SYSTEMATIC LITERATURE REVIEW ON APPLICATION PROGRAM INTERFACE - BASED ANDROID MALWARE DETECTION

EHODA E.,[1] ADEBAYO O.S.,[2] ISMAILA I.,[3] OJENIYI J.A.,[4] OLALERE M[5]

[1,2,3,4] Cyber Security Science Department,

Federal University of Technology, Niger State Nigeria.

[5]Islamic University, Uganda

Correspondence : eehoda@gmail.com

Abstract.

Over the years, various malware detection approaches have been proposed in a bid to address evolving malware threats landscape in android operating system. Systematic literature reviews to analyze these detection approaches have been carried out, but none have been tailored to identifying challenges with android malware detection based on the use of Android program interface (API) features, hence there is no aggregated information on what work has been done by researchers in this area. This research, therefore, presents a systematic literature review on API feature based android malware detection literatures between 2018 to 2022 collected systematically using PRISMA frameworks. This study seeks to identify the challenges faced in android malware detection over the years, methodologies used to address them and limitations of API based feature detection. These useful insights documented in this research will serve as valuable resources which researchers can leverage on to improve the detection of android malware.

Keywords: *Android Platform, Malware Detection, Application Program Interface, PRISMA Framework*

Introduction

The number of smartphone users in the world has grown from 4.435 billion to 6.648 billion from 2017 to 2022 and this number constitutes 90.72% of the world's population according to a report by Statista. The large percentage of mobile phone users motivates attackers to target mobile phone platforms, predominantly android operating systems with malwares as more persons are interconnected and exposed to the threats. With more people exposed to the threat and proliferation of various applications in the android platform, comes the increased burden to protect users' devices against malicious applications and attacks. Over the years, a number of malware detection approaches have been proposed in a bid to address evolving malware threats landscape in android operating system. Researchers have employed the use of static features such as permission and strings, however this approach contends with the challenge of code obfuscation and other malware evasion techniques. Systematic literature reviews to analyze these detection approaches have been carried out by some researchers, but none have been tailored to android malware detection based on the use of API feature, hence there is no aggregated information on what work has been done by researchers in this area. This research, in a bid to identify the recent challenges associated with the detection of malicious applications on the android devices, carried out a systematic review of the existing detection strategies using API features. API feature based android malware detection papers between 2018 to 2022 were collected systematically using PRISMA frameworks and challenges faced in android malware detection over the years, methodologies used to address them and limitations API based feature detection is subject to have been identified and documented by this research. Also type of analysis employed and datasets used by the researchers as well as performance reported by the papers were highlighted. The remaining parts of the research are organized as follows: the related work is discussed in the section 2

of this work while the section 3 is used to discuss the methodology adopted to carry out the review. Section 4 discusses the analysis and result and section 5 presents the conclusion.

Related Work

Several works have been done in the area of android malware analysis and detection but more of the works were based on static features. Application Programming Interface is one useful feature that can define the behavior of applications therefore considering the limitation of permission based static features; malware researchers have drifted to approaches that use the behavioral pattern of applications in their quest to improve android malware detection mechanism. There are systematic literature reviews that have been conducted by different researchers on android malware detection but we have not come across any that examined android malware detection based on the use of API features. There is therefore no aggregated information on what work has been done by researchers in the area of API feature based analysis and detection. (Ashawa and Morris, 2019) carried out a systematic review of the malware detection techniques used for android devices. The review highlighted strengths and limitations of various detection techniques but not much was said on techniques using API based features. Similarly, (Ehsan et al. 2022) conducted a systematic literature review on android platform, analyzing articles focused on permission analysis for malware detection. On the other hand, Ya et al (2020) examined static analysis techniques for malware detection. They categorized static analysis into methods that are opcode based, program graph based, symbolic execution based and android characteristics based. They observed that static analysis methods are effective but are however faced with some challenges that needed to be addressed to improve android malware detection. Other SLRs dealt with the subject of android malware detection from a general perspective, hence our motivation to carry out a systematic literature review narrowed down to techniques that employ use of API based features.

Research Methodology

This section shows the methodology used in carrying out the systematic literature review. The steps followed are presented using figure1.

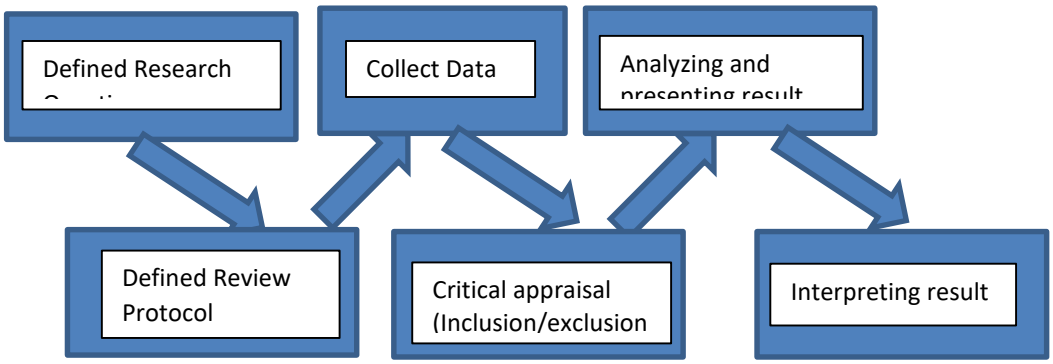

Figure 1. Systematic Literature Review Steps

Review Protocol

PRISMA framework was used to carry out this review. PRISMA stands for Preferred Reporting Items for Systematic Reviews and Meta-Analysis. It is an evidence-based minimum set of items for reporting systematic reviews and meta- analysis.

Data Collection

Articles searched and used for this review were systematically collected using the PRISMA framework.

a) Identification Stage

Material used for review were identified using search keywords and literature identified from IEEE, SCIENCEDIRECT and Google Scholar. The breakdown is as shown below;

A = Articles identified from IEEE - 98

B = Articles identified from Science Direct - 65

ADs = Articles identified with Google Scholar – 50

Table 1: Review Data Collection Parameters

Criteria	
Search Keyword	"Android Malware Detection" AND "API Call" was used to search for the review articles and databases searched are Science Direct and IEEE

Inclusion	A1 = Journals and conference papers A2 = Papers that discussed API as feature A3 = Papers between 2018 - 2022
Exclusion	B1 = Papers later than 2018 B2 = Paper not explicitly related to android malware detection using API features B3 = duplicate copies indexed in other databases

b) Screening Stage

The papers collected were subjected to a screening procedure applying inclusion criteria A1, A2, A3 and exclusion criteria, B3 where 27 duplicates were removed. Furthermore, applying exclusion criteria B2, title and abstracts were reviewed to reduce the articles to the subject area in focus. This reduced the papers collected for review to 54. The breakdown from each database is as shown in table 2. below

Table 2: Papers Collected from Database

DATABASE	Number of Papers
IEEE	25
SCIENCE DIRECT	18
GOOGLE SCHOLAR (Springer, ACM, Research gate)	11
TOTAL	54

c) Eligibility Stage

Quality of papers collected was assessed to ensure they are useful for the research. Duplicates were removed, abstract were thoroughly reviewed in line with the research focus and no paper was removed as a total of 54 papers were retained.

d) Inclusion Stage

Based on the inclusion and exclusion parameter earlier defined, all 54 papers screened were used for the review.

Critical Appraisal (Inclusion/exclusion)

The processes carried out between screening stage and eligibility stage defined in the PRISMA framework constitute the step of critical appraisal in the methodology. The papers returned after searching the database using the above keywords and inclusion/exclusion criteria were critically appraised. Papers within the scope of research bordering on API calls were kept while papers outside the scope which discussed other methods of detection other than use of API were screened. Furthermore, papers not related such as those that discussed cloud-based detection; windows OS or PE based and IoT based detection were screened leaving a total of 54 papers for synthesis in the review. Figure 2 shows the article search flow from the identification stage to articles included for the study

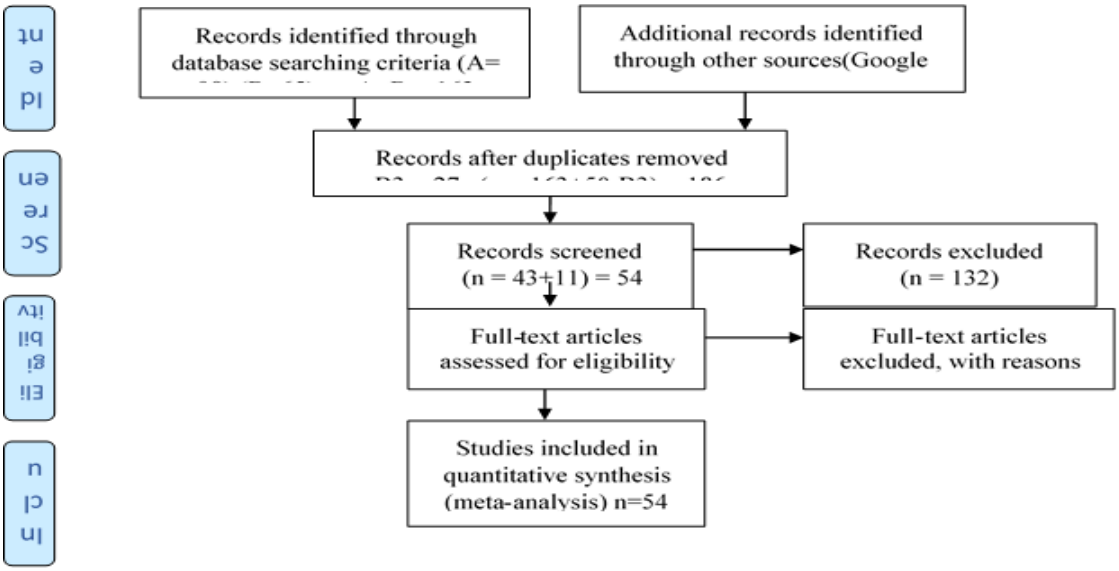

Figure 2: Prisma Flow Diagram

1. Analyses and Result Presentation

In this section, we provide answers to our research questions using papers studied. Figure 3 shows frequency distribution of papers from database sources, figure 4 shows frequency distribution of datasets used.

RQ1 - Are there challenges with Android Malware Detection approaches that prompted consideration for use of API features?

There have been a lot of efforts put into proffering solutions to malware attacks in android platforms by different researchers. However, these solutions have been fraught with myriad challenges which prompted researchers to consider the use of API call features with different models to address.

Wang et al (2020) observed that lack of descriptive distinctive feature of malware behavior and intent poses a challenge for android malware detection and proposed invocation of local sensitive API call using function call graph to address the challenge. Elsewhere, the work by Alzaylaee et al (2020) noted that employing static analysis using extracted features like API calls, commands and intent are prone to obfuscation where malicious code is concealed to prevent detection. Therefore, an approach to detection which considers extracting android permissions (static features) before execution of application and then extracting the API calls and Intents during execution (dynamic feature) was proposed using multilayer perceptron classifier, (MLP. Kumar and Ciza, 202) also alluded to the problem of code obfuscation with static analysis but employed a different approach to solving it. Obfuscation has also been mentioned in other reports (Lu, et al. 2019, Moutaz et al. 2020, Elayan and Mustafa 2021 and Michele et al, 2019). Other reports (Arindaam et al. 2020, Stuart et al. 2021) noted inability to detect zero-day malware as a challenge while (Roopak et al. 2020) cited the problem of multicollinearity and data overfitting in most classifiers used. The challenge of dynamic code loading was also stated by Elayan and Mustafa (2021) while Pang and Bian (2019) averred

that dynamic analysis is not efficient and malware detection can be limited due to execution time and code trigger condition. The research by Feng et al, (2020) also noted poor efficiency of dynamic analysis due to limitation in code coverage while (Kumar et al 2021) indicated that the problem of unbalanced dataset affected precision, recall, and F1 score values of the classifiers used and should be considered for investigation in the future.

RQ2 – Are there challenges with use of API in Android Malware Detection and can the challenges affect the detection?

Literature synthesized suggests that API features can be employed dynamically or statically with appropriate models to solve malware detection issues effectively. However, challenges also exist that can affect android malware detection where API features are used. Roopak et al, (2020) used conditional dependencies among relevant static and dynamic features (API calls, permissions and system calls) which are required for an app to work were used in a Tree Augmented Naive Bayes based hybrid malware detection mechanism and observed that few malicious software can evade the detection model by using adversarial techniques. Consequently, the researchers suggested future work for more powerful Bayesian models to be built for effectively identifying such adversarial malware applications by employing reinforcement learning techniques. Elsewhere, Michele et al, (2019) in their research relied on system API information to distinguish ransomware from other malicious and benign applications. Although inclusion of sample into the training set, meant the approach worked well against string obfuscation and heavy antistatic obfuscation done with class encryption, evasion is possible with this approach using semantically equivalent user implemented packages/classes/methods. Further to this, adversarial attack can also affect the outcome. According to Millar et al (2021), APIs usage as features requires a lot of feature-engineering and domain insight hence not effective in zero-day scenarios. Good accuracy and F1 score was achieved in work by Feng et al (2021) when they used graph neural networks to automatically capture critical information from call graphs rather than manual selection of

API calls, API call sequences and call traces. However, obfuscation techniques like packing, dynamic code loading and bytecode encryption could not be handled. Also, the approximate call graph used cannot capture reflection, implicit callback and implicit control flow and this could be exploited to evade detection.

In another research, Hadiprakoso et al (2020) declared that classical machine learning algorithms are dependent on feature engineering. This brings to the fore issues of expert domain knowledge needed for representation of the features as well as attackers' capability to evade detection once the features. Shen et al (2019) argued that evolution in modern malware has made reliance on simple information flow ineffective because modern malware performs complex computations before, during, and after collecting sensitive information and also, benign applications now use the same information that malicious applications gather. By performing N-gram analysis on sequences of API calls that occur along Complex-Flows' control flow paths to identify unique and common behavioral patterns present in Complex-Flows they developed a new mobile malware detection technique based on information flow.

Call Graphs gives information about API calls and shows relationship between methods in applications. However, Feng et al (2020) revealed that use of precise call graph consumes resources and results in poor efficiency. The work by Yang et al, (2021) also raised the same concern of resource consumption. They noted that thousands of APIs are provided by android platforms, therefore analysis of all function call graphs would consume large resources. Wang et al, (2022) on their part stated that API call sequences are usually too long, therefore a truncated segment of the API call sequences or its statistical features in malware detection was used by some researchers but it suffers from high false alarm because execution order information of the applications are lost. From the foregoing it can be seen that code coverage limitation, trigger conditions, dependence on expert knowledge for

feature selection, high volume of resource consumption, imbalance dataset among others constitute challenges that can affect malware detection where API call features are employed.

RQ3- Are there appropriate methods available to address the challenges of android malware detection using API features?

Various researchers have proposed solutions to address perceived challenges in android malware detection. Alzaylaee et al, (2020) proposed DL-Droid, which used dynamic stateful input generation to enhance code coverage. DL-Droid also focused on dealing with code obfuscation and employed real devices to avoid anti-emulator tendencies of malware. Their approach to detection considered extracting static features of android permissions before program execution, and then extracting the API calls and intents dynamically during execution using Multilayer Perceptron classifier, MLP. Kumar and Ciza, (2021) in their research also attempted to deal with the problem of code obfuscation with static analysis. The authors identified suspicious API classes and methods used by Malware apps and generated MSA (Multiple Sequence Alignment) corresponding to API class sequences present in malware applications to overcome malware evasion techniques using machine learning classifier Profile Hidden Markov Model(PHMM).

Elsewhere, Arindam et al, (2020) proposed a light weight detection framework that operates upon only 50 features. The method adopted for their research analyzed API calls extracted from smali code, maps the API Calls to certain features (permission) and constructed a frequency-based feature vector for each application. The approach bridges the existing gap of high need of resources such as time, space and computational power in existing work. Similarly, a malware detection system, MAPAS which learns behaviors of malwares by using a deep learning algorithm (CNN) and detects malware based on common patterns of API call graphs of malware was proposed to effectively deal with issues of high computing resources (Kim et al, 2022). Stuart et al, (2021) on the other hand addressed issues of expert domain knowledge required for feature engineering and attendant inability to detect zero

day scenarios. They proposed a solution that used Convolutional Neural Network CNN to learn from a limited set of only 210 proprietary Android API packages that have no expert pre-categorization as sensitive or otherwise.

Exploiting the advantages of deep learning to address featuring engineering in classical machine learning, (Hadiprakoso et al 2020) designed a new system that compiles static and dynamic analysis features such as API call sequence, system command, manifest permission, intent and process the data using a deep neural network. (Wang et al, 2022a) introduced an efficient extraction algorithm for API call sequences, which contains two sub-algorithms. The first sub-algorithm simplifies the function call graph from a multigraph to a simple graph, and the second develops a pruning depth-first search. The authors posit that the existing API call sequence extraction methods are laborious and time- consuming, which seriously decreases the efficiency of static analysis, hence the need to adopt the methodology.

Wang et al, (2022b) addressed the challenge of high false alarm caused by use of statistical features or truncated segments of API call sequences with their proposed FGL_Droid. FGL_Droid converts dynamic API call sequence into a function call graph, joins the function call graph feature and extracted permission request feature to carry out malware detection. The function call graph retains most of the application execution order information with significantly reduced sequence size and missed behavior information during conversion is made up for with the advanced features of permission requests extracted.

RQ4 - Are the available detection methods effective?

The effectiveness of detection methods used in various studies was largely measured with the performance evaluation metrics Accuracy, Precision, Recall and F1 Score. The approaches to android malware detection employed had good results with the various metrics hence taken to be effective. However, these performances could be investigated further as the type, size and nature of the sample of dataset used could result in bias in performance.

RQ 5 - Are there datasets being used in the primary studies?

The datasets used by selected primary studies and their frequency is represented in the figure 3.

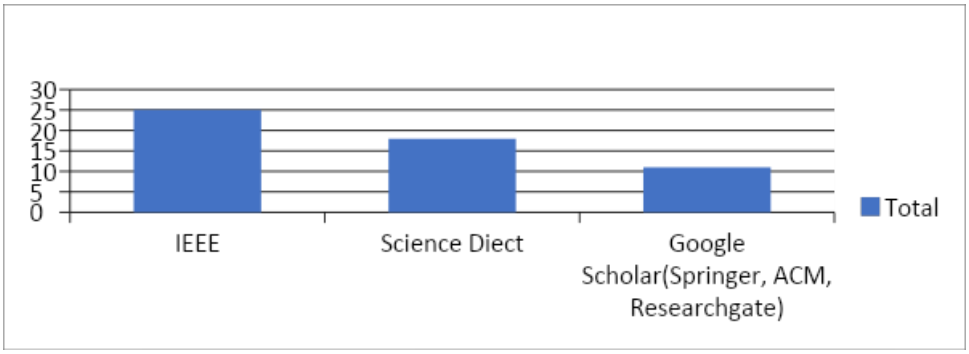

Figure 3: Frequency distribution of papers from Database Sources

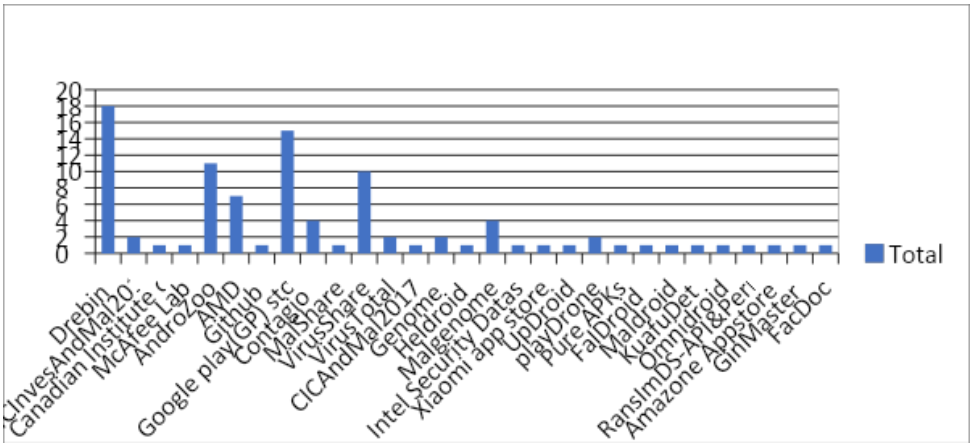

Figure 4: Frequency Distribution of Datasets used

5. Conclusion

This review examined literature on android malware detection with API call features. The study revealed that various researchers have employed the use of static analysis and dynamic analysis using API call features in detecting malicious android applications. The review also outlined different methodologies/algorithms with their performances and datasets used by authors in malware detection. Furthermore, challenges with android

malware detection in the research area which include code obfuscation, dynamic code loading issues with static analysis as well as limited code coverage, high resource consumption, execution time and trigger condition issues with dynamic analysis were identified among others. Various machine learning/deep learning methods and approach employed in the detection and analysis as revealed in the review provide useful insight researchers can leverage to improve android malware detection and is considered as a valuable contribution in this work.

References

Alzaylaee M. K., Yerima S. Y., Sezer, S. (2020). "DynaLog; An automated dynamic Analysis Framework for characterizing android applications". Journal of Computers and Security. 89 https://www.sciencedirect.com/science/article/pii/S0167404819300161

Arindam R., Singh J. S., Gitanjali J., Kapil S. (2020) "Android Malware Detection based on Vulnerable Feature Aggregation" in Procedia Computer Science. 173, 345-353.

Ashawa M.A., Morris S. (2019). Analysis of android malware detection techniques systematicreview,dspace.lib.cranfield.ac.uk.8(3):177-187 https://dspace.lib.cranfield.ac.uk/handle/1826/15057

Ehsan A., Catal C. Mishra A. (2022). Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review. Sensors, mdpi.com. 22(20): 7928 https://www.mdpi.com/1424-8220/22/20/7928

Elayan N.O., Mustafa M. A. (2021). "Android Malware Detection Using Deep Learning" in Procedia Computer Science. 184:847 – 852, https://www.sciencedirect.com/science/article/pii/S1877050921007481.

Feng P., Ma J., Li T., Ma X., Xi N., Lu D. (2020)."Android Malware Detection Based on Call Graph via Graph Neural Network". https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9353764.

Feng P., Ma J., Li T., Ma X., Xi N., Lu D. (2021). Android Malware Detection via Graph Representation Learning. Mobile Information Systems, hindawi.com, https://www.hindawi.com/journals/misy/2021/5538841/

Hadi Prakoso R. B., Buana I. K., Pramadi Y. R. (2020). "Android Malware Detection Using Hybrid-Based Analysis and Deep Neural Network" https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9332066.

Kim J., Ban Y., Ko E., Cho H., Yi J.H. (2022). MAPAS: a practical deep learning-based android malware detection system. International Journal of Information , Springer, https://doi.org/10.1007/s10207-022-00579-6

Kumar S., Ciza T. (2021). "ProDroid; An Android malware detection framework based on profile hidden Markov model" in Pervasive and Mobile Computing 72. https://www.sciencedirect.com/science/article/pii/S1574119221000109.

Kumar S., Mishra D., Shukla S. K. (2021). "Android Malware Family Classification: What Works API Calls, Permissions or API Packages" https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9699322

Lu X., Jiang F., Zhou X., Yi S., Sha J., Pietro L. (2019). "ASSCA: API sequence and statistics features combined architecture for malware detection" in *Computer Networks*, **Vol 157**, 99-111. https://www.sciencedirect.com/science/article/pii/S138912861930461X

Michele S., Davide M., Francesco M,Aaron V. C., Fabio M., Giorgio G. (2019)."On the effectiveness of system API-related information for Android ransomware detection" in Computers and Security. 86:168-182, https://www.sciencedirect.com/science/article/pii/S0167404819301178

Moutaz A., Mamoun A., Andrii S., Abdel Wadood M., Albara A. (2020). "Intelligent mobile malware detection using permission requests and API calls" in Future Generation Computer Systems."107:.509-521, https://www.sciencedirect.com/science/article/pii/S0167739X19321223.

Pang J., Bian J. (2019)."Android Malware Detection Based on Naive Bayes", https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9040796

Roopak S., Tony T., Sabu E. (2020). "A TAN based hybrid model for android malware detection" in Journal of Information Security and Applications. **54**: 102483 - 102483 https://www.sciencedirect.com/science/article/pii/S2214212618308263.

Shen F., Vecchio J. D., Mohaisen A., Ko .S Y., Ziarek L. (2019). "Android Malware Detection Using Complex-Flows" in IEEE Transactions on Mobile Computing. **18**(6) https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8423084

Stuart M., Niall M., Jesus M., Paul M. (2021). "Multi-view deep learning for zero-day Android malware detection" in Journal of Information Security and Applications. 58(3) https://www.sciencedirect.com/science/article/pii/S2214212620308577

Wang W., Wei J., Zhang S., Luo X. (2020). Malware Detection Based on Local Sensitive API Invocation Sequences", IEEE Transactions on Reliability.69(1)174-187, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8776652

Wang W., Ren C., Song H., Zhang S., Liu P. (2022a). FGL_Droid: An Efficient Android Malware Detection Method Based on Hybrid Analysis. Security and Communications Networks, https://www.hindawi.com/journals/scn/2022/8398591/

Wang T., Xu Y., Zhao X., Jiang Z., Li R. (2022b). Android malware detection via efficient application programming interface call sequences extraction and machine learning classifiers. IET Software, Wiley Online Library, https://doi.org/10.1049/sfw2.12083

Yang Y., Du X., Yang Z., Liu X. (2021). Android malware detection based on structural features of the function call graph. Electronics, mdpi.com, https://www.mdpi.com/961828

Ya P., Xiuting G., Chunrong F., Yong F. (2020). A Systematic Literature Review of Android Malware Detection Using Static Analysis. 8: 116363-116379 DOI 10.1109/ACCESS.2020.3002842, *IEEE Access*.

MALARIA VECTOR CONTROL: CHALLENGES AND FUTURE STRATEGIES

WANGAI, L. N., KAMAU, K. K., WAIRIMU, B.M., KAMAU, L. N., NJUGUNA, M. N., ALWORA, A.

Department of Health Sciences, Kirinyaga University, KENYA

Correspondence: *kkamau@kyu.ac.ke*

Abstract

The current demand for the eradication of malaria marks a new-fangled chapter in the antiquity of this illness. This has been brought about by the striking decreases in malaria caused by administration of efficient medications and vector control. However, the emergence of pesticide resistance poses a challenge to this approach. Alternative tools must be developed to continue supporting or potentially replace insecticide-based vector control methods. Long-lasting insecticidal nets (LLINs) and indoor residual spraying (IRS) continue to be the mainstays of the majority of National Malaria Control Programs in Africa, despite the large number of promising control tools tested against mosquitoes. These strategies are not enough to successfully control malaria. While these techniques are successful in lowering malaria incidence, their overall effectiveness in lowering malaria prevalence is often limited. Additionally, efficiency of LLINs and IRS is threatened by the rising rates of pesticide resistance in the targeted mosquito populations. Thus, although larvicidal treatments can be beneficial, using them in rural regions is not advised. To enhance mosquito vector control efforts and improve their quality and delivery, it is important to focus on integrated approaches. Successful malaria eradication requires close collaboration between parasitologists and entomologists, along with a comprehensive evaluation of epidemiological impact of innovative mosquito vector control strategies. This review discusses current malaria vector control strategies and highlights challenges, and promising tools that are expected to contribute to malaria eradication.

Keywords: *Malaria, Vector Control, Current Challenges and Future Strategies.*