | It is a listener for changes notification in the mobile phone. | Dangerous |
| A10 | getCallerInfo | To retrieve caller information. | Dangerous |

Then 550 dataset randomly picked from the Google Play Store are tested with the proposed classification and database inside the CallDetect app. Based on the testing, it was found that 39 out of 550 mobile apps matched with the call log exploitation in Table V. Fig. 10 and Table VII are the summarization of the test results. Based on the testing process, communication has the highest frequency with 23.1%, followed by entertainment with 20.5% and games with 15.4%. Mobile phone users most commonly use these three categories. Entertainment with its call log feature makes user life more comfortable but also exposes them to possible call log exploitation. As for games, this is something to be pondered, since most games do not need call features inside the game. It could lead to possible premium call rates if an attacker is abusing it. Other categories, in the form of social media, browser, emulator, fitness, and lifestyle, have the lowest possible call log exploitation.



|  | Communication | Entertainment | Game | Tool | Photo | Wall paper | Social Media | Browser | Emulator | Fitness | Lifestyle |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Series1 | 9 | 8 | 6 | 5 | 3 | 3 | 1 | 1 | 1 | 1 | 1 |

Fig. 10 CallDetect evaluation result

TABLE VII
EVALUATION RESULTS WITH GOOGLE PLAY STORE

| **Category** | **Number** | **Percentage** |
|---|---|---|
| Communication | 9 | 23.1 |
| Entertainment | 8 | 20.5 |
| Game | 6 | 15.4 |
| Tool | 5 | 12.8 |
| Photo | 3 | 7.7 |
| Wallpaper | 3 | 7.7 |
| Social Media | 1 | 2.6 |
| Browser | 1 | 2.6 |
| Emulator | 1 | 2.6 |
| Fitness | 1 | 2.6 |
| Lifestyle | 1 | 2.6 |

The significant formation of 13 API call log classifications is that these APIs are used as the database for CallDetect app. It is capable of detecting potential call log exploitation inside a mobile phone. Once it detects any potential harm, it will trigger a message to the user and asks the user to uninstall it to avoid further harm. Furthermore, with the existence of these 13 APIs call log classifications, developers will use it as a form of guidance and will implement secure coding in developing mobile apps. Furthermore, it could be used as guidance for mobile apps developer concerning how attackers could exploit a smartphone via the API of the call log. Nonetheless, users must be aware of the apps they installed, since these APIs might pose financial risks for smartphone users in terms of premium rate phone calls and scam phone calls. Users and developers must be aware that API concerning call logs could be exploited by malware.

## IV. CONCLUSION

In this paper, based on the CallDetect app that has been developed, it is proven that possible call log exploitation via API could be detected and responded. It is inspired by apoptosis and uses 13 new API call log classifications as its database. It is the right solution in detecting any new mobile apps with potential call log exploitation. Based on the evaluation conducted, 7% of the tested mobile apps matched with the possible call log exploitation where mobile apps from communication, entertainment, and game categories have the highest score, respectively. Mobile phone users commonly use these categories. Hence user awareness and solutions such as CallDetect app offer proper detection and preventive mechanisms. In the future, other surveillance features in the form of SMS, camera, audio and GPS, should be integrated with the CallDetect app for better performance, and more comprehensive detection and preventive mechanisms.

## ACKNOWLEDGMENT

The authors would like to express their gratitude Ministry of Higher Education (MOHE), Malaysia and Universiti Sains Islam Malaysia (USIM) for the support and facilities provided. This research paper project is under FRGS grant: [P5-2-50-50819-KPT-FRGS-FST].

## REFERENCES

[1] A. Verma, S. Arora, and P. Verma, “Android OS, its security and features,” Int. J. Recent Res. Asp., vol. 4, no. 3, pp. 241–251, 2017.
[2] M. Kumar, “Mouabad Android Malware is calling to Premium numbers; Generating revenue for its Master,” 2013. [Online]. Available: https://thehackernews.com/2013/12/mouabad-android-malware-calling-to.html. [Accessed: 15-Jun-2020].
[3] Matthew Broersma, “Google Play CallJam Malware Infects Half A Million Users,” 12-Sep-2016. [Online]. Available: https://www.silicon.co.uk/mobility/google-play-malware-premium-calls-197557. [Accessed: 15-Jun-2020].
[4] Katia Gonzales, “Telecom Fraud: $29 Billion And Counting - Why It Matters More than Ever in the Digital Era | Horizon House Publication Inc.,” 04-Apr-2018. [Online]. Available: https://www.telecomengine.com/article/telecom-fraud-29-billion-and-counting-why-it-matters-more-than-ever-in-the-digital-era/. [Accessed: 15-Jun-2020].
[5] H. Shewale, S. Patil, V. Deshmukh, and P. Singh, “Analysis of Android Vulnerabilities and Modern Exploitation Techniques,” ICTACT J. Commun. Technol., vol. 5, no. 1, pp. 863–867, 2014.
[6] D. Jones, “Implementing biologically-inspired Apoptotic behaviour in digital objects : An Aspect-Oriented Approach,” no. March, 2010.
[7] M. M. Saudi, M. Woodward, A. J. Cullen, and H. M. Noor, “An overview of apoptosis for computer security,” in Proceedings - International Symposium on Information Technology 2008, ITSim, 2008, vol. 3.
[8] R. Sterritt, “Apoptotic computing: Programmed death by default for computer-based systems,” Computer (Long. Beach. Calif)., vol. 44, no. 1, pp. 59–65, Jan. 2011.
[9] P. Ravi Kiran Varma, K. P. Raj, and K. V. Subba Raju, “Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms,” in Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017, 2017, pp. 294–299.
[10] E. M. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, “MalDozer: Automatic framework for android malware detection using deep learning,” in DFRWS 2018 EU - Proceedings of the 5th Annual DFRWS Europe, 2018, vol. 24, pp. S48–S59.
[11] K. A. Talha, D. I. Alper, and C. Aydin, “APK Auditor: Permission-based Android malware detection system,” Digit. Investig., vol. 13, pp. 1–14, Jun. 2015.

[12] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, “MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention,” IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 83–97, Jan. 2016.

[13] S. Y. Yerima, S. Sezer, and I. Muttik, “High accuracy android malware detection using ensemble learning,” IET Inf. Secur., vol. 9, no. 6, pp. 313–320, Nov. 2015.

[14] Z. Wang, J. Cai, S. Cheng, and W. Li, “DroidDeepLearner: Identifying Android malware using deep learning,” in 37th IEEE Sarnoff Symposium, Sarnoff 2016, 2017, pp. 160–165.

[15] Kamesh and N. S. Priya, "Security Enhancement of Authenticated RFID Generation," International Journal of Applied Engineering Research (IJAER), vol. 9, no. 22, pp. 5968-5974, 2014.

[16] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, “Significant Permission Identification for Machine-Learning-Based Android Malware Detection,” IEEE Trans. Ind. Informatics, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[17] D. Li, Z. Wang, L. Li, Z. Wang, Y. Wang, and Y. Xue, “FgDetector: Fine-Grained Android Malware Detection,” in Proceedings - 2017 IEEE 2nd International Conference on Data Science in Cyberspace, DSC 2017, 2017, pp. 311–318.

[18] M. Mohd Saudi and A. Husainiamer, “Mobile Malware Classification via System Calls and Permission for GPS Exploitation,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 6, pp. 277–283, 2017.

[19] P. Burnap, R. French, F. Turner, and K. Jones, “Malware classification using self organising feature maps and machine activity data,” Comput. Secur., vol. 73, pp. 399–410, Mar. 2018.

[20] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, “Detecting Android Malware Leveraging Text Semantics of Network Flows,” IEEE Trans. Inf. Forensics Secur., vol. 13, no. 5, pp. 1096–1109, May 2018.

[21] Z. Abdullah and M. M. Saudi, “RAPID-Risk assessment of android permission and application programming interface (API) call for android botnet,” Int. J. Eng. Technol., vol. 7, no. 4, pp. 49–54, 2018.

[22] S. Chen et al., “Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach,” Comput. Secur., vol. 73, pp. 326–344, Mar. 2018.

[23] S. Y. Yerima and S. Sezer, “DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection,” IEEE Trans. Cybern., vol. 49, no. 2, pp. 453–466, Jan. 2018.

[24] M. Abou-Ghali and J. Stiban, “Regulation of ceramide channel formation and disassembly: Insights on the initiation of apoptosis,” Saudi J. Biol. Sci., vol. 22, no. 6, pp. 760–772, Nov. 2015.

[25] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,” in Network and Distributed System Security Symposium(NDSS), 2014, pp. 1–15.

[26] M. Yusof, M. M. Saudi, and F. Ridzuan, “A new mobile botnet classification based on permission and API calls,” in Proceedings - 2017 7th International Conference on Emerging Security Technologies, EST 2017, 2017, pp. 122–127.

[27] Z. Li, L. Sun, Q. Yan, W. Srisa-An, and Z. Chen, “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, LNICST, 2017, vol. 198 LNICST, pp. 597–616.

[28] 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 Proceedings - 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2014, 2016, pp. 3–17.

[29] R. Sihwail, K. Omar, and K. A. Z. Ariffin, “A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis,” Int. J. Adv. Sci. Eng. Inf. Technol., vol. 8, no. 4–2, pp. 1662–1671, Sep. 2018.