ANDROID MALWARE DETECTION MODEL WITH NEGATIVE SELECTION ALGORITHM AND WHALE OPTIMIZATION ALGORITHM

BY

NDATSU, Zainab

MTECH/SICT/2017/6717

A THESIS SUBMITTED TO THE POSGRADUATE SCHOOL,

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGERIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTERS OF TECHNOLOGY (MTech) IN CYBER SECURITY SCIENCE

OCTOBER, 2021

ABSTRACT

This research is Android malware detection model with Negative Selection Algorithm and Whale Optimization Algorithm. Negative selection algorithm has principles and mechanisms to solve problems including the detection of malware. Negative Selection Algorithm with whale optimization was used as optimizer for the selection of best features of android application. The aim is to propose an android malware detection technique for the detection of android malicious applications. The model consists of the basic approach and techniques to achieve good model for the detection of android malicious applications. The research methodology of Data Analysis, which involves validation through experimentation, is employed to achieve this. The results show that the models of selected permission-based features are more accurate than those models without the selection of features. The true positive rate and false alarm rate of selected features are also in better forms than those of classifying features without selection. This research development of Android Malware Detection Model which achieved an improvement in detecting malware with a result of 98.7% performance accuracy.

TABLE OF CONTENT

Table of Contents

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background to the Study

This thesis presents Android malware detection model with Negative Selection Algorithm and Whale Optimization Algorithm. The model developed consists of the basic approach and techniques in achieving good model for the detection of android malicious applications (Milosevic *et al*., 2017). Negative Selection Algorithm is an immune algorithm that has deficiency in the selection process to select the good features from the features set. Whale optimization algorithm (WOA) has been used as optimizer for the selection of good features of android application in order to achieve better results and model performance. The level of development of mobile devices has brought to people's ways of living cannot be overemphasised. The early phone can only be used to place call and sending text messages without the need of connection to any financial transactions. The arrival of smartphones has revolutionised the mode of communication and data processing. For example, with smartphone people no longer visit cyber café where they can browse for various data communication functions. They can remains in their convenient place and perform related transaction. Again, with mobile phone, related banking transaction can be performed; foreign transaction can be carried out on the smart phone such as video conferencing, online application and many others. Although the emergence of smartphone has also brought little disadvantages like loss of job, loss of valuables at times over fraud transaction and others. However, the merits of smartphone outweigh the associated demerits. According to the research of Milosevic *et al*., (2017), it was predicted there will be about 6.1 billion mobile device users by 2020 .The smartphone arrives with different operating system (OS) which includes windows operating system, iOS, and android operating system. Among these, android OS is the most popular and friendly due to its openness and application availability in different open sources. This android OS is own by Google corporation which has made the android application free in an open market. This openness of application has made android a soft target for malicious software (Adebayo *et al.*, 2013). These malware targets at android application include Trojan, Spyware, Ransomeware, Virus, and Worm. Malware on a smartphone attacks by launching an attack by starting a new process and redirecting the programme flow of a valid application (e.g. messaging activity) to execute its malicious code within a genuine security context. Android applications consist of basic two features: permission based and application programming interface (API). These two features are normally considered in the classification and detection of malicious application.

Malicious software seeks to steal personal and sensitive data from mobile devices by exploiting device vulnerabilities and convincing users to install applications that give the malware creator unauthorised root access to infect the device. Bluetooth, SMS, GPS, phone jail breaking, and premium rate-based malware attacks are all possibilities.

In 2017, Nokia released the "Threat Intelligence Report," which examines malware behaviour in communications networks and found that 72 percent of network infections target smartphones, with Android accounting for 68.50 percent, Windows PC accounting for 27.95 percent, and others accounting for 3.54 percent. The majority of infections, 69 percent, target Android devices (Alqahtani *et al*., 2019).

The thymus is responsible for T-cell development and is protected by a blood barrier that effectively keeps nonself antigens out of the thymic environment. As a result, the majority of components identified in the thymus are self-representative rather than non-representative. As a result, through a process known as negative selection, T-cells with receptors capable of recognising these self-antigens presented in the thymus are weeded out of the T-cell repertoire (Castro & Timmis, 2003). In this thesis, negative selection algorithm with whale optimization algorithm shall be used to optimize machine learning applications for android application's classification in either good applications or malware. In other to do this, permission-based features of android were optimized using Negative Selection Algorithm with Whale Optimization Algorithm (NSA-WOA). The aim of the research is to improve performance accuracy with Negative selection algorithm and whale optimization algorithm for selection technique to optimise the data function. Previously use detection methods usually either or not applied feature selection techniques to build models. The use of this feature selection usually has overall effect on the accuracy of the detector. The effort to enhance detection accuracy, which determines the accuracy of detection models, remains a daunting undertaking. This study uses the whale optimization method to improve the negative selection algorithm and classify the features using three distinct classification algorithms: Nave Bayes, Random Forest, Decision Tree classification techniques, and neutral network.

The rest of the thesis is structured: The second chapter discusses relevant efforts to this study. Chapter three discusses the proposed classification model with its constituent framework, chapter four is discussion of result and chapter five is the conclusion, recommendation and contributions to knowledge.

1.2 Problem Statement

Android malware have continued to attract the operation of smartphone with android operating system due to the availability of its applications in the open market. These researches Brown *et al*.,(2017) and Dewanje & Kumar (2020) have presented techniques to detect the android malicious application (malware) with low accuracy and low false positive rate. However, this technique is faced with the following limitations:

1. The existing techniques use ineffective feature selection technique which resulted into low accuracy and high false positive rate.

2. The classification algorithms used, due to the presence of redundant features yield classification model with low accuracy.

1.3 Aim and Objectives of the study

The aim of this research is to develop android malware detection model for the detection of android malicious application. The objectives are to:

i. Optimise negative selection algorithm (NSA) with whale optimization algorithm (WOA) to improve feature selection.
ii. Develop android malware detection model with an improved technique above with classification algorithms.
iii. Evaluate the detection model using Accuracy, False Positive Rate and True Positive Rate performance metrics.

1.4 Scope of Study

The research is limited to detection of the permission features of android based application using the negative selection algorithm with whale optimization algorithm for the detection of android applications into malicious or benign.

1.5 Limitation of study

The limitations of the study are highlighted :

1. This model involves the use of only permission based attributes of android application for the model to be effective.
2. Many android attributes contain no identifiable features.

1.6 Significance of the Study

In the study of Brown *et al.*,(2017) Negative Selection algorithm has been used for attribute selection in malware detection, but the algorithm was deficient in the selection process. An improved data model will subsequently improve the performance in terms of detection accuracy and reduction in false positive rate.

Therefore, the use of WOA to optimize negative selection algorithm will lead to better result in detecting android malware and this study will be of abundant relevance as well as worth to future malware detection.

1.7 Motivation of the study

The use of negative selection algorithm for the detection of malware on the desktop computer and in other areas has been reported but requires improvement in the selection process. The basic motivation of this research is to enhance the performance of the negative selection algorithm using whale optimization algorithm to improve detection model and the detection strategy.

1.8 Justification of the study

The Whale Optimization Algorithm (WOA) used in this study as an optimizer is expected to produce detection results with high accuracy and low false alarm rate when compared with existing models.

CHAPTER TWO
2.0 LITERATURE REVIEW

2.1 Preamble

This section examines and discusses the existing related works in the use of Negative Selection Algorithm and Optimization Algorithms to improve the detection accuracy of detecting android malware models.

2.1 Malware

Malware is a program developed with the intention of causing harm to computer system and it is also referred to an Intrusion program, which is created by malware hackers for purposes of financial gain, destruction, challenges or retaliation (Adebayo *et al.*, 2013). Computer malwares include computer viruses, worms, Trojan, Botnets, Spyware, Keyloggers and many other destructive malwares. A malware detector is a system introduced to analyze and identify malware. A malware detector can be a commercial virus scanner that detects malware using binary signatures and other heuristic rules and algorithms, or a firewall that monitors electronic device gateways. Malware is a term used to describe computer programmes that have malevolent intentions (Adebayo *et al*., 2013).

New malware which appears every day is believed to be the most altered versions of the previous one using enlightened replication method. An enormous number of samples have been used. Having a large number of data sets contributes to the prophetic ability and dependability of the built model that gives a presentable outcome(Dewanje & Kumar, 2020). Detecting, analysing, and removing malware is in high demand across computing and mobile platforms. Many researchers have developed many approaches and algorithms, including (Christodorescu *et al*., 2005; Siddiqui 2008; Shabtai *et al*., 2011; Eder *et al*., 2013; Mirjalili & Lewis, 2016).

2.1.1 Malware types

They are in different forms and can be found in more than one class (Tahir, 2018). They are:-

1. Worms which are malicious part of program that duplicate, transmit over storage medium, consume internet and system resources which leads to low performance of the system. They can duplicate themselves, antivirus scanners are able to identify these codes because of multiple existence (Tahir, 2018)..
2. Virus which affects system and other files by duplicating itself. It attaches itself to files mostly the executable files, programs and spread over the system and network system which lead to low performance and denial of service.
3. Rootkit which creates an environment for itself and other malware by taking control of the operating system. Avoid to be detected and consider as normal applications by malware antivirus in the system by using the masking techniques.
4. Trojan Horse which serves as useful package but it has negative reasons. They don’t duplicate but it is transmit into a system by network interaction for instance downloading. It takes vital evidence, users' activities can be observed, and data on the system if they exist can be deleted, altered, or damaged.
5. Spyware which takes user’s vital information with their knowledge or spy on activities of the user. It however is connected on the system without their acquaintance and unknowingly collects the material and give to the designer. Big establishment for instance Google also utilise spyware to get their users information requirement.
6. Cookies which are text files that contain information that is saved on the user's computer by the web browser for future use. Cookies appear to be harmless, but when they are employed by malware, they constitute a menace.

7. Adware which place advertisement on users systems without their authorization and disturb the present action perform by the user. Their purpose is to get monetary again. It has negative effect as other malicious program.
8. Sniffers which are computer program that watch and write the traffic of the network. Users' activity can indeed be monitored, and content on the platform can be erased, edited, or damaged if it exists.
9. Botnet which gives hacker access to control and harm the system. They are a collection of compromised systems that are managed by hackers/attackers and used to carry out nefarious operations without the owner's knowledge. Denial of service attacks can be form by them, spam messages be send by them and also steal users information.
10. Spam which is Junk emails which are identical emails send to many users at the same time. It takes many of bandwidth and also causes low performance of the system.
11. Keyloggers which is type of spyware that make use of record key strokes to access credit card details, passwords, and other vital and important figures. It gets into a system from installation of malware program or visiting any infected site by user.
12. Ransomware which became frequent threat for network computer. Ransomware encrypt users data, stop some software and denial users the use of operating system unless their requests are met. Their request is mostly in form of finance. it is not guaranteed that the system will be released. (Tahir, 2018).

2.1.2 Different techniques for malware detection

Detection techniques of Malware can be divided into three (3) groups heuristic based, signature based and specification based Tahir (2018). The three techniques detect and identify malware and ensure systems safety from those malwares that can cause a potential loss data and resources.

1. Heuristic based

In order to detect and address known and undiscovered malware threats, this detection technique identifies or distinguishes between normal and aberrant activity of a computer system. There are two (2) steps to the detection process. The operations of the computer system are examined without being attacked in the first stage, and a record of critical information is preserved that may be tested in the event of an assault. In the second stage, this difference is examined in order to identify a specific type of malware.

The following three (3) key modules are contained in the activity detector utilised in heuristic based techniques. These techniques are Collection of Data, Algorithm Matching and Interpretation.

Collection of Data: in this module the collection of data is done either dynamic or static.

Algorithm Matching: This module is in charge of matching the activity signature with the information converted from the interpretation module. Activity detector explains the functionality of how the three (3) modules work together.

Interpretation: This module interprets and transforms the collected data from data collection module into intermediate form.

2. Signature Based

In malicious codes a line of bit defines as signature is encoded in its code to detect malware type in future. The detection technique based on signature is used by recent antivirus software. The antivirus software separates the code of the infected file and searches for patterns that are specific to a malware kind. Malware signatures are stored in a database and then compared throughout the detection process. String or pattern scanning or matching is another name for this type of detection technology. It can also be static, dynamic, or hybrid.

3. Specification based

Programs are examined in terms of their specifications in a specification-based detection technique, which looks for normal and anomalous behaviour. The main difference between this technique and heuristic based detection techniques is that Heuristic-based detection techniques used machine learning and AI algorithms to detect valid and invalid programme activity, whereas specification-based detection techniques analysed the behaviour defined in the system specification. This approach involves a manual comparison of a system's regular operations. By reducing false positives and raising false negatives, it overcomes the limitations of heuristic-based approaches. The specification-based detection method is derived from the heuristic-based detection technique, and each malware detection technique can be hybrid, dynamic or static (Tahir, 2018).

2.1.3 Detection techniques for android malware

The malware detection technique is divided into classification, analysis, malware detection and eventual containment (Adebayo & AbdulAziz, 2014). Classification techniques which include association mining, machine learning, rule-based decision tree and many others have been used in the classification of computer programs into malicious or benign set. Malware analysis is the process of finding instances of malware utilising various schemes and properties of known malware characteristics. On the other hand, malware detection entails quickly identifying, detecting, and validating any incidence of malware in order to prevent additional system harm. The final task is to contain the infection, which entails attempting to stop the pathogen's aftereffects and preventing future system harm.

Several techniques have been used for detecting android malware applications. These techniques are pattern recognition detection, anomaly detection, rule based detection, misuse detection among others.

2.1.2 Android malware analysis

Android malware analysis is the analysis of android application to examine malicious contents. Android malware analysis is divided into dynamic and static analysis. Due to numerous obfuscation measures, static analysis has become ineffective. Dynamic analysis fills in the gaps left by static analysis (Adebayo & AbdulAziz, 2014). In the dynamic analysis, strategies such as monitoring changes to the system registry and inserting hooks into the system interface or library were utilised. Dynamic analysis, on the other hand, can be prone to substantial false negative and false positive rates because the heuristics are not based on the core properties of malware.

Static analysis statistically examines the code of program rather than really running it. This static analysis approach has the advantage of being able to analyse all of the code and possibly record the entire programme performance, liberated of any one pathway performed throughout execution time. Furthermore, the ability of statics analysis' to spot fresh malware or malware variants is limited.

Some recent researches have to use the Dynamic and Static Analysis. Which involves both static and dynamic techniques in a simultaneous form to examine the malicious programs?

The existing detection techniques used Particle Swarm Optimization inclusive Apriori Algorithm (Adebayo & AbdulAziz, 2014),with Aapriori association analysis for its signature extraction (Muazzam *et al*., 2008) which was typified with drawbacks. Particle Swarm Optimization was utilised by the authors to optimise the development of candidate detectors (flag bearers), it will enhance the identification procedure by lowering false positives and enhancing true positives, a large number of Android applications were collected both malicious and benign. After a thorough study of programme samples, the features from both samples were retrieved. To choose high-ranked features from the set of created features, three feature selection procedures were applied. The association rules were created using the features that were used to detect malicious Android applications. (Adebayo & AbdulAziz, 2019).

2.2 Negative Selection Algorithm

T cell precursors travel from the bone marrow to the thymus in the biological immune system, where they grow into T cells. T cell progenitors lack the ability to express T cell markers such as the T cell Receptor. A Negative Selection Algorithm is distinguished by a specific matching algorithm based on a similarity distance or measure. (Dasgupta, 2011) In the Biological Immune System, this algorithm simulates the development and interaction of T cells. Antibodies against all known self-patterns are created using negative selection as a precursor Figure 2.1 shows Negative selection process. Because any antibody matching self is deleted before fielding, false-positive errors against a static self are eliminated.

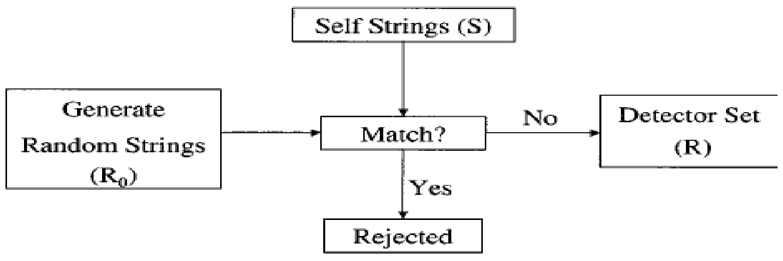
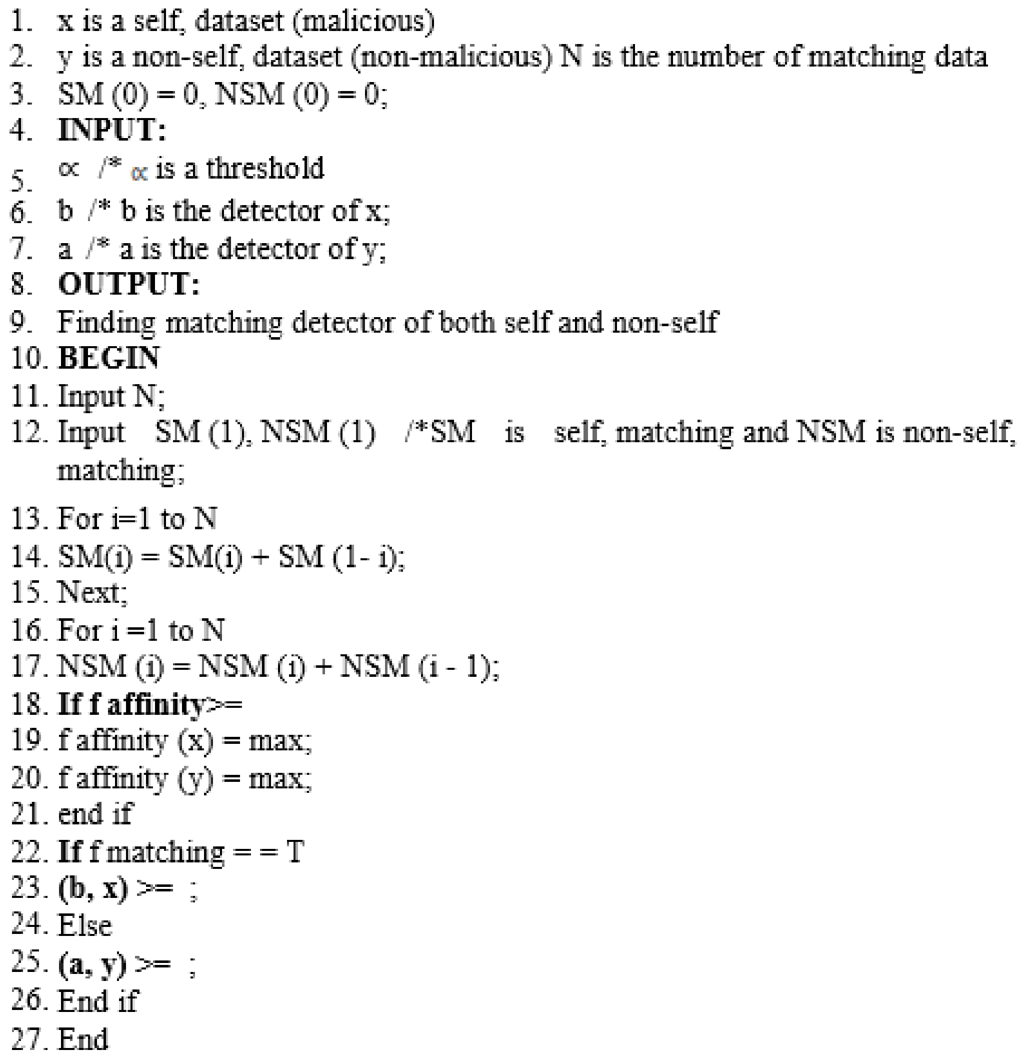


Figure 2.1: Negative Selection process (Castro & Timmis, 2003)

2.2.3 Negative Selection Algorithm (Idris 2014)



```
1.  x is a self, dataset (malicious)
2.  y is a non-self, dataset (non-malicious) N is the number of matching data
3.  SM (0) = 0, NSM (0) = 0;
4.  INPUT:
5.  ∝  /* ∝ is a threshold
6.  b  /* b is the detector of x;
7.  a  /* a is the detector of y;
8.  OUTPUT:
9.  Finding matching detector of both self and non-self
10. BEGIN
11. Input N;
12. Input   SM (1), NSM (1)   /*SM   is   self, matching and NSM is non-self,
    matching;
13. For i=1 to N
14. SM(i) = SM(i) + SM (1- i);
15. Next;
16. For i =1 to N
17. NSM (i) = NSM (i) + NSM (i - 1);
18. If f affinity>=
19. f affinity (x) = max;
20. f affinity (y) = max;
21. end if
22. If f matching = = T
23. (b, x) >=  ;
24. Else
25. (a, y) >=  ;
26. End if
27. End
```

2.3 Related work

Some closely related work that use almost the same approach as this current study are (Shabtai *et al.,* 2011), (Siddiqui, 2008) and (Agrawal & Srikant ,1994). Frequent technique used by malware writer is code obfuscation (Abhijit *et al*, 2008) that prevents the detectors from detecting its present. This approach can be phenotypic or transformational in nature. To escape detection, a metamorphic virus hides itself completely, whereas a polymorphic virus uses code insertion and transposition to obscure its decryption loops (Christodorescu *et al*., 2005). In order to carry out its destructive deeds, metamorphic malware uses techniques such as register renaming, dead code insertion, block reordering, and command substitution.

Another strategy used by malware authors is to modify and include new behaviour in their software in order to improve its strength and viability.

Static analysis, dynamic analysis, and a hybrid of both static and dynamic approaches are three types of malware analysis and detection solutions that have been used in the past. Static analysis is the method of quantitatively examining a program's code without actually running it (Shabtai *et al.*, 2011). Without installing the software, static analysis analyses it for harmful patterns. In this static analysis on a smartphone, the sandbox decompresses installation files and disassembles the proper executable. The static analysis approach has the advantage of covering the full code and, as a result, possibly capturing the entire programme behaviour, independent of any one path taken during run-time. The statics analysis, on the other hand, is limited in its ability to detect new malware or malware variants.

In contrast, dynamic analysis is required to compensate for the inadequacies of static analysis produced by various obfuscation tactics, rendering static analysis worthless. Dynamic analysis isolates the application in a sandbox, which intervenes and logs low-level interactions with the system for later study using an Android emulator, which is commonly used to test as well as debug conventional Android apps. Eder *et al*., (2013) developed the Framework for Analyzing Android Applications (ANANAS) and a lightweight Malware Detection System for Android-based mobile devices (ANDROMALY), they are also studies on malware detection on mobile platforms (Shabtai *et al*., 2011).

A new detector model and matching rule model were created for comparing both non-self and self, in order to improve the performance of the detector model, the negative selection algorithm with their unique matching technique help their model to solve the limitation of a normal negative selection algorithm in defining harmfulness of self and non-self. Their paper adopt spam model based and neural network on Negative selection algorithm for solving complex problems in spam detection (Idris, 2012).

A new research to analyse, collect applications on smartphone and their file activity in Android research. With the collaboration of Android user community, that will be able to differentiate between malicious and benign applications of the same name and version, matching anomalous behaviour of known applications, that studying software behavioural functions in Android framework is a possible way for malware detection (Kinholkar, 2015). The research of a Multiple-Detector Set Artificial Immune System (MAIS) and a validation scheme to stop the increasing threat of Android malware (Brown *et al*., 2017). The research detect mobile malware based on the information flows in Android apps.This approach achieved a 93.33% accuracy with a true positive rate of 86.67% and a false positive rate of 0.00%.

The research of An Improved Negative Selection Algorithm-Based Fault Detection Method describes an intelligent fault detection system that incorporates machine learning and the Negative Selection Algorithm (NSA). Unlike existing model-based and signal processing methods, their suggested NSA utilised just self-patterns for detector generation, with a further training stage integrated for identification of unseen spaces in non-self-space and synthesis of new detectors. (Abid *et al*., 2020).

The study of Yang *et al*., 2020 suggest a negative selection strategy based on antigen density clustering in this paper (ADC-NSA). Partially produced detectors are constructed utilising density clustering to remove the repetitious coverage and gaps caused by randomly generated detectors in the original technique. However the algorithm has less detection in the area with less density (Yang *et al*., 2020).

2.3.1 Optimization Algorithm

This is review of optimization algorithms which are the Whale Optimization Algorithm and particle Swarm Optimization Algorithm.

2.3.1.1 Whale optimization algorithm

The best agents' estimation is the most crucial task in the whale optimization algorithm. The whale optimization algorithm is broken down into three categories (Chen *et al*., 2019).

2.3.1.1.1 Encircling of prey

The Humpback whale encircling its prey according to the following equations 1 and 2:

$$\bar{D} = \left| \bar{C}.\overline{X^*}(t) - \bar{X}(t) \right| \qquad (2.1)$$

$$\bar{X}(t+1) = \overline{X^*}\,(t) - \overline{A.D} \qquad (2.2)$$

Where t indicates the current iteration of the optimization. A A and C C are coefficient vectors of the equation.

X^* represents the position vector, which should be updated in each iteration for better solution attainment. Vectors A and C were calculate using the equation 2.3.

$$\bar{\mathrm{A}} = \overline{2\mathrm{a}}\,.\bar{\mathrm{r}} - \bar{\mathrm{a}} \qquad (2.3)$$

$$\bar{\mathrm{C}} = \overline{2\mathrm{r}} \qquad (2.4)$$

Where a is a linear vector that decreases in size from 2 to 0 over iterations, and r is a vector that is in the range [0, 1]. A search agent's position (X, Y) can be modified based on the position of the current best record (X*, Y*). By varying the value of vector A and C, multiple positions around the optimal agent emerge.

2.3.1.2 Bubble-net attacking method (exploitation phase)

To build the bubble-net behaviour of humpback whales, a spiral mathematical formulation is used between the position of whale and prey to replicate the helix-shaped movement of humpback whales.

2.3.1.3 Search for prey (exploration phase)

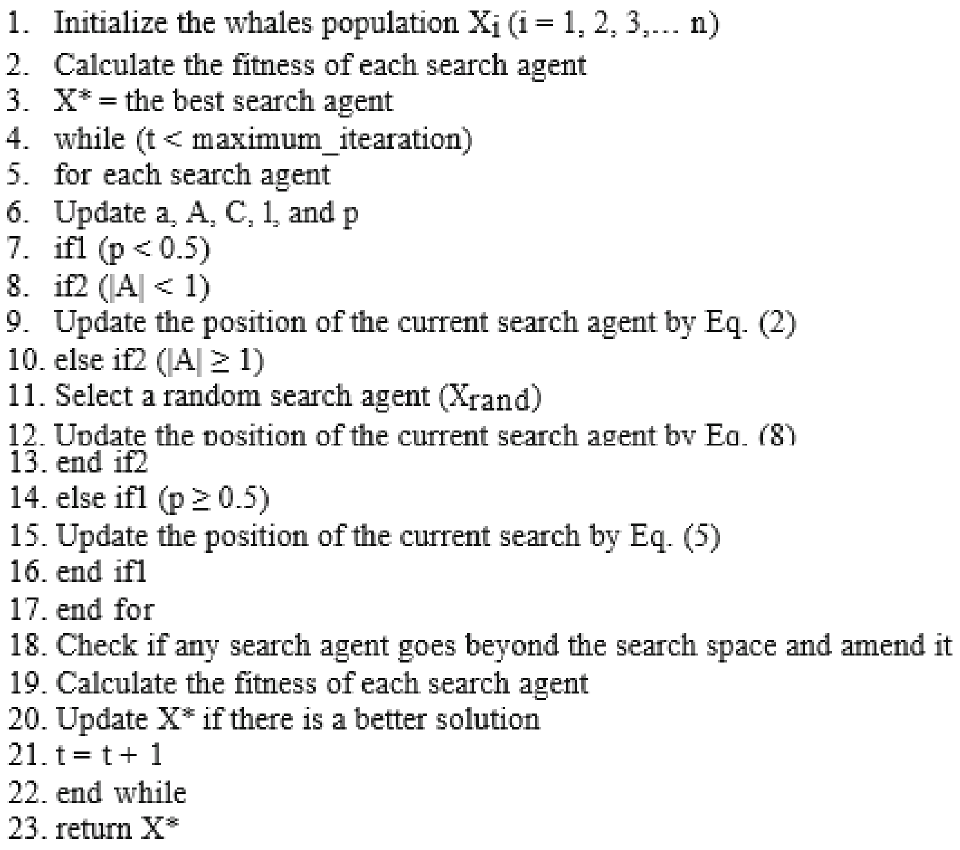
If A >1 or A - 1, the search agent is updated as indicated by a randomly chosen search agent in place of the best search agent to have global optimizers (Rana *et al*., 2020). In the exploration phase all the search agent are search to choose the best agent.

The bubble net attacking method and the shark net attacking method are the two basic ways to assault whales in the exploring phase (spiral updating position as well as shrinking surrounding mechanism). To improve WOA's exploration-exploitation trade-off, Improve Whale Optimization Algorithm (IWOA) combined WOA's operators with DE's mutation operator (Mostafa Bozorgi & Yazdani, 2019).

The authors of the report utilised two new efficient methods, Lévy flight (LF) and chaotic local search (CLS), that were simultaneously introduced to the WOA to promote the basic version's inclusive exploration and confined exploitation features, as well as to thoroughly investigate the searching capability of both Balanced Whale Optimization Algorithm (BWOA) and basic WOA. Furthermore, on a set of benchmark situations, the experimental results obtained by BWOA are compared to the results obtained by other common optimizers, and the findings can credibly demonstrate that the LF and CLS utilised in BWOA may improve the basic characteristics of conventional WOA (Chen *et al*., 2019).

The research of Rana *et al*., (2020) provides a thorough examination of the recently created swarm-based meta-heuristic optimization algorithm known as the Whale Optimization Algorithm (WOA), which is based on the humpback whale hunting manoeuvre (Megaptera novaeangliae). There is no extensive literature evaluation of the novel WOA that the writers are aware of (Rana *et al*., 2020).

2.3.2 Algorithm: Whale optimization algorithm (Rana *et al*., 2020)



```
1. Initialize the whales population $X_i$ (i = 1, 2, 3,… n)
2. Calculate the fitness of each search agent
3. X* = the best search agent
4. while (t < maximum_itearation)
5. for each search agent
6. Update a, A, C, l, and p
7. if1 (p < 0.5)
8. if2 (|A| < 1)
9. Update the position of the current search agent by Eq. (2)
10. else if2 (|A| ≥ 1)
11. Select a random search agent ($X_{rand}$)
12. Update the position of the current search agent by Eq. (8)
13. end if2
14. else if1 (p ≥ 0.5)
15. Update the position of the current search by Eq. (5)
16. end if1
17. end for
18. Check if any search agent goes beyond the search space and amend it
19. Calculate the fitness of each search agent
20. Update X* if there is a better solution
21. t = t + 1
22. end while
23. return X*
```

Kennedy and Eberhart introduced Particle Swarm Optimization (PSO) as a stochastic optimization method (Mariot & Leporati, 2015). PSO works by modelling a set of candidate solutions to an optimization issue as a swarm of particles that move through the search space in a coordinated manner. The choice of velocity parameters has been shown in the literature to have a significant impact on PSO performance. (Mariot & Leporati, 2015).

2.4 Classification Algorithms

The classification algorithms are data mining tools used for the detection of android application attribute to generate new model. The classification algorithms used with NS-WOA in this research are Naïve Bayes, Decision tree (J48), In the classification challenge for detection modelling, Random Forest and Neutral Network are two of the top methods. The model chose the best characteristics for the proposed detection model and modelled them using the previously discussed classification methods. In order to improve the structure of attributes by reducing redundant attributes, this study used an NSA-WOA to pick dataset features. Using the whale humpback search technique, the NS-WOA finds the best features from the dataset in this scenario. The classification methods are then applied to the best set of classification features. For the detection system, this yields the optimum classification model. Depending on the class to which they belong, the model can be used to identify Android applications as dangerous or benign.

2.4.1 Naïve bayes classifier

In the estimate of datasets with many features, naive Bayesian classifiers are commonly utilised. It is assumed that characteristics have separate variable distributions, allowing the feature to be estimated independently. The Nave Bayes classifier estimates data without taking into account the data's dimensionality.

2.4.2 Decision tree (J48)

Models build based on a recursive partition technique that has the interest of dividing the data set by applying a single variable at each level. This variable is selected with a given method.it define a set of cases in which all the cases belong to the same class.

J48 is a decision tree implementation based on the Classifier 4.5 (C4.5) decision tree. At each recursive stage of implementation, it uses knowledge acquired to select the best features. J48 decision tree is used to split and train the dataset. Split criteria of decision tree are Information Gain (I G) in these criteria data set are divide for training and testing.

Mathematical equation of Decision Tree (Carlos, Mantas & Abellán 2014)

$$IG(c, x) = H(c) - \sum_i P(X = X_i)\, H(C / x = x_i) \quad (2.5)$$

where

H(c) = ∑j P (C = Cj) log p(c = cj) (2.6)

Similarly

H(C/X = Xi) (2.7)

Let C be the class variable, $\{ X1, X2 \ldots . Xn\}$ the set of features, and X a general feature.

Info-Gain = (IG), P = probability

2.4.3 Neutral network

Neural networks are algorithms, which compute, from an input x (for example an image), an

output y. As shown in figure 2.2, this output is most often a set of probabilities: for example the first output is the probability that the image contains a cat (the closer this number is to 100%, the more it means that the algorithm is sure of itself), the second is the probability that the image contains a dog, etc. To simplify, we will consider in our examples only two classes: cats and dogs, but in practice one can consider an output y with several thousands of classes. The researchers also restrict the research to the example of images, but neural networks are also very efficient in recognizing texts or videos.

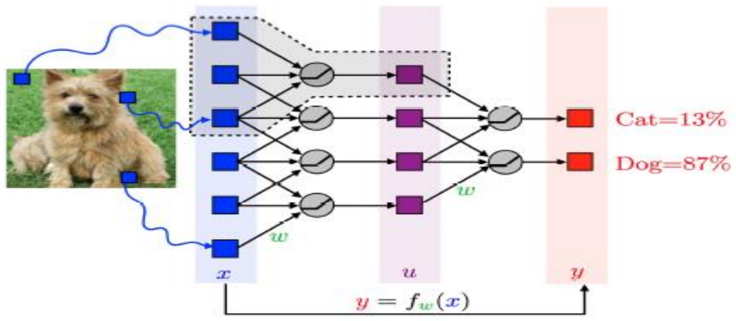


Figure 2.2: Neutral network obtain from Gabriel Peyré (2017)

2.4.4 Random forest

Random Forest is a collection of unpruned classification or regression trees constructed from training data bootstrap samples. In this algorithm, every node is usually split among small subset of the randomly selected input feature. a random forest is a predictor consisting of a collection of randomized base regression trees $\{r_n(X, \Theta_m, D_m,)\ m \geq 1\}$, where $\Theta_1, \Theta_2 \ldots$ outputs of a randomizing variable . These random trees are combined to form the aggregated regression estimate.

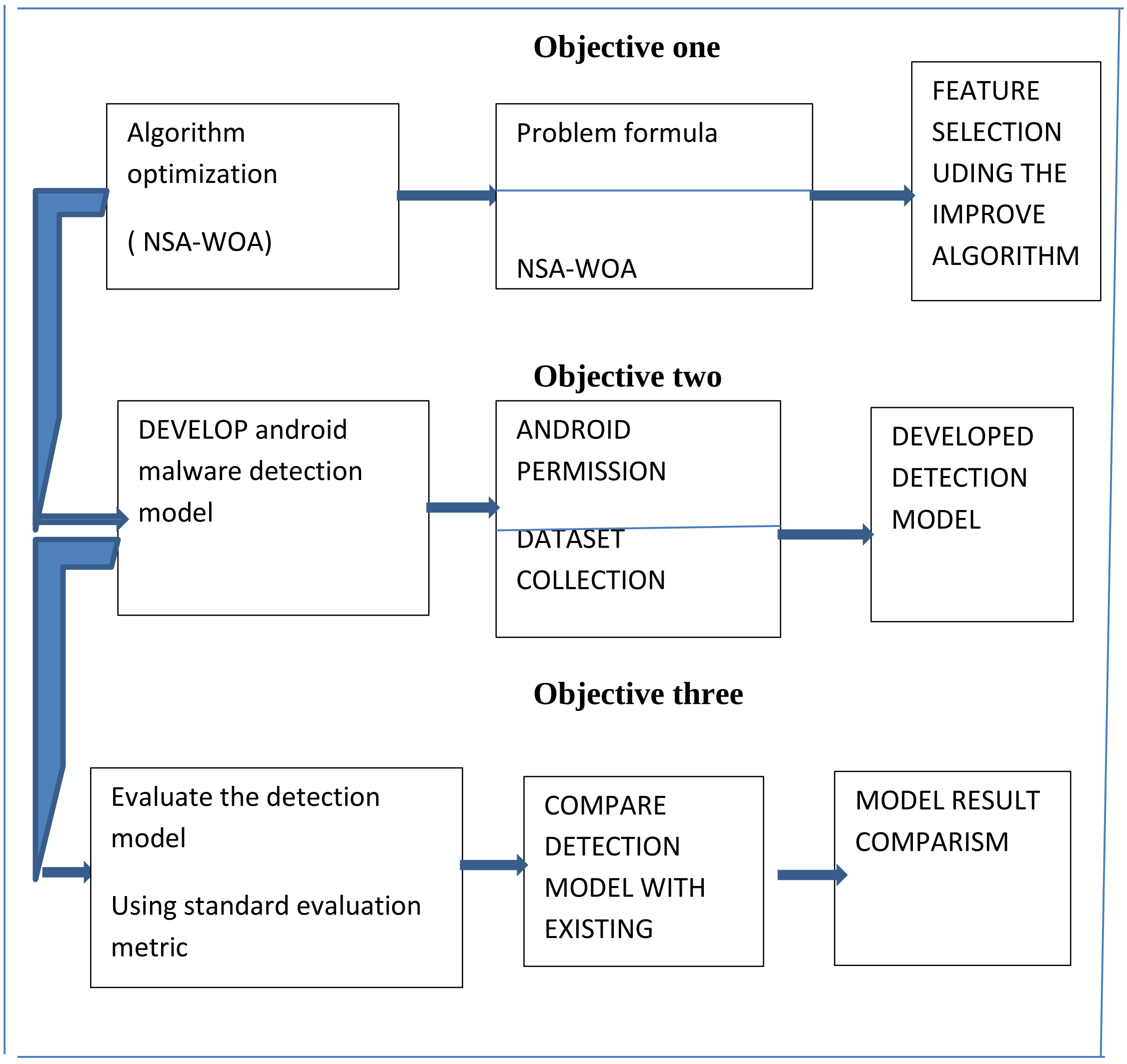
$$\bar{r}_n\ (X, D_n) = E_\theta\ [r_n\ (X, \theta, D_n)] \tag{2.8}$$

Where E denotes expectation with respect to the random parameter, conditionally on X and the data set Dn. In the following, to lighten notation a little, we will omit the dependency of the estimates in the sample, and write for example r_n (X) instead of rn(X, Dn). Note that, in practice, the above expectation is evaluated by Monte Carlo, i.e., by generating M (usually large) random trees, and taking the average of the individual outcomes (Biau, 2012)

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

This chapter clearly illustrates the methods involved in reaching the research's goal. It is vital to highlight that the research approach used in this study was Data Analysis, which included testing for validation. The subsequent sections present research processes in this study**.**



Objective one
Algorithm optimization (NSA-WOA)
Problem formula
NSA-WOA
FEATURE SELECTION UDING THE IMPROVE ALGORITHM
Objective two
DEVELOP android malware detection model
ANDROID PERMISSION
DATASET COLLECTION
DEVELOPED DETECTION MODEL
Objective three
Evaluate the detection model
Using standard evaluation metric
COMPARE DETECTION MODEL WITH EXISTING
MODEL RESULT COMPARISM

Figure 3.1: Research methodology based on study objectives.

3.1 Problem Definition

The problem is to detect applications of android into malicious or benign application. The android application is represented using AP, the malicious application is represented using MP while the benign application is defined using BA.

3.1.1 Problem identification

The problem was identified after thorough review of the literatures in chapter two. The problem include: the performance of the existing models were not measured, the false positive rate still relatively high, and the accuracy of the detection is not yet satisfactory. The successful application of negative selection algorithm in the detection of malware motivates this study to improve the algorithm. In order to remedy the lacuna of the existing researches, this research will use whale optimization algorithm to improve the performance of the new model in related to existing ones.

3.1.2 Problem formulation

The problem is a detection problem to detect malicious android applications into either malicious or good application. In order to do this, Android application executable are represented using AP while the extracted android features were represented using AF. The malicious application (MA) and benign application as (BA). The dataset and feature set were represented as:

$$AP = \begin{pmatrix} ap_{11} & ap_{12} & \cdots & ap_{1n} \\ ap_{21} & ap_{22} & & \vdots \\ \vdots & \vdots & & \\ ap_{m1} & ap_{m2} & \cdots & ap_{mn} \end{pmatrix} \quad (3.1)$$

$$AF = \begin{pmatrix} af_{11} & af_{12} & \cdots & af_{1n} \\ af_{21} & af_{22} & & \vdots \\ \vdots & \vdots & & \\ af_{m1} & af_{m2} & \cdots & af_{mn} \end{pmatrix} \quad (3.2)$$

The feature is represented in n by n dimensional binary vector where 1 represents the present of a feature and 0 denotes absence of a feature in an application.

$$AF = \begin{cases} 1 & if\ af\ is\ present\ in\ the\ application \\ 0 & if\ af\ is\ absent\ in\ the\ application \end{cases} \tag{3.3}$$

A function F such that AF ⟶ {MA, BA} is to be find using the defined selected features to train machine learning algorithms. This classification yields pairs

$$(ma_1, ba_1, ma_2\ , ba_2 \ldots\ ma_n, ba_n) \in (MA, BA) \tag{3.4}$$

3.1.3.1 Dataset analysis

Data collecting

The dataset consists 1000 good android application gathered from the official website of Google Play store and 1500 malicious applications downloaded from contagiomobile website and contagio minidump. Virus total (VirusTotal, 2015) online application was used to scan the benign applications to ensure they are truly good android files. Android features were extracted from the .apk executable. The features were normalized and transformed into numerical n by n dimensional vector. In the feature vectors, the binary number 1 is used to represent the presence of a feature while 0 is used to represent absence of a feature as represented in equation 3.3.

Data preprocessing: This involves loading the dataset and selecting the number of attributes.

3.1.3 Optimization Of Negative Selection Algorithm With Whale Optimization Algorithm

Whale optimization algorithm was use to optimise negative selection algorithm in generation of random strings as shown in figure 3,2

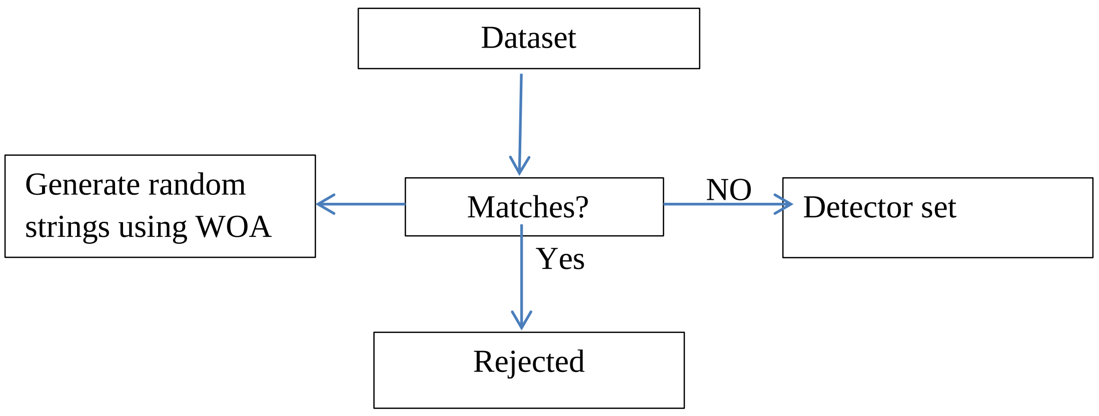


Figure3. 2: Negative Selection Algorithm with whale Optimization

```
1.  BEGIN
2.   x is a self, data set (malicious)
3.  y is a non-self, data set (non-malicious)
4.  N is the number of matching data SM(0)=0,  NSM(0)=0
5.   INPUT:
6.   ∝  /*  is a threshold
7.   b /*  b is the detector of x;
8.   a /*  a is the detector of y;
9.   OUTPUT:
10.  Finding matching detector of both self and non-self
11.  Initialize the whales population Xi (i = 1, 2, 3,… n)
12. . Calculate the fitness of each search agent
13. . X* = the best search agent
14.  while ( t < maximum_itearation)
15. for each search agent
16.  Update a, A, C, l, and p
17. . if 1 ( p < 0.5)
18. if 2 ( |A| < 1 )
19.  Update the position of the current search agent by  2.1
20. end
```

3.2 Design and Implementation of Detention Model

The proposed model involved the application of a bio inspired, whale optimization algorithm to improve the negative selection algorithm for feature selection and the classification algorithms for classification of the android application features.

3.2.1 Process of feature selection selected for training.

The process of feature selection of android permission based features.

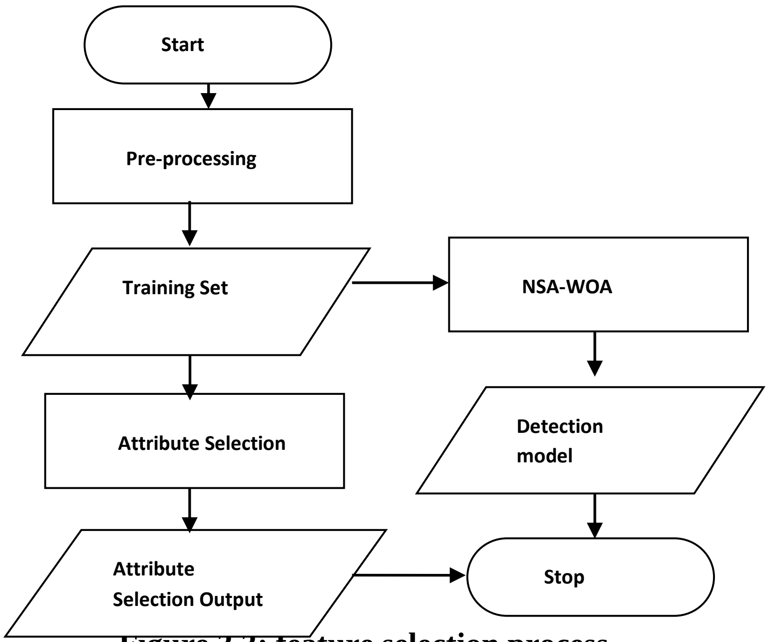


Figure 3.3: feature selection process

Negative selection algorithm (NSA) is improved using whale optimization algorithm. This improvement algorithm was used for the selection of features of android applications for the learners in order to build classification model. NSA-WOA was used as feature selection algorithm. The entire extracted permission-based features will be used for the selection process. The android permission-based features were picked using NSA-WOA and then trained with classification algorithms for improved performance in order to optimise Negative selection process with whale optimization method. The WOA employed the fitness function to figure out which features corresponded to the desired index fold. The suggested model framework and data model are depicted in Figures 3.3, 3.4 and 3.5.

3.2.2 Pseudo code of the Detention Model

```
1.  Start
2.  Load permission features
3.  If features is normalized go to step 5 else go to step 4
4.  Normalize features dataset
5.  Select features with NSA-WOA
6.  Use  dataset with Classification Algorithm
7.  If application is detected as malicious go to step 8 else go to step
    9
8.  Report as Malicious Application
9.  Report as Benign Application
10. Stop
```

Figure 3.3 pseudo code of the Model

3.2.3 The detection model

The malware detection model comprise of two main phases as shown in figure 3.4; the attribute selection and detection. The model starts by taking the dataset described earlier as input, then the optimized negative selection algorithm is use for the selection of features. The selected features were then used to feed the different classification algorithms to build the model. The results obtained are used to evaluate the model using three different performance metrics which including Accuracy, True positive rate and false positive rate.

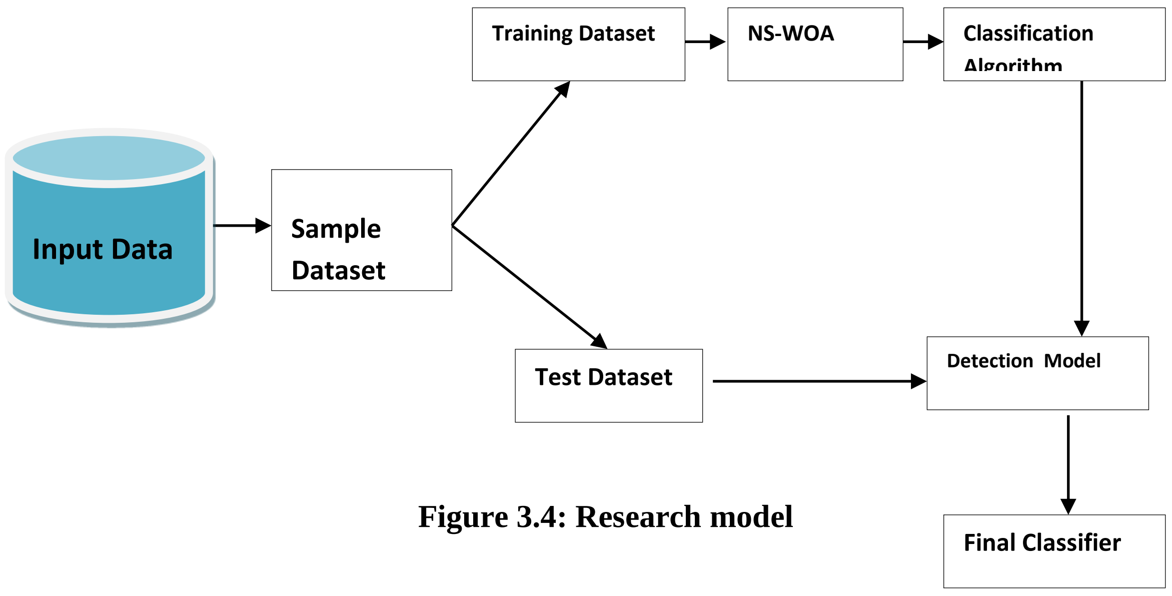


Figure 3.4: Research model

3.4.2 Flowchart of the model

The flowchart of the model shows flow of execution of the research model as shown in figure 3.5. The system start then load android permission features, if the feature are not normalize ,it normalise it, then select the feature with NSA-WOA ,after that it apply classification Algorithms discussed and test if the data are malicious or benign then end.

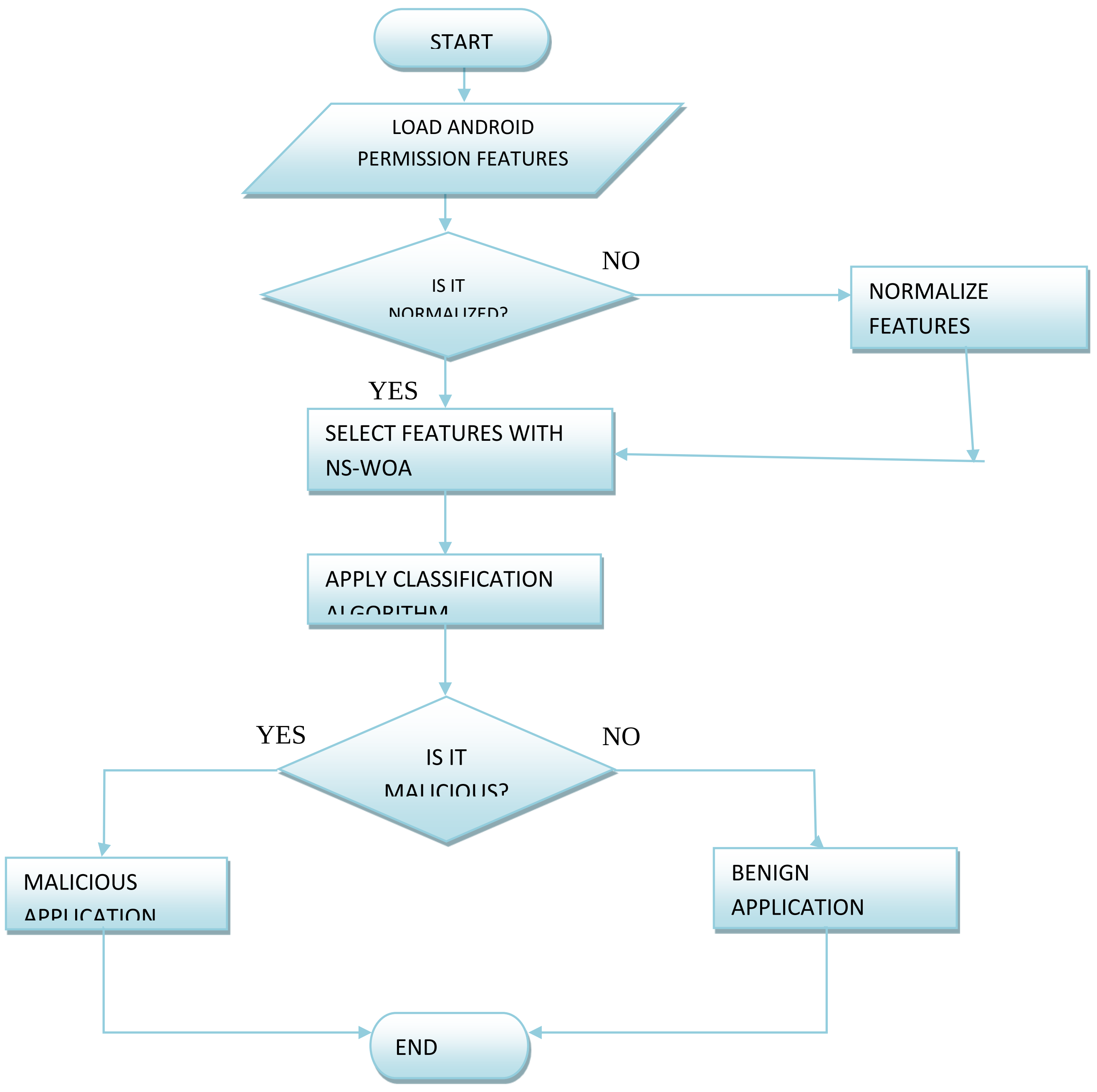


Figure 3.5: Flowchart of the system

3.5 Experimental Setup

The Waikato Environment for Knowledge Analysis (WEKA) toolbox was used for machine learning applications during the classification phase, whereas MatLab R2012a was employed for attribute selection. The selected feature using an improved algorithm was trained using the aforementioned classification algorithms in this tool.

3.6 Performance Metrics

Statistical test were used to measure the performance of the research model. The metrics like true positive rate, false positive, accuracy were used.

TP (True positive) was described as the android malicious application that was actually classified as malware i.e. **TPR** is the proportion of positive instances classified correctly.

TN: A benign android application is one that has been classified as such. TNR stands for the proportion of correctly classified negative situations.

FP: FPR is the proportion of negative instances labelled incorrectly as positive in a non-malicious android application, i.e. (malware).

FN: FNR is the proportion of positive instances incorrectly categorised as negative in a malicious android application that was classified as benign (non-malicious android application)

Therefore:

The fraction of correctly identified cases is measured by the accuracy (attributes)

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

$$TPR = \frac{TP}{TP+FN} \tag{3.2}$$

FP: FPR is the proportion of negative instances labelled incorrectly as positive in a non-malicious android application, i.e. (malware).

$$FPR = \frac{FP}{FP+TN} \tag{3.3}$$

$$FNR = \frac{TN}{TN+FN} \tag{3.4}$$

3.7 Evaluation of detection model

This evaluates the detection model using performance metrics in 3.1, 3.2 and 3.3.

3.7.1 Performance metrics

The accuracy of detection model was done using the metrics: Accuracy, FP Rate and TP Rate.

3.7.1.1 Accuracy

The percentage of the dataset properly categorised by an algorithm is used to determine its accuracy. It looks at positives or negatives on a case-by-case basis, therefore other performance indicators were used in addition to accuracy.

$$A = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \tag{3.5}$$

Where

FP = False Positive, TP = True Positive, FN = False Negative and TN = True Negative.

Negative and Positive represents the detection assumptions; true and false signify the detection expectations.

CHAPTER FOUR

4.0 RESULTS AND DISCUSSION

The chapter discusses experiment results and performance assessment of the model. The experiment was carried out using classification algorithm and Negative Selection Algorithm with whale Optimization Algorithm for selection of attributes.

Results

The results of the model obtained from classification algorithm with NSA are presented in Table 4.1:

Table 4.1: Detection Result After Applying NSA

Models	TPR	FPR	ACC
J48-NSA	0.714286	0.204545	0.82381
NB-NSA	0.847059	0.125	0.911905
NN-NSA	0.920354	0.052941	0.955952
RF-NSA	0.953623	0.030405	0.97483

In table 4.1 the first column represent models, the second column represent the result of True Positive Rate (TPR) from the models, the third column represent the result of False Positive Rate (FPR) from the models and the last column represent the result of Accuracy (ACC) from the models. It shows the result of the classification model when the classification algorithms are applied on the data without any selection of the data with Negative selection Algorithm and whale optimization algorithm (NSA- WOA)

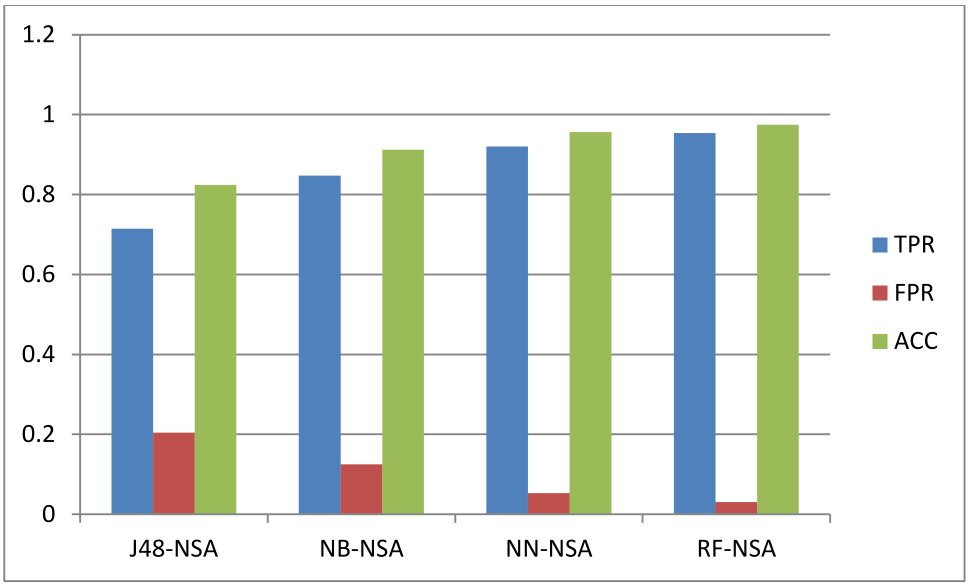


Figure 4.1: Detection without NS-WOA

The graph in Figure 4.1 represent the result in Table 4.1

Table 4.2: Detection Results after Applying NS-WOA

Models	TPR	FPR	ACC (%)
J48-NSA-WOA	0.771429	0.165138	91.2
NB-NSA-WOA	0.877358	0.068807	95.6
NN-NSA-WOA	0.88	0.035047	97.8
RF-NSA-WOA	0.936759	0.022267	98.7

Table 4.2 present detection result of the models each row presents the result of TPR, FPR and ACC. of each model. It shows the results of the models after the selection of data using Negative selection with whale optimization algorithm (NS-WOA).

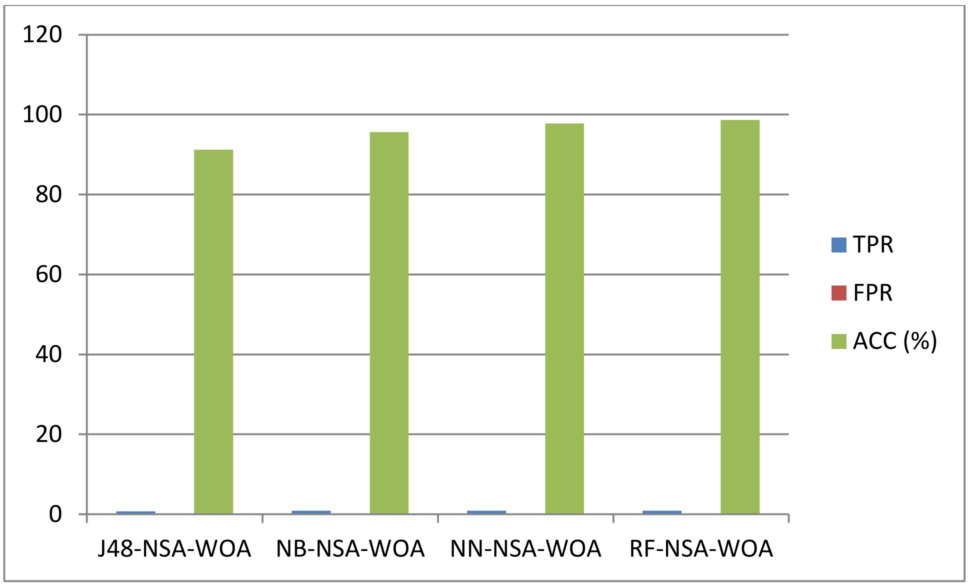


Figure 4.2: graph showing detection result after applying NSA-WOA

The graph in Figure 4.2 represents the result in Table 4.2.

Table 4.3: compering of results of other models with classification algorithm and NS-WOA detection model

Models	TPR	FPR	ACC (%)
J48-NSA-WOA	98.25	0.0192	98.17
PSO	96.0	0.0490	95.80
RF-WOA	0.8883	0.1332	97.8
RF-NSA-WOA	0.936759	0.022267	98.7

Table 4.3 shows the result of comparison with other models. The result show on the table that the model (J48-NSA-WOA) obtains TPR of 98.25, FPR of 0.0192 and ACC of 98.17. PSO obtains TPR of 96.0, FPR of 0.0490 and ACC of 95.80. RF-WOA obtains TPR of 0.8883, FPR o.1332 and ACC 97.8. RF-NSA-WOA obtain TPR of 0.936759, FPR of 0.022267 and ACC of 98.70.

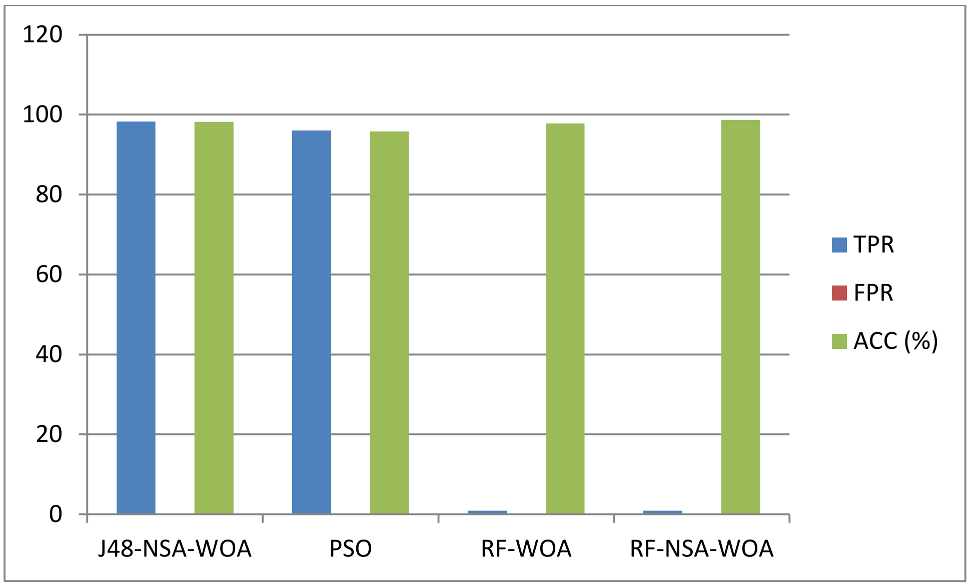


Figure 4.3: graph obtain from comparison with other models

The graph in Figure 4.3 represent the result in Table 4.3.

On the three tables TPR is the rate at which the classifier (NS- WOA) is detecting correctly the malicious application as malicious while FPR is the rate at which the model (NS-WOA) is classifying non malicious application as malicious. The accuracy of an algorithm is the rate at which the model is classifying correctly, the malicious and non-malicious application. It evaluates the percentage of the dataset correctly classified or detected by the algorithm.

Results Discussion

The three (3) performance metric used to measured and determine the effectiveness of the models are False Positive Rate (FPR), Accuracy (ACC) and True Positive Rate . The result shows that random forest with NS-WOA has the best results in term accuracy of 98.7%with least false positive rate of 0.22267 Figure 4.2 clearly shows Random forest with Negative selection with whale optimization algorithm to have best accuracy and lowest false alarm rate. The result also shows that other algorithm with NS-WOA has selection technique perform better with better Accuracy percentage and with least False Positive Rate compare to the ones without NS-WOA. In addition, Figure4.1 shows that the False Positive Rate of detection models without NS-WOA is negligible, that is the false alarm rate is relatively low compared to the results in Table 4.1. Figure 4.2 shows that the accuracy of classification models with NS-WOA is relatively high compared to the results in Table 4.1. Table 4.3 compare the results of some models with the NS- WOA model to ensure the confidentiality, integrity of the model which give the advantage to the availability of the usage of android application.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATIONS.

5.1 Conclusion

This research of android malware detection model with Negative selection algorithm and whale optimization algorithm (NS-WOA) selects the best features for the detection model. This is to reduce the features redundancy and duplication. The results show that the model with NS-WOA as selector is better than the one without selector in terms of accuracy and false alarm rate. This research is limited to only the permission features of the android applications. It is therefore, sufficient to say that the application of NS-WOA to the selection of attributes for data classification usually produce good and more accurate model than the otherwise.

This study ensures selection of better feature for the development of detection model. NSA-WOA with classification algorithm produces better accuracy of the detection model. The better model increases the confidentiality, integrity of the android application and has the advantage of the availability of the usage of android application.

5.2 Recommendations

1. The quantity of collected dataset can be increased to improve the accuracy of the results.
2. Negative selection Algorithm with whale optimization algorithm should be made available in various classification tools at an affordable rate.

5.3 Contributions to knowledge

i. The development of an improved android malware detection model.

ii. This research achieved an Improvement in detecting malware with a result of 98.7% performance accuracy.

REFERENCE

Abhijit, B. H. Xin, Kang, G. S., & P. Taejoon, "Behavioral detection of Malware on Mobile Handsets," June 17–20, 2008, Breckenridge, Colorado, USA. ACM 978-1- 60558-139-2/08/06, 2008.

Abid, A., Khan, M. T., Haq, I. U., Anwar, S., & Iqbal, J. (2020). An Improved Negative Selection Algorithm-Based Fault Detection Method. *IETE Journal of Research*, 1–12. https://doi.org/10.1080/03772063.2020.1768158

Adebayo, O. S., & Abdul Aziz, N. (2019). Improved Malware Detection Model with Apriori Association Rule and Particle Swarm Optimization. *Security and Communication Networks*, *2019*, 1–13. https://doi.org/10.1155/2019/2850932q

Adebayo, O. S., & AbdulAziz, N. (2014). Techniques For Analysing Android Malware. *International Conference on Information and Communication Technology For The Muslims World (ICT4M) 2014*, 1–6.

Agrawal, R., & Srikant, R., "Fast algorithms for mining association rules," In Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215, pp. 487-499, September 1994.

Alqahtani, E. J., Zagrouba, R., & Almuhaideb, A. (2019). A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms. *2019 Sixth International Conference on Software Defined Systems (SDS)*, 110–117. https://doi.org/10.1109/SDS.2019.8768729

Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, Yael Weiss "Andromaly": a behavioral malware detection framework for android devices". Journal of Intelligent Information Systems 38(1) (January 2011) 161{190}, 2011.

Biau, G.(2012). *Analysis of a Random Forests Model*. The journal of machine learning research 13 (2012): 1063-1095.

Brown, J., Anwar, M., & Dozier, G. (2017). An artificial immunity approach to malware detection in a mobile platform. *EURASIP Journal on Information Security*, *2017*(1). `https://doi.org/10.1186/s13635-017-0059-2

Castro, L. N. de, & Timmis, J. I. (2003). Artificial immune systems as a novel soft computing paradigm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, *7*(8), 526–544. https://doi.org/10.1007/s00500-002-0237-z

Chen, H., Xu, Y., Wang, M., & Zhao, X. (2019). A balanced whale optimization algorithm for constrained engineering design problems. *Applied Mathematical Modelling*, *71*, 45–59. https://doi.org/10.1016/j.apm.2019.02.004

Christodorescu, M., Jha, S., Seshia, S. a., Song, D., & Bryant, R. E. (2005). Semantics-Aware Malware Detection. 2005 IEEE Symposium on Security and Privacy (S&P'05), 32–46. https://doi.org/10.1109/SP.2005.20

Dasgupta, Dipankar, Yu, S., & Nino, F. (2011). Recent Advances in Artificial Immune Systems: Models and Applications. Applied Soft Computing, 11(2), 1574–1587. https://doi.org/10.1016/j.asoc.2010.08.024

Department of Computer Science, Virtual University of Pakistan, & Tahir, R. (2018). A Study on Malware and Malware Detection Techniques. *International Journal of Education and Management Engineering*, *8*(2), 20–30. https://doi.org/10.5815/ijeme.2018.02.03

Dewanje, A., & Kumar, K. A. (2020). *A New Malware Detection Model using Emerging Machine Learning Algorithms*. I.J. of Electronics and Information Engineering, Vol.13, No.1, PP.24-32, Mar. 2020. http://.org/ 10.6636/ijeie.2021.03 13(1).04

Idris, I. (2012). Model and Algorithm in Artificial Immune System for Spam Detection. *International Journal of Artificial Intelligence & Applications*, *3*(1), 83–94. https://doi.org/10.5121/ijaia.2012.3107

Kinholkar, M. B. P. (2015). *Study of Malware Detection Technique for Apk and SDK File Using Artificial Immune*. *8*, 5.

Mariot, L., & Leporati, A. (2015). Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications. *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15*, 1425–1426. https://doi.org/10.1145/2739482.2764674

Milosevic, N., Dehghantanha, A., & Choo, K. R. (2017). Machine learning aided Android malware classification R. *Computers and Electrical Engineering*, *0*, 1–9. https://doi.org/10.1016/j.compeleceng.2017.02.013

Mostafa Bozorgi, S., & Yazdani, S. (2019). IWOA: An improved whale optimization algorithm for optimization problems. *Journal of Computational Design and Engineering*, *6*(3), 243–259. https://doi.org/10.1016/j.jcde.2019.02.002

Rana, N., Latiff, M. S. A., Abdulhamid, S. M., & Chiroma, H. (2020). Whale optimization algorithm: A systematic review of contemporary applications, modifications and developments. *Neural Computing and Applications*, *32*(20), 16245–16277. https://doi.org/10.1007/s00521-020-04849-z

Siddiqui, M. A. "Data Mining Methods for Malware Detection," A dissertation submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Modeling and Simulation in the College of Sciences at the University of Central Florida, Orlando, Florida, 2008.

Thomas Eder, Michael Rodler, Dieter Vymazal, Markus Zeilinger "A Framework For Analyzing Android Applications". Workshop on Emerging Cyberthreats and Countermeasures ECTCM 2013.

VirusTotal. (2015). *Free Online Virus, Malware online Scanner*.. https://www.virustotal.com/en//.2015

Yang, C., Jia, L., Chen, B.-Q., & Wen, H.-Y. (2020). Negative Selection Algorithm Based on Antigen Density Clustering. *IEEE Access*, *8*, 44967–44975. https://doi.org/10.1109/ACCESS.2020.2976875