ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

School of Electrical and Computer Engineering

Graduate Programs

Neural Network Based Malware and Suspicious Activities Detection Based on User Activity

By

Tegegn Kebebaw

Supervised by

Getachew Alemu (Ph.D.)

December 2018

Addis Ababa, Ethiopia

Neural Network Based Malware and Suspicious Activities Detection Based on User Activity

Tegegn Kebebaw

A Thesis

Submitted to School of Electrical and Computer Engineering, Addis Ababa Institute of Technology, Addis Ababa University in partial fulfillment of the requirements for the Degree of Masters of Science in Computer Engineering

Addis Ababa

Ethiopia

Declaration

I declare that this piece of work is my own and all sources of materials used for this thesis work have been accordingly acknowledged. The thesis has been submitted in partial fulfillment of the requirements for the degree of Master of Science at Addis Ababa University and is reserved at the university library to be made available to users. I truly declare that this thesis work is not submitted to any other institution anywhere for the award of any academic degree, diploma, or certificate. With accurate acknowledgment of the source, users are free to use this thesis without special permission.

Name: Tegegn Kebebaw Signature: _____ Date: _____

As a thesis research advisor, I hereby approve that I have read and evaluated this thesis prepared, under my guidance, by tegegn Kebebaw, entitled "Neural Network Based Malware and Suspicious Activities Detection Based on User Activity". I recommend it be submitted as fulfilling the thesis requirement.

Getachew Alemu	_____	_____
Name of advisor	Signature	Date

Approval sheet

Tegegn Kebebaw_____

Student` s Name Signature Date

The following examiners and faculty members certify that this student has successfully presented the necessary written and oral presentation of this thesis for partial fulfillment of the thesis-option requirements for the Degree of Master of Science in computer engineering

Approval of Board of Examining:

_____ _____ _____

Name of Chairperson Signature Date

_____ _____ _____

Name of Advisor Signature Date

_____ _____ _____

Name of Internal Examiner Signature Date

_____ _____ _____

Name of External Examiner Signature Date

Acknowledgments

I would like to give my acknowledgement to my Family who was been with me supporting in any way they can during this research work, this research wouldn't be where it is without the help of my colleagues, I like to name those who helped me a lot in many ways, Mr. Mohamed, Mr. Donyam and Mr. Haymanot. Thank you for letting me conduct my experiment in your personal computers, I really appreciate what you did.

Mr. Ezana, Mr. Tesfaye, Mr. Tesfahuneegn and my dearest brother Mr. Girma I never forget your assistance when I need it most, thank you all.

Finally, I would like to thank my Advisor Doctor Getachew Alemu who was been with me since the beginning guiding me through the whole process, and Doctor Srinivas for the positive response you gave when I need it most.

Abstract

Malware is a software program specially designed to disrupt, damage, or gain unauthorized access to a computer system. The number and types of malwares are increasing in alarming rate since the first time the word malware or virus has been heard. Currently there are more than half a million malwares produced in every day. In order to protect computers from such attack different methodologies have been proposed and implemented. However, none of the solutions are able to give absolute cure for malwares. In this work, another method to protect personal computers from any kind of malware by learning user activity is presented. The premise of the proposed solution is based on 'human activity is repetitive behavior in nature'. So, this repetitive behavior can be used to differentiate normal application programs the user usually uses in its computer and new arriving programs or intruders. There is a newly developed dataset collection mechanism presented in this paper which will help to record the user activity of the user while using its computer. Using the collected user activity or datasets the neural network algorithm trained. After training, each computer will have its own neural network model. During testing, dataset collector program will capture a running program and evaluate using the neural network model trained by the dataset collected from the same computer. Based on the evaluation output, the specific running program will be categorized as malware or normal application process. In this research, the experiment has conducted in three different computers in order to understand if this approach will help to relate human activity with the program they used and the resource consumed during execution. From the experiment conducted based on ten days of knowledge (dataset), the proposed approach able to predict whether the running process is malware or normal application with 82%, 75% and 69% of accuracy for three different experiments. From this experiment, the proposed approach can be applied to detect malware programs.

Contents

List of Figures

List of Tables

List of Equations

List of Acronyms

PID: Process identity number

DDOS: Distributed Denial of Service

NN: Neural Network

RNN: recursive Neural Network

DBN: Deep Belief Network

FAP: File Access Pattern

FAP: File Access Pattern

FPR: False Positive Result

CPU: Central processing Unit

TC2: Test Computer 2

FP: False positive

FN: False Negative

TP: True Positive

TN: True Negative

Key words:

Malware, Neural Network, User Activity

Chapter one: Introduction

This chapter will discuss the back ground of malwares, and anti-malwares, introductions about different malware families, how malwares could be differentiated from normal application programs, the objective of the research, significance of the study and problem statement are included in this chapter.

1.1 Background

Computer viruses have been threats to a computing device since 1949 fred cohen [east 2018 #26], the first man to use the word virus and make a research for his PH.D. thesis in 1971 [plc. 2017 #24]. He defines the word virus for the very first time as “a program that can 'infect' other programs by modifying them to include a possibly evolved copy of itself”. With the infection property, a virus can spread throughout a computer system or network using the authorizations of every user using it to infect their programs. Every program that gets infected may also act as a virus and the infection rate grows rapidly.”[plc. 2017 #24]

Since this[1] day the computing world started to worry and feel not secure to use computers. As a response to protect computers from such malware, different programmers and researchers have been suggesting different ways to protect the computing world from malware programs. But at the time the number of computers is very small and most of them are main frame computers. That means they can’t be attacked easily since the access is limited. When the number of computers increases from time to time the number and types of malwares are also increasing gradually. The first anti malware or anti-virus program was introduced in 1981 by ray Tomlinson and known as “The Reaper” which is a virus program by itself [inc. 2018 #29], but written to remove the first ever documented virus known as “creeper virus”. After the first anti malware introduced, different antimalware programs have been produced to protect the cyber world from a lot of malwares produced every day.

Different approaches have been used to remove malware from the computer. The first method used to write an antimalware program is to write another malware program which hunts down and kill the specific program (virus). During this era, different antimalware software was introduced, but they are not efficient to detect every malware. Then the second approach began to record the file signature of malwares and to remove by checking their signature. This one is more general

[1] The day malware programs introduced | 1949

protection mechanism, if the malware signature is once detected it will be saved in the signature database. And if a program is running the signature will be compared to the list of signatures in the database. If it has similar signature to one of the lists in the malware signature database, it can be suspected as malware program and removed from the computer [company, 2018 #29].

The third approach is to use code structuring and analysis. This era is the recent innovation introduced in the antimalware industry and it is more efficient than the previous methods. There are many kinds of research performed to this specific area by varying the algorithm and parameters used in order to analyze the code. They are also efficient in many ways, but they are not able to secure the computing world from malware threat absolutely. The recent research conducted by the AV-TEST Institute shows, the number of malwares released in every month is around 11.5 million [institute , 2018 #27]. Figure 1 shows the number of malware programs registered to AV-Test Institute database.

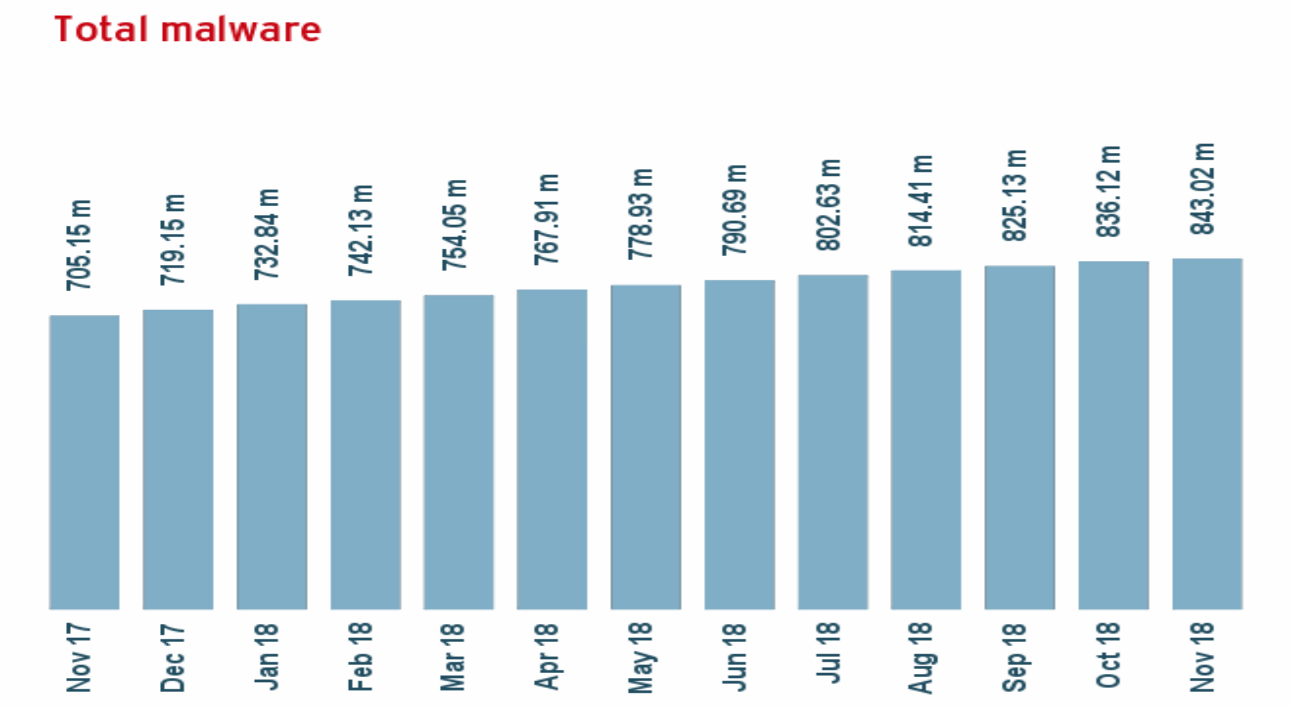

Figure 1: The total number of malwares registered [institute , 2018 #27]

The diagram in Figure 1 shows the total number of malwares registered for one year. The malwares may be new or already existing ones. The diagram shows the number of malware is increasing every month in millions. Figure 2 presents only the number of new malwares discovered each month for one year.

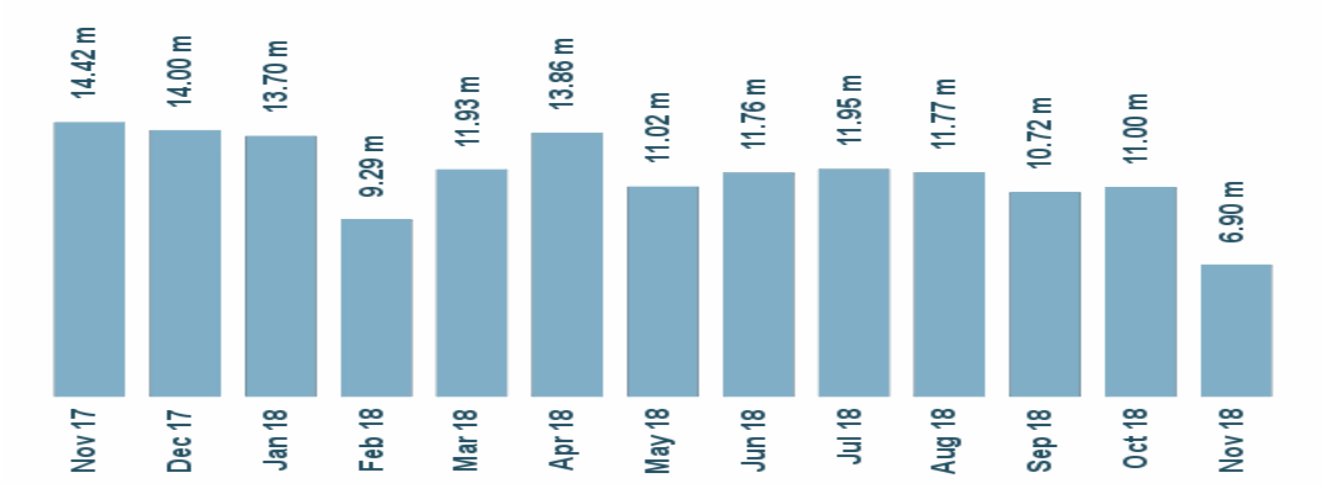

Figure 2: Number of new malwares produced in a day [institute , 2018 #27]

Figure 2 shows the number of malwares produced every month is counted to be in millions. The number of new introduce malwares make very hard to detect or categorize malwares into specific groups. With this number of malwares being produced every month the approaches that used today will never good enough to protect the computing world from all kind of malware. So, it needs to have a permanent solution which may not require analyzing individual malwares signature or code structuring.

1.2 Statement of the Problem

In every day when we open our computer and start to work on, most of the users think what if our computer is infected by a virus? Do any one spying on us? What makes my computer so busy? And other questions that make us uncertain about the computer we use. The major reason what makes us think like this is we have no idea how many processes are running inside our computer, or we may not able to differentiate the regular normal programs from the abnormal ones. So, we will try our best to protect our devices from virus and malwares. To do so we may use different approaches, some of us may isolate our computer from the network and other external devices like flash. Some people use antivirus, the first one looks good but it can't let us do our job well. That will leave us with one choice to use antiviruses. That will leave us to wonder how this antimalware programs actually work. Antivirus use different approach to detecting malwares. The most common approach anti-malwares use to detect malware is code analysis or signature-based detection method.

The code analysis method is the most recent trend practiced in antimalware industries. This approach analysis the code structure of every malware and create a data base of malware programs code structure. During scanning the algorithm will compare the structure of the scanned programs with list of malware code structure in the database. If it feats, then the program will be categorized as a malware and removed from the computer permanently. But this approach consumes a large amount of user time and computer resource. Most of all, it is almost impossible to have all malwares structure in one place, especially the ones that are not discovered yet.

The other approach that anti-malwares use is signature-based identification. This approach will have ideally all file signatures of malware programs in the database. When the anti-malware scans for malware it will check the file signature of all files inside the computer. If the signature of any program is similar to the signatures in the database list, than that program will identified as malware. And the file will be removed from that computer permanently. But the database of the anti-malware installed in that computer will not contain all malware structure all the time. So, it will protect the computer only from the malwares that this anti malware knows. And it needs to update the database daily.

Generally, no anti-malware approach that is being used today is able to protect us completely. The reason is currently available Antiviruses don't know all malwares, viruses and other suspicious activities running in our computer. They deal only with what they already know that is the major gap in the currently existing anti-malware technology.

1.3 Objective

The objective of this research is to develop a malware and suspicious activity detection system that protect personal computers from any kind of malware and suspicious activities, by studying user activity.

1.3.1 Specific objective

- To gather the requirement for the program parameters[2]
- To develop a dataset collection mechanism which captures all user activity [3]
- To design a dynamically learning algorithm that identifies the feature of applications inside the computer system.[4]
- To extract features of applications that will be used for network training.[5]
- Collect dataset for training.
- To apply the trained neural network model to classify the process as a normal or suspicious process

1.4 Scope of the research

The scope of this research is to study the malware protection methods, characteristics of malwares, currently used methodologies in anti-malware technology, the problem with the currently applied methodology, proposing a new methodology, design to a new methodology, implementation of dataset collection mechanism, implementation of neural network algorithm to be used, experiment using different computer and recorded the output of the experiment. Conclusion about the result and comparison of this research work with the previous works in the same area is also the scope of this research work.

[2] The requirements to define what parameters of process will help us to study the user activity
[3] Since the dataset is required for each user separately, the dataset collection mechanism need to be Automated and should be real time.
[4] Flow chart presented in the methodology section
[5] Filtering mechanism

1.5 Significance of the Study

The study of this research will be significant for all personal computer users and antimalware industries. For personal computer users, it will protect their computer from any kind of attack based on their daily activity by learning how they usually use their computer and if something suspicious activity observed the program will alarm user and take action. And it doesn't need internet connection to update the database or to analyzing each and every program object code, which consume a large amount of computer resource and user time. For antimalware industries it will introduce new methods of building antimalware systems for personal computers and large servers which consume a small amount of resource and time.

Chapter Two: Literature Review and Related Work

This chapter will introduce the basics about malware and other basic concepts that have been used in the rest of the research, different paper review presented and the summary of the research works conducted and how the proposed research objective is different from the previously conducted research has been explained well under this chapter.

2.1 Basics of malware

Malware is defined as an unauthorized program which is running inside our computer system and use all the computer resource without the knowledge of the owner [Inc., 2018 #31]. This malware can steal our resource, corrupt our data and system. They can also make our system busy so that we cannot use our own system effectively. They can cause unpredictable and large damage to computer hardware and other resources inside the computer. There are different kinds of malware, according to recent research conducted by the organization called 'Vera code' the following are listed as common types of malware [Zeltser 2018 #28]:

- adware,
- bots,
- bugs,
- root kits,
- spy ware,
- Trojan horses,
- viruses, and
- Worms.

They are classified based on how they behave and used inside the computer system.

Adware: Adware is a kind of malware that pops up in your window while you are using it. Especially if you are connected with the internet there will be different ads in your browser, most of them are sponsored by the advertisement companies to introduce their organization or product. But some ads will miss use the basic objective, they will record the activity of the user, they even record any password and user name, visited address. Basically, they will record every activity of the user. [Zeltser 2018 #28]

Bots: Bot malware is usually known as boot net. A bot net is a network of infected computers that can be controlled remotely, forcing them to send spam, to spread viruses or to participate in DDOS attack without the consent of the owner. This is the most dangerous malware that hackers use to attack supercomputers. [inc. 2018 #29]

Bugs: Bugs are one of the malware categories that are being used by every hacker; they are simple and easy programs that carry infection with them to infect every computer or files they contacted. But they need some kind of traveling mechanism to spread from computer to computer once they get in to the computer, they will infect every file and program inside that computer. [inc. 2018 #29]

Root Kits: These categories of malware are the world's most dangerous malware. Root kits are programs that get access to your computer without your knowledge and they can do everything they need. Assume if there are many administrator users in your computer and you are the owner, but you can't control them, they may even kick you out if they are using the computer resource in full capacity, it's very embarrassing kind of malware, you don't even know their existence, since it able to hide itself at the lower layer of the operating system. [inc. 2018 #29]

Spyware: Spyware is the hardest kind of malware to catch. Spywares programs record data like the browsing history, credit card number, or passwords and it will pass this sort of information to the third party. This kind of malware is very hard to even know you are a victim, since they are not able to use many resources and make your computer busy. So they have a good chance they may remain undetected for a long time. [MEtcalf 2018 #25]

Trojan Horses: These kinds of malwares hide themselves inside the normal application we usually use, or inside the email content. They can make any kind of destruction in our computer depending on how they are programmed; they travel from place to place by portable device and internet. [Zeltser 2018 #28]

Viruses: They are distractive agents, and their objective is to destroy everything they can. Once they are inside a computer system they can duplicate themselves in any environment and destroy the computer file, program or even hardware itself. They are a very dangerous program now a day. [Pagliery. 2018 #29]

Malware: Malware is a short term for malicious software. It is software that is specifically designed to disrupt, damage, or gain unauthorized access to a computer system. there are many kinds of malwares that exist today some of them are spyware, phish net and browser hijackers [institute , 2018 #27]

2.1.1 How to Protect Computers from malware

The first and the best way to secure our computer 100% from any attack is to isolate the computer we want to be secured from the rest of the world, that means there is no internet, no flash, or CD, or anything that carry data and plugged to the isolated computer. Isolating computer will ensure the complete security of the computer from any kinds of attacks. But it is almost impossible to use this approach since the basic purpose we use a computer is to process data and interact with other computer users through internet, or to exchange files through a flash drive or other devices. It is difficult to practice this approach for a long time [Bishop, 2003 #16; Perlman, 2016 #15].

Another method is to use different kinds of anti-malware software. This[6] is software which installed inside our computer and protects our system from malwares. This approach will work well if we able to use the original full package and update the anti-malware regularly. This method is more like general prevention. It includes updating the fire wall and operating systems with possible security packages regularly. [Bishop, 2003 #16]

The last mechanism to protect the computer from attack is to be careful while we use the computer, especially when you are online don't click everything, don't open your email without reading the subject first and make sure they are from trusted sites. Also try to understand the process running inside your computer [Perlman, 2016 #15].

2.2 Properties of malwares

Malwares are programs, and they have all the properties of programs. But malwares have some distinguished property than other regular application programs. This research interested in the

[6] Anti-malware Software

property of malware from the point of resource usage inside our computer while they are running. Some of the property is explained here. [Bishop 2003 #5]

- Malware programs run inside our computer without the user (owner) knowledge. They are an unauthorized program.
- They consume a very large amount of resource if their objective is a denial of service.
- They corrupt our data. During this process they may use a very large amount of CPU and Memory.
- They can spy inside our computer; in this case, the malware programs don't use many resources as the matter of fact they consume an almost negligible amount of resource on average
- They run inside our computer for a long time, normal programs can be used frequently, but not for long time because the user will stop the service when they done with it. But malwares they run until our computer shutdown.

Generally, from the point of resource usage, most malware programs consume either a large amount of computer resource or an almost negligible amount.

2.3 malware Identification Methodologies

Most of the anti-virus programs that exist today use four techniques to classify programs as suspicious malwares or normal program [Bishop 2003, de Almeida 2016, Zeltser 2018].

✔ ***Signature-based detection***

✔ ***Heuristics-based detection***

✔ ***Behavioral detection***

✔ ***Cloud-based detection***

2.3.1 Signature Based Detection

This technique is the most common technique used by most anti-virus programs. The basic work flow in this technique is, collect the signature of malwares as much as possible and save it to the database. While scanning the computer try to compare each program's signature with the list of signatures in the database. If the running program signature matches with one of the signature in the database, than that program will be detected as malware and action will be taken. This

approach is very efficient and effective only if we have all lists of signatures for malwares. But we are not able to know the signature of malware that is going to be developed for future, so we don't have all list of malwares signature. This program may protect as from the previously identified malware but not from future malware. Statistic shows nearly half million malwares will be released every day, [institute , 2018 #27] having this information it is very hard to protect our computer using this technique. [Bishop 2003 #16]

2.3.2 Heuristics-based detection

Heuristic based detection is another technique to detect 'malware programs by generically detecting new malware by statically examining files for suspicious characteristics without an exact signature match' [Kang and Kang 2016 #10, Perlman, Kaufman et al. 2016 #15] in this approach the anti-malware program analyze the programs object code and try to find if there is any junky code, or instructions that may create damage to the computer resource. If the analyzed files include any suspicious[7] command or instruction, it will be identified as malware and action will be taken by the user or by anti-malware itself. This approach will solve the problem of storing the signature of every malware in data base it needs just some list of instruction, which may cause damage in the computer or in a file. But it is not accurate, many malware will be identified as normal programs and normal programs will be suspicious according to this technique approach. And this technique will consume a lot of time and resource to evaluate each and every program's source code to search for listed instruction. So, it is not efficient and accurate.

2.3.3 Behavioral Detection

Behavioral detection technique observes how the program executes, rather than merely emulating its execution. This approach attempts to identify malware by looking for suspicious behaviors, such as unpacking of program code, modifying the hosts file or observing keystrokes. Noticing such actions allows an anti-virus tool to detect the presence of previously unseen malware on the protected system. This technique performs better but, it may not able to identify the malwares easily. That means it is not able to identify all malware by itself it needs to add another mechanism to ensure the security. [Bishop 2003 #16, company 2018 #29]

[7] Suspicious is defined by the content of the database that analyzes the code.

2.3.4 Cloud Based Detection

This approach is the most recent technique to detect malwares. There would be a secure computer which accepts requests, evaluate and return the result. And there will be other client computers which are running virus infected programs. When the client suspect the running program is infected program, it will send the suspected program for cloud computing. The computing machines retrieve information from secure computers and evaluate the suspected program. Decision will be sent back to the requester. By doing such computation the anti-malware may able to detect malwares, but there are some problems with this technique too, it needs to be connected always, and what if the secure computer is not actually secured, there is no way to be sure. [inc. 2018, institute , 2018 #27; Pagliery., 2018 #30]

2.4 Anti-malware programs

Anti-malware pogrom is a program that is basically designed to search and destroy a specific malware they are hunting. But from time to time the technology of protecting computers from malwares are also increasing with the number of malware numbers that are increasing in alarming rate, the first anti-malware is similar to the malware itself. it will run inside your computer without your knowledge and search the specific malware and destroy it. After that, they try to modernize their way since the number of malwares is increasing it's not possible to find each malware by its name. so, they apply signature-based method, which is the anti-malware will keep the list of malware programs signature and when they see one, they will stop the program from executing and they might destroy it from that machine. [de Almeida 2016 #11, company 2018 #29, east 2018 #26]

The anti-malware industry is not that much as related to the number of malwares produced every day, there is an unlimited number of malwares until today, and most of them use this signature-based detection mechanism [company, 2018 #29]. The currently arising antimalware started to use the code structure analysis method to detect malwares. In this approach the code of every malware programs have similar characteristics. They try to categories the structure of the malware in different families. When the anti-malware scans, it doesn't just check the signature, it also studies the structure the running program code. If the structure has similarity to any structure inside its database, the program will be considered as a malware and it will be removed from that computer. [inc. 2018 #12]

The most recent invention that is being researched a lot in this area is to apply the artificial neural network concept to produce the anti-malware system. This area has been conducted a lot of research, the objective of this research paper is also related to this concept directly. So once again let's have some basic concept about artificial neural network concept.

2.5 Artificial Neural Network Algorithms

The artificial neural network is the machine learning process to protect computers from different human made threats. This neural network tries to simulate the human being brain system. The idea is if we able to teach machines or programs as we do in human being since we are a kid, the machines also get the ability to process and identify patterns by themselves. Neural network uses a different number of layers and nodes depending on the data set characteristics they are going to train. Basically, the neural network requires at least two-layer, input layer, and an output layer. And most of the times we need to add at least one middle layer between the input layer and the output layer and it is called the hidden layer. [Peretto 1992, Aminanto and Kim 2016, Tobiyama, Yamaguchi et al. 2016, Wang and Yiu 2016]

Usually, the neural network is modeled as follows.

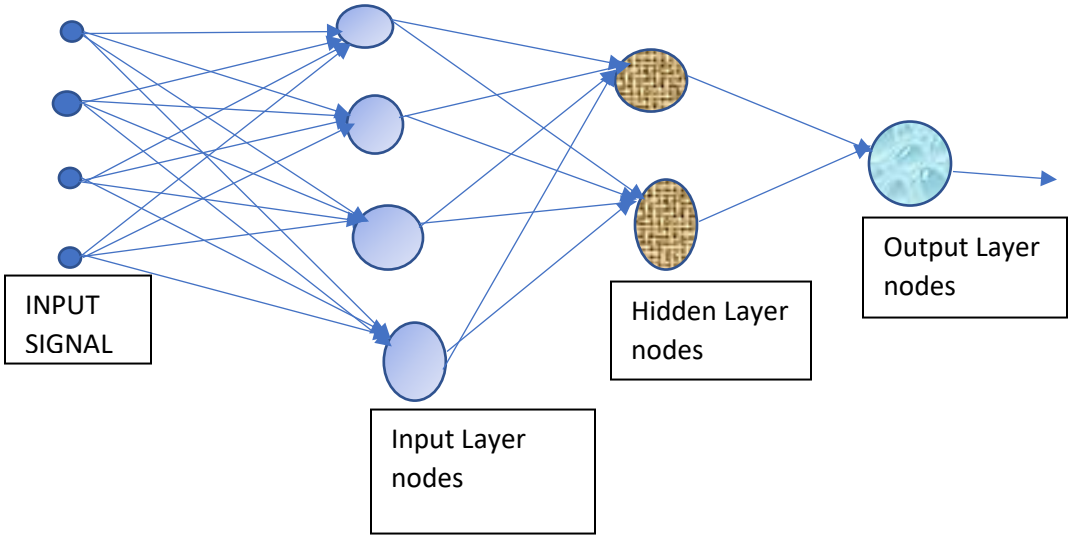

Figure 3: a typical neural network model

Input: is the values you want to train using the given neural network model, this input is usually numerical value or pictures in the form of Matrix table or wave signals. These inputs each of them represent something that makes sense to the neural network model. Layer: is the first layer of the network, it contains a different number of nodes; each node accepts every input given from the input side and performs an operation usually using sigmoid activation function. [Haykin, 2009 #22]

The sigmoid function is the most common method to evaluate the neural network output of each node. It requires the input signal, the weight, and the epsilon. Epsilon used to facilitate the network to converge easily (like the catalyst in a chemical reaction), it is the constant value chosen randomly. This value usually relies on between 0.5 and 0.8 the most common value is 0.65. [Gurney, 2014 #19]

The output of each node is computed using the following formula.

$$f(x) = {}^{1}/_{(1 + e^{(-x)})} \qquad \textit{Equation 1}$$

This formula is the common formula used to evaluate the output of each node including the output layer value. But it is possible to use any other formula which may be suitable according to the problem you are dealing with.

In a neural network, the number of layers is dynamic it could be varied from problem to problem, but the input layer and the output layer is must. [Peretto, 1992 #21]

The common model of the neural network is 1 input layer 1 middle layer and 1 output layer. Each layer can contain a different number of nodes, but the output layer contains the number of output vector required. If it is classification the number of nodes in the output layer would be equal to the number of classes, we would like to classify them. If it is an evaluation, we would have one output node which gives us a single output value. The number of nodes on the input layer and in the output, the layer is completely depending on the developer choice. [Lippmann, 1987 #11]

Artificial network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections. [Haykin, 2009 #22

Artificial neural network model can be distinguished by the following characteristics.

- A set of the processing unit (neurons, cells)
- A state of activation Y_k for every unit, which equivalent to the output of the unit.

The total input to unit k is simply the weighted sum of the separate outputs from each of the connected units plus a bias or offset term $Ø_k$:

The process would be generalized as [Peretto 1992 #11]

$$s_k(t) = \sum_j w_{jk}(t)\, y_j(t) + \theta_k(t). \qquad \textit{Equation 2}$$

If propagation rule introduced it would be expressed as. [Aleksander and Morton 1990, Peretto 1992]

$$s_k(t) = \sum_j w_{jk}(t) \prod_m y_{j_m}(t) + \theta_k(t). \qquad \textit{Equation 3}$$

An activation function F_k which determines the new level of activation based on the effective input $S_k(t)$ and the current activation $Y_k(t)$ (i.e the update) [Gurney 2014 #13]

$$y_k(t+1) = \mathcal{F}_k(y_k(t), s_k(t)). \qquad \textit{Equation 4}$$

Often the activation function is a non-decreasing function of the total input of the unit [Peretto, 1992 #21].

$$y_k(t+1) = \mathcal{F}_k(s_k(t)) = \mathcal{F}_k\left(\sum_j w_{jk}(t)\, y_j(t) + \theta_k(t)\right), \qquad \textit{Equation 5}$$

The activation function is not restricted to non-decreasing function, for smoothly limiting function often a sigmoid function is used [Gurney, 2014 #19].

$$y_k = \mathcal{F}(s_k) = \frac{1}{1+e^{-s_k}} \qquad \textit{Equation 6}$$

- An external input $Ø_k$ for each unit
- The method for information gathering (the learning rule)
- Error signal

Another important concept in the artificial neural network is the topology of the network. This is simply the pattern of connection between the units and the propagation of data. For this connection, the main decision made is between feed-forward network and recurrent network. Most of the above-mentioned points will be available in every neural network model, but some of them may get ignored in different models, their effect may get negligible. [Lippmann 1987, Aleksander and Morton 1990, Peretto 1992]

2.6 Malware Detection with Deep Neural Network Using Process Behavior

The objective of this research is to develop a security system that protects the computer from any kind of attack with zero-day age, by analyzing the behavior of process using deep learning neural network approach. They applied deep learning neural network to extract the feature they need from the process and they apply Recursive neural network to classify the selected process as a normal or malignant process. From their experiment, they are able to get 0.8, 0.96, and 0.92 precision for the three-conditional experiment they conducted by varying the number of hidden layers they used during their experiment. [Tomiyama, 2016 #12]

In the experiment, they are trying to analyze the behavior of the process by extracting features using their API, the individual API call represents the operation of the process, multiple API calls represent the activity, and whole recorded API calls represent the behavior of the process, from this they are able to extract the following log information for RNN phase,

Time :(how long the process was running)

Process Name: (the command being executed)

PID: (unique process ID)

Event: name of the operation

Path: address of the file

Result: result of the operation

Detail: more information about the operation

To do this they record the process behavior in some time interval for N times and based on that they selected their features and apply RNN to classify weather the process is malware or not.

Critics: Normally when we talk about the process behavior it always depends on how the user uses it, so it is hard to decide the behavior of the process with different users or it's hard to generalize as average, since different users use the same program in many ways the selected parameters don't completely represent the total behavior of the process, the other thing is, there are many malwares that act as normal application programs and this program may not able to detect such kind of malwares. such kind of malwares are identified only by the user of the computer, so there are many parameters missed from the parameters that represent the process behavior presented in this paper. But based on the performed experiment the result acquired very satisfactorily.

2.7 A hybrid Malicious Code Detection Method based on Deep Learning

In this paper, they proposed hybrid malicious code detection auto encoder and deep belief network (DBN). They used auto encoder deep learning method to reduce the dimensionality of data and they apply deep belief network to detect malicious code. Based on the experiment they performed they achieved 95.34% of TPR and 9.02% of FPR with an accuracy of 91.4% the CPU usage of the experiment is 1.126 seconds using DBN and they achieved 96.79% of TPR, 15.79%FPR with accuracy of 89.75% and it takes 2.625 seconds by applying a hybrid system of the two algorithms [Li, Ma et al. 2015 #3]. As it has observed from the experimental result using a hybrid of the two methods may increase the TPR but its performance is reduced and there is also a large percent of FPR.

In this research, the problem is the method they used to analyze the binary code structure of the programs one by one and train the structure of the normal and malware programs. This approach is not effective all the time since the malwares always change their structure. And if large data set applied the accuracy may increase but it will take a large CPU time so its performance may become critical and become difficult to use it.

2.7 A multi-task Learning Model for malware classification with Useful File Access Pattern from API call sequence

The objective of this research paper is to build a model that classifies malware based on API call sequences, semantic aware and machine learning. This research, they apply RNN based Auto encoder to automatically learn a low dimensional representation of malware from its raw API call sequence. and they used multiple decoders under different supervision to give more information other than class and family label of malware [Wang and Yiu 2016 #4].

They make multi task malware learning model based on API call sequences; their model consists of two decoders. For malware classification and the other one for file access pattern (FAP) generation given the API call sequence of malware. They are able to classify the malware family based on seq2seq model for classification and FAP generation.

This research presented a novel idea for malware classification, but this paper presented only how able to classify the malware to their respective family class by analyzing the binary file of the programs, of course after they are identified as malware. Another problem with this the process may take a large amount of CPU time if we try to analyze each and every code of the programs that exist in our computer, so it is not efficiently in CPU time usage.

2.8 Deep Neural Network Based Malware Detection Using Two-Dimensional Binary Program Features

In this research they introduced a new approach to detect a malware that exists in a computer system by applying deep neural network using feed forward algorithm. It is old research most probably the first step to using machine learning approach for such application. they achieved 95% of True positive detection rate and 0.1% of false positive rate (Saxe and Berlin 2015 #9).

The problem with this research is the methodology they used. Their algorithm performs detection by analyzing the binary code of the program. That means the algorithm used will be trained ones and detect the malware based on the weight adjustment that is being made by the learning algorithm. It is not flexible and can't learn artificially and it takes a huge amount of memory and processor time. So it is not possible to deploy their algorithm in personal computers.

2.9 Neural Network Artificial Immune System for Malicious Code Detection

This research presents an intelligent adaptive self-learning and self-organizing system for malicious code detection based on the integration of artificial immune system and the artificial neural networks. To develop this system, they used a simple neural network with feed forward learning algorithm. According to the Authors, their system can categorize the application by extracting some features from the binary code of the program and it will classify them as malicious software or normal software. the system performs well according to the data they provided in their research paper (Haykin, Haykin et al. 2009 #22).

The problem observed in the approach is that: even if the system may recognize any kind of malware it is not dynamic. It can't learn from its mistake or new things from user or from itself. The other problem, it requires a large CPU time and storage location. Since it analyzes the binary file of the program to extract feature, this process by itself slow the performance of the computer and annoy the user.

2.10 A Multi Task Neural Network for Dynamic malware Classification

This research investigates the effect of the different parameter in the deep neural network to classify malware (malware Vs. benign). The classification method will be to use neural network algorithms by varying the number of Layers, activation function and the size of data set used to

train the algorithm. They used about 4.5 million datasets for training which is extracted dynamically from different files. And 2 million data set for testing. they achieve 0.358% of error rate during binary classification task and 2.94% error rate of a standard malware family classification architecture (Wang and Yiu 2016 #3).

From the experimental result achieved for 2-class binary classification: 0.3711% for 1-layer base line model, 0.3702% for 2-layer base line model and 0.3683% of error rate for 4-layer base line model and 2.935% for 1 layer. And 2.935% for 1-layer base line model, and 2.970% for 4-layer base line model, and they also performed a different experiment for multi task malware classification and they achieved a remarkable result.

The problem with this paper is its very coasty experiment. It requires a high capacity of processing machine and large dataset. The features are extracted directly from the binary file of the program, which are around 4,000 features for each file. Which is difficult to test this much features in normal computing machine.

2.11 Paper Summary

The objective of every researcher is to provide a better way in its field of study, all reviewed papers do the same, and their basic objective is to apply different methods to provide a security system that protects a computer from any kind of attack efficiently. As presented in the paper review section all the researchers follow different method and approach to reduce the error rate of the malware detection by applying different neural network algorithm.

The common thing most the paper have together is they are all trying to classify the malware either as a normal program or malignant or to their specific families by analyzing the binary file of the programs and extracting some features from data set. They achieved a very small error rate by using different algorithms. And it can be generalized increasing the dataset and the number of layers that most researchers used. It is possible to reduce the error rate in a very small amount but a significant one.

Generally all of them use the same approach with different methods and different amount of dataset. This paper presents a new approach of program classification and the methodology is completely different the one reviewed in the previous papers. Some other research is also trying to classify based on the behavior of the programs, their approach is still highly concerned by analyzing the binary file of the program. By learning what function calling structure and file accessing system the malware and normal programs used. And based on that they try to extract some pattern for malware if the structure of the program follows that pattern then that specific program will be classified as malware, otherwise it would be normal program, this approach is closer to the approach proposed in this research than all other papers, but it is not the same, our approach doesn't concern about the content of the binary file, the program structure or the file structure the programs used, our approach is just to study the user activity and to detect bad programs that don't authorize by the user of that computer. Basically, this paper is useful for personal computer protection. The objective of this research paper is to detect malware only by analyzing the user activity and resource usage by each process.

The previous papers basic objective is to reduce the detection error rate in general. The concern of this research is not to reduce the detection error rate, rather the research concerned to apply a new methodology to detect malware only by learning from the user activity. Generally, this research paper is completely different from the rest of the research performed in objectivity, methodology and approach applied to perform the research.

Chapter Three: Methodology and Design

This chapter will discuss the proposed research methodology and the design of the whole research, the experimental setups and related issues.

3.1 Research Methodology

The research methodology followed in this research is the following.

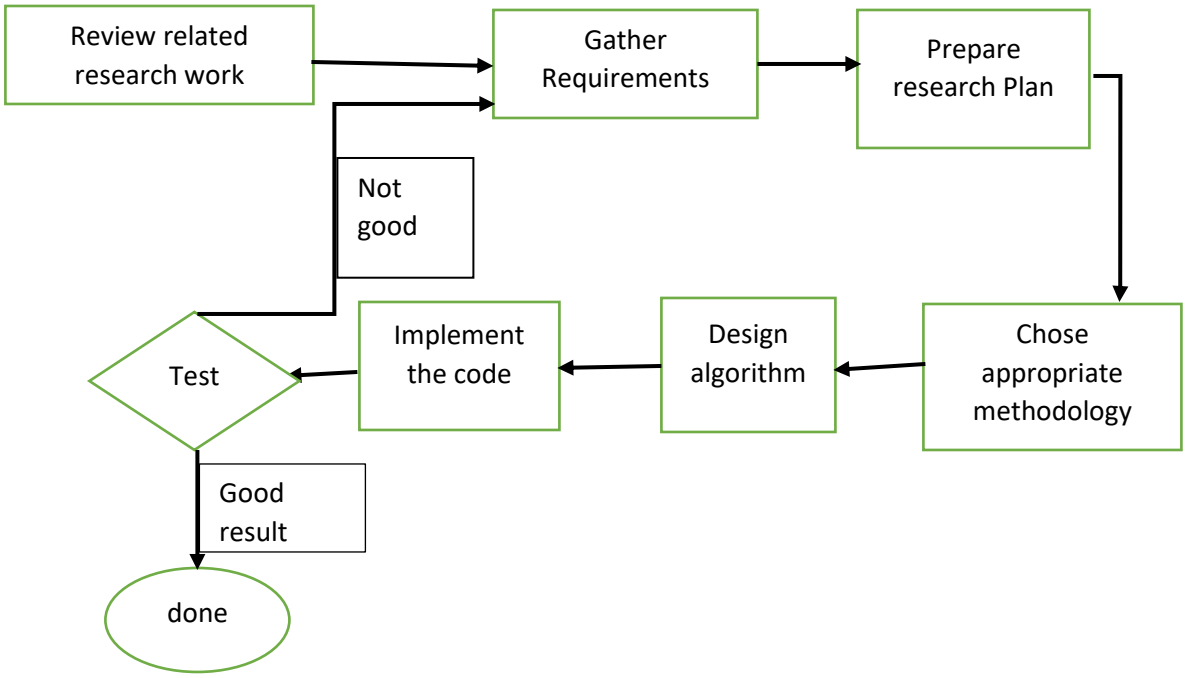

Figure 4: Research methodology

The first job in any research is to study the related work that has been done by other researchers in the area they are working on. Different research works have discussed in chapter two in detail, the objective of all researchers is to apply neural network system to detect or classify malware. They used almost similar approach, which is to crack the binary file of program and study different structural pattern that may differentiate malware from normal programs and to classify them to related families. They also accomplished an incredible result in terms of error rate reduction or increasing positive true test result, and they used the same approach to perform their approach and some of the researchers used to many learning datasets to increase their true positive rate, but still security is the biggest issue that exists currently in cyber world, this will lead us:

- The approach they are using is not efficient or the result they are presented are corrupted. If that is the case it will lead us to find

- New approach to detect malware, that is completely different from the approach that was been applied in the paper review mentioned and apply it to classify a process as a normal or malware or suspicious activity only by analyzing the activity of the process and the user history.
- To perform the same research with the completely different dataset and measure the performance and efficiency of the programs.
- To apply a different algorithm that is not been used before for such kind of problem.

- All the mentioned approach may solve the problem but the first solution "to follow a new approach" will support the proposed objective of the research.

3.2 Requirement Gathering and Parameter Definition

Requirement gathering is the second task performed for this research, requirements gathered by reading from different on-line resources, by reading different books, by studying the process behaviors in the computer. This phase identifies:

I. The parameter used,
II. The neural network algorithm used, and the number of layers used in the neural network.
III. The programing language used, the operating system, and
IV. Methodology followed to classify the malware.

3.2.1 The Parameters used

From the requirement gathered, the following parameters are identified for this research.

- CPU usage of the process: usually this parameter will be returned in % but it identified that the % returned by the operating system is not a normal percent, if it is the total CPU usage should be equal to the sum of the individual process % but it is not. This experiment writes four different programs. This program produces CPU bounded process when programs run in parallel. They use 100% CPU and when running all the four programs in parallel the amount of CPU Usage is reduced for all process and none of them are using 100%. And after we run this programs in a different mode for several times, we found out that the returned % is only for the % of CPU that the program is using, it is obvious most computers have more than one CPU inside so, this % is the usage of the processor that this process is using. So, to get the

actual total CPU Usage is not that much important for this research. But we can say the amount of CPU usage by the computer will vary by different factors.

- Memory usage: this is the next very important parameter that characterizes the Process property, there are three types of memory that would be used by every process:
    1. Physical memory: The memory area that is allocated for this specific area by the Kernel program
    2. Shared memory: The memory area that is shared with other process
    3. Virtual memory: The memory that is used when the actual memory allocated by the kernel is not enough, it is located in the hard disk of the computer.

All of them will be returned in number so it is not hard to use it in the algorithm. But it requires normalization.

- Status of the Process: Processes have their own state, we believe it will contribute to identifying the process characteristics, and there are four states of the process, Sleep, Running, Daemon or Zombie. So, we will assign a binary value for their state and this will be changed to numerical values, so it will be used in the computation.
- Time: this parameter will tell us for how long the process was runing in this machine despite its state. This parameter will be returned in numerical value.
- Program file location: this is another significant characteristic of a process where they put their file, most of the actual or normal programs save their file in program files, so we will put binary value for their location if they are in the program file, or public shared folders
- Program file Size: this is another characteristic of a program to differentiate from malware, since usually malware file size is very small, and normal application programs size usually larger. So, it requires putting bench mark and representing them in numerical value.
- The frequency of access: this parameter is very important but a little bit hard to consider it, because there is no common function to retrieve such information, so it need to keep a record of a process seriously, in a data base or in the file. As it records the date, time, process name whenever the process starts running.
- Owner of the process: this is another parameter chosen to classify the programs since it plays a major roll to detect where the program belongs. And we found out there are differently built in users with the operating system like GDM, syslog, avahi, … in addition to the users, we obviously know or created. So, we think these users may create leakage of information by executing some programs by their authorization without the actual user knowledge.

Numerical values will be assigned for regular users and such kind of users to use it in computation.

- Listening to a specific port (yes no or (0 and 1)): this parameter will tell if the program is communicating with other computers or not.

3.2.2 The Neural Network Algorithm and Model Chosen

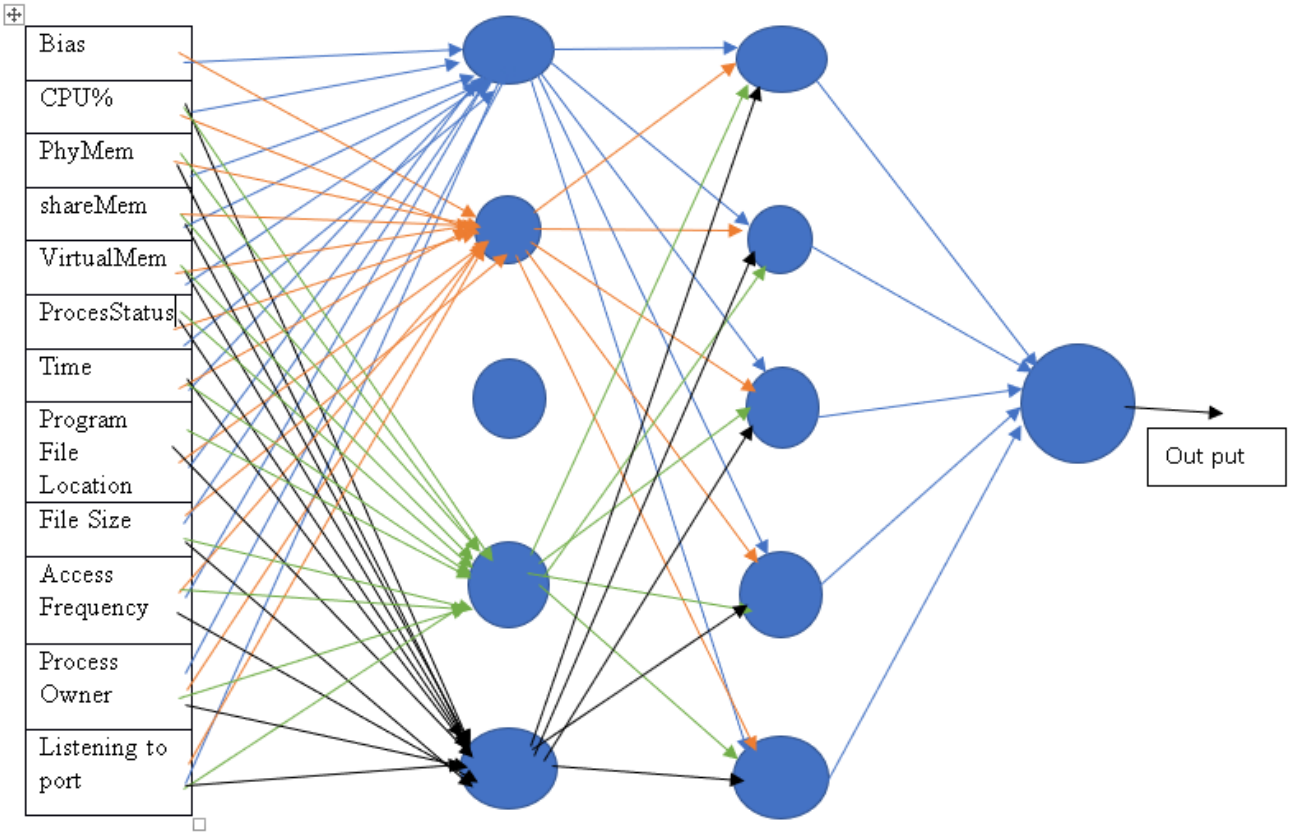

Figure 5: Proposed neural network model

3.2.3 Programing Language and Experimental Environment used

This research use C programing language for dataset collection code and MATLAB to train and prediction phase, C programing is the fastest language for execution, and we are familiar in c programing, in addition it will also give us the advantage of calling system functions easily as CMD commands, that is used to retrieve the parameters selected. Another reason is the experiment is conducted in Linux Environment, first it is open source we able to use or modify the OS codes as we want if it is required, another advantage is, it has many commands to retrieve the parameter selected, the third advantage is, we will enjoy C programing in this environment without installing external compiler and it will boost the execution speed of the program.

3.2.4 Neural Network Training Methodology

The following two approaches chosen for this research, the first approach is to collect dataset and to filter the required parameter which is going to be used in the neural network algorithm, and the second approach shows how the program capture Process, extract the most important feature from the captured process, evaluate using the trained algorithm and apply another neural network algorithm to detect and classify the process as malware or normal process. This approach will be used to detect a malware.

Now gathering this all requirement will lead us to the first job of the actual experimental work that would be to write an application to retrieve the parameters gathered.

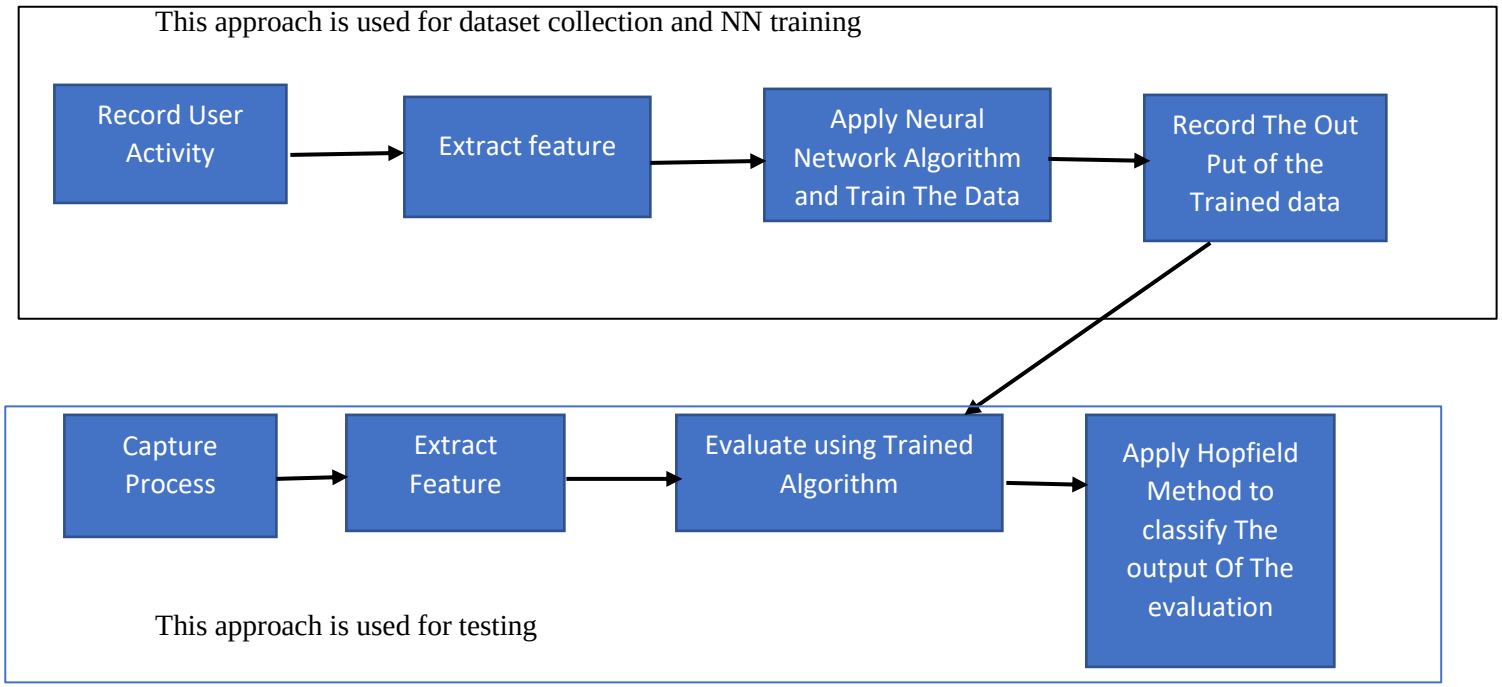

Figure 6 Neural Network Training Methodologies

3.2.5 Retrieve the Required Feature Methodology

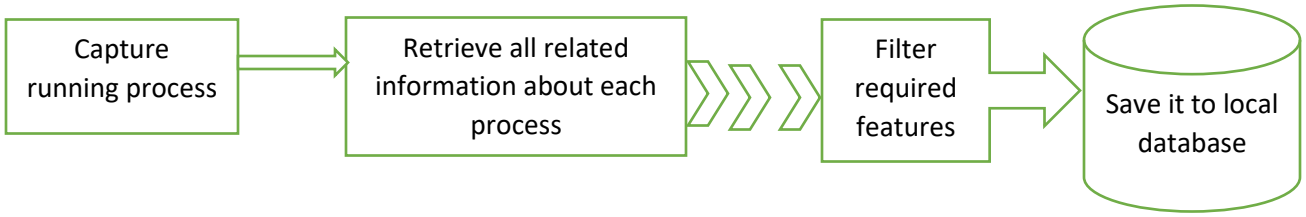

Figure 7: Feature Gathering Methodology

Recording user activity and retrieve the required feature is the basic of the whole experiment, and it is a difficult one. This program will retrieve, the PID, owner of the process, virtual memory

usage, physical memory usage, shared memory usage, process status, CPU usage, time elapsed, and the command being executed. All these parameters are retrieved using the system call called top command in Linux, and save the result in file, so that later the information will be retrieved from here, another script to retrieve the rest parameter from the operating system and to quantify the parameters is also need another program, which we already finished and ready to collect the dataset which will be used to collect all the dataset required for neural network training phase,

3.3 Design to Apply the Selected Approach

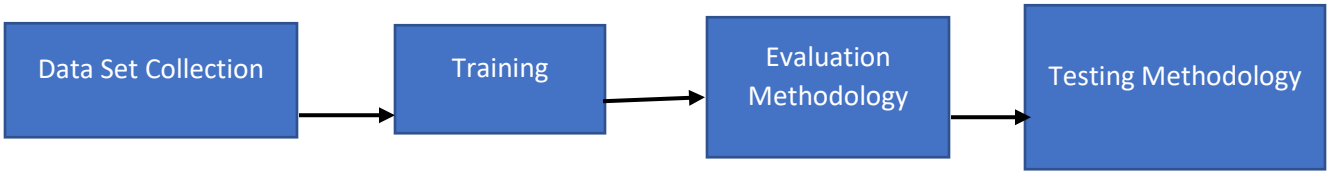

Figure 8: Design to apply selected approach

3.3.1 Dataset Collection and Training Plan

In data collection plan, collecting and analyzing data-set as well as jobs listed below are done.

- ✓ Develop a script program which retrieves all parameter
- ✓ Select experimental computer more than 3 computers that run Unix operating system
- ✓ Run the script programs on the selected computers and record and save data on each computer by running the program for 10 continuous days with a 10-minute interval for at least 4 times in a day for one hour while the user is using the computer.
- ✓ This experiment will provide 6 (number of samples taken in each iteration) X 4 (number of samples taken in a day) X 10 (number of days' sample data taken).

  From this there will be a total of 240 datasets for each process for each computer and the total number of datasets will be varied from computer to computer based on how many processes are running inside the user computer, so the total number of datasets for neural network for this specific research will be 240XN (N: number of process running during dataset collection phase).
- ✓ After Dataset collection phase completed filtering the dataset will be the next phase, filtering is the process of removing unwanted or unfitted data from the dataset collected from each

computer, after filtering the data set the dataset would be ready to train the neural network algorithm individually and become ready to protect against any malware or suspicious activity on that machine.

3.3.2 Flow Chart for program capturing and evaluation after algorithm training

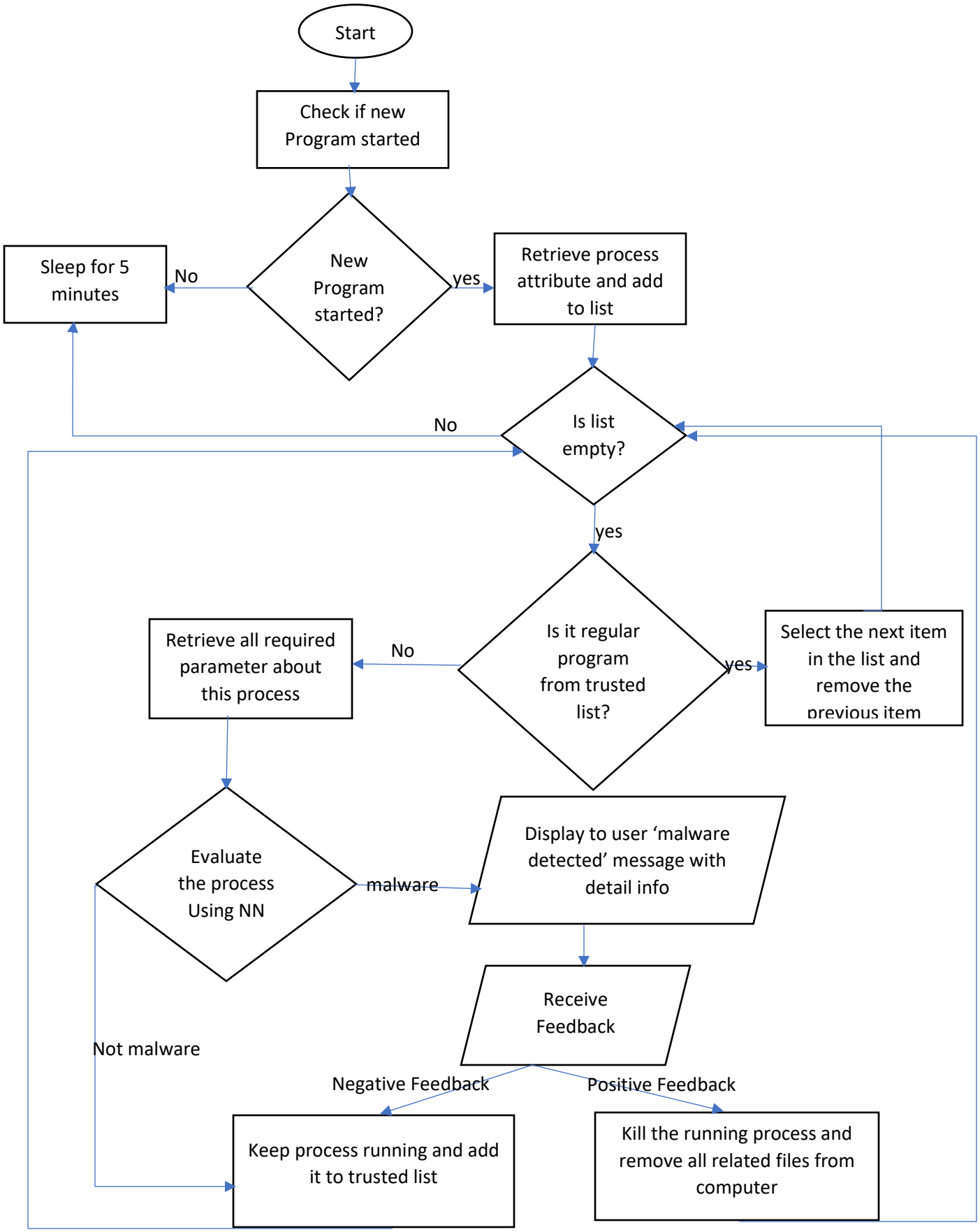
Start
Check if new Program started
New Program started?
No
Sleep for 5 minutes
yes
Retrieve process attribute and add to list
Is list empty?
No
yes
Is it regular program from trusted list?
No
Retrieve all required parameter about this process
yes
Select the next item in the list and remove the previous item
Evaluate the process Using NN
malware
Display to user 'malware detected' message with detail info
Receive Feedback
Not malware
Negative Feedback
Positive Feedback
Keep process running and add it to trusted list
Kill the running process and remove all related files from computer

Figure 9: Evaluation Plan after Training

After the training process is completed successfully there should be some sort of program which captures and evaluate the result. This program will be expected to run inside the computer forever. It is not processor consuming process; it starts running only when a new process starts running. Figure 9 flow diagram shows the after-training process. First the program will capture the new arising process, to check whether the process is new or not different approach could be used, but the simplest way is to check the registry in some time interval. Or to use the temporary file as a database to save the list of process that was been running after the last evaluation, and compare it with the currently retrieved process list, any way this implementation detail will depend on the developer's approach, in any case, it will not make any different in the final process.

The next phase is, if the new process is captured it would be to retrieve the related parameter required for evaluation and give to the neural network to evaluate those process. But if we keep evaluating every process started to run in a compute there will be a lot of repetition of evaluation process, so to remove this repetition the program will create to file as a database. The first one would be used to save the list of process that has been evaluated previously and approved by the user as a normal program (not malware) this programs will be trusted by the user and by this program too. The next file will save the list of programs that are condemned as malware by the user, this will save a lot of time and processing time of the computer, so the program will be fast. The program will first check every new arising process if they are in the pass list or in the black list files.

If the newly started program found in the pass list there is no need to evaluate this process by the neural network algorithm it is already evaluated and passed, so this process will be allowed to continue execution, but if the program is found the black list file, the program will stop execution and all related files to this program will be hunted down and removed from the computer.

The final case is if the process name is not in any of the lists, then this program is new. So it has to be evaluated by the program. To do this all related parameter will be retrieved from this specific process and filtered. After the filtration is conducted the parameters will be passed to the neural network algorithm which is already trained and ready to evaluate another process. The algorithm will receive this parameter and return the evaluation result back to the sender program. The received result would be some sort of number this number indicates whether the process is malware or not. To do this indication the hope field method of the very first neural network

activation function would be used. And the algorithm will return us whether this program would be classified as malware or not.

The returned result would either this process is classified as malware or not malware. If the process is classified as not malware then no need to make an additional assessment their fore, the program will go back to sleep and wait for the new program to start. If the process is classified as malware by Hopfield[8] algorithm, the program will make an additional assessment to make sure about this process. So, the program will send a message to the user all related resource usage and process name information to get confirmation whether it is malware or not. If the user confirms this process as a malware the process name and related information would be saved in the black list and the process would be interrupted and all related files to this program will be removed from the computer. If the user confirms the process as a normal process (not malware) then it will be saved to pass lit file and process will continue execution. The evaluation program will do this continuously as far as there is a new arising process on the computer. In meantime, the program will stay in the sleeping mood.

[8] See page 34 last paragraph for explanation

Chapter Four: Experimental setup and Recorded Results

This chapter contain Experimental setup of the research, the implementation of the design, and the output of each experimental.

4.1 Experimental Setups

To perform the experiment, the first job after collecting the data set is to develop the script which retrieves the selected parameters of the process attribute. The problem here is all the required attributes cannot be retrieved by one script, so in order to get the required parameter, we have to develop three different classes which use the output of one class as the input of the next one. Structurally it could be defined as follows.

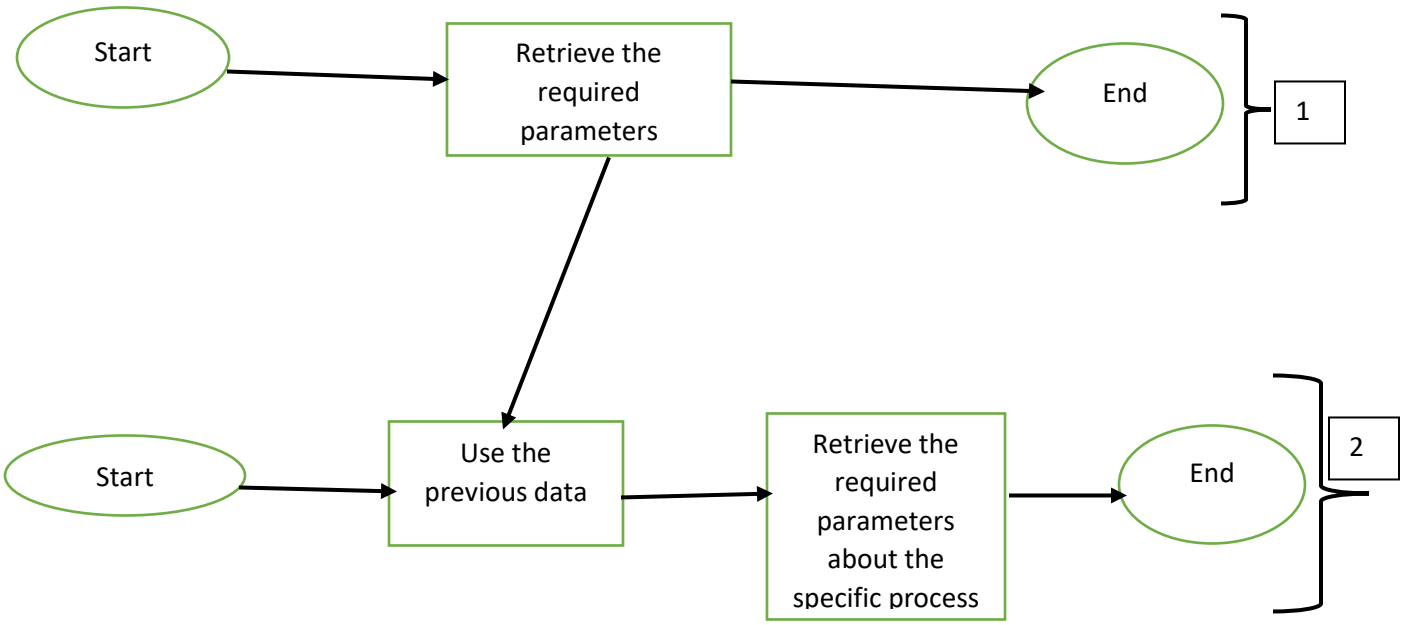

Figure 10: parameter retrieval approach

Figure 10 will retrieve basic information about each process, like process ID, process name, memory usage, and processor usage. This parameter is not enough for our selected parameters. So, the output of diagram one class will be used as input to the next class to retrieve the remaining parameters about each process so the parameters like Process ID and Process name would be used as input parameter for the second class(program).

After retrieving all the required parameters about each process there would be filtering phase. In this phase only, the parameters required would be filtered and saved to file (database). Figure 11 shows the selected and filtered parameter of each process from the test computer.

USER	Command	externalcomunication	FileLocation	VIRT	RES	SHR	Status	%CPU	%MEM	TIME
root	/usr/bin/p+	0	1	177956	99592	50336	S	25.0	2.5	0:00.36
tegegn	sh -c top +	0	3	4592	940	872	R	18.8	1.5	0:00.75
syslog	/usr/sbin/+	0	1	256536	3724	2784	S	0.0	0.1	0:00.06
message+	/usr/bin/d+	1	1	49216	5840	3616	S	0.0	0.1	0:00.85
avahi	avahi-daem+	1	1	49328	352	0	S	0.0	0.0	0:00.00
systemd+	/lib/syste+	0	1	65844	5784	5128	S	0.0	0.1	0:01.17
gdm	(sd-pam)	0	1	254028	2364	32	S	0.0	0.1	0:00.00
whoopsie	/usr/bin/w+	1	2	399188	13724	11828	S	0.0	0.4	0:00.03
kernoops	/usr/sbin/+	1	2	56744	2596	2152	S	0.0	0.1	0:00.01
rtkit	/usr/lib/r+	0	1	188076	3244	2952	S	0.0	0.1	0:00.02
colord	/usr/lib/c+	0	1	333528	15956	10884	S	0.0	0.4	0:00.16
www-data	/usr/sbin/+	1	1	317872	7936	2052	S	0.0	0.2	0:00.00

Figure 11: Filtered Process parameter

The final phase of the dataset collection would be to create a uniform measuring unit, some of the parameters are numerical some of them are character, some of them are percentile, since it is not computed in the same area, it needs to be converted to similar measurement, numerical measurement is chosen since it is used by the majority of the parameters needed. And it also used by the neural network as a common measurement.

The percentile measurement is converted to a numerical value just by removing only the percentile sine. It doesn't need any additional conversion since it always lies between zero and one hundred. The other parameter required to convert to a numerical value is whether the process is using the internet or not. For this measurement zero and one (0 & 1) are chosen when zero represents the process is not using the internet and 1 represents the process is using the internet.

The last measurement required a conversation to numerical representation is the file location of the command being executed. normal programs usually stored in the same directory for example in windows operating system when we install the program the commands and all necessary files would be copied to in a directory called program files.in this case, we try to represent the most likely to be a normal programs directory with a higher numerical value, the less like program location to a middle numerical value, and the less or not likely program file location files to a very small numerical value. Since there is three class from our classification the higher numerical

value assigned to 2 (two), the middle numerical value assigned to number 1 (one) the very small numerical value assigned 0(zero).

The final value which would be the input to the neural network would be all numerical value. The screen shoot is shown here.

```
CPU% VIR     RES   ShR    S Pr usr Time FileL EXCOM
25.0 177956 99592 50336 1 3 0 2 0 0
18.8 2025404 342660 141700 1 3 0 2 0 0
12.5 2678996 339216 143460 1 3 0 2 0 0
6.2 3742884 184488 89728 1 3 0 2 0 0
6.2 1433236 12812 9492 1 3 0 2 0 0
0.0 4592 852 784 1 3 0 2 154072 0
0.0 4592 1704 1576 1 3 0 2 154072 0
6.2 48080 4196 3532 1 3 0 3 0 0
0.0 220248 8648 6460 1 3 0 2 0 0
0.0 0 0 0 1 3 0 2 0 0
```

Figure 12: quantified process parameters

The parameters selected shows the basic information about the process is running by the user. That means the programs will be created or start only when the user opens them or they are an illegal process.

Figure 12 shows the list of parameters taken to train the neural network algorithm, the data has been taken from three different computers as a research proto type, from each computer the data set has been collected for ten continuous days while the user is performing day to day activity in the PC the user is not aware of the program that collects the data set and it doesn't bother to monitor the dataset collector program. After the data set collection is completed eight (8) normal user applications have been chosen to test the trained algorithm and another eight (8) malware programs have been injected in the computer to test the algorithm.

This chapter discusses the neural network output and the test we performed and we will discuss the representation and the output result of all test case computer.

- The Objective of this training is to train the algorithm as the error is as small as possible.
- Assumptions.

1. all programs that are recorded as a dataset in the data collection process are normal programs (the computer is free of any malware)
2. The expected value of every process evaluation is expected to be near 0.5, from the collected data we train the set as they give as the expected value, the expected value for each process output is 0.5, so if the trained algorithm gives us 0.5 results for each process than the neural network is converged perfectly, if the evaluation result is greater than this result or smaller than the expected result, in that case, the algorithm fails to converge.
3. This research train only the normal programs so that the algorithm will be able to identify only if the process is a normal program or not,
4. The decision whether the algorithm evaluation result of the process is positive or negative will be decided by another neural network algorithm called Hopfield Method. This algorithm is a very simple one; it accepts input (i.e. the evaluation result of the process by the trained network) and compares it with the minimum and maximum output result recorded during the training dataset. If this (input data) is between the two minimum and maximum values acquired during the training, then it will give positive result (this process will be characterized as a normal program), otherwise whether the input is smaller than the minimum value recorded during training or larger than the maximum value recorded during training, the out will be negative (this specific process will be suspected to be a malware program).

4.2 Test Computer one Experiments

Test computer is hp ProBook 450 model core i3 computer, it has 4.2 Ghz speed and 4Gb RAM the dataset is collected while the user is using the computer as usual. From this computer it's been able to collect 35,874 (thirty-five thousand eight hundred seventy-four) a number of processes in our dataset collection process and able to get 29,378 (twenty-nine thousand three hundred seventy-eight) number of the filtered process which would be used directly in the training process.

4.2.1 Test Computer 1 Neural Network Training Model

The following is the dataset model recorded after the training phase has been completed. Figure 13 shows the Neural network output for the Test computer one, Figure 10 shows the output of the dataset relays between 0.494 minimum value and 0.506 of the maximum output evaluation result. From the data set collected for ten continuous days, it can be concluded that the resource usage of the processes in this specific computer give the maximum value of 0.506 and a minimum value of 0.494 when they evaluated by the neural network algorithm using the adjusted weight during training. So, it can interpret the result of individual processes evaluation result based on this result. Any process that is running in this specific computer and being evaluated by the algorithm trained by this data set should give the evaluation result between 0.495 and 0.506 other wise they will be categorized as a malware program.

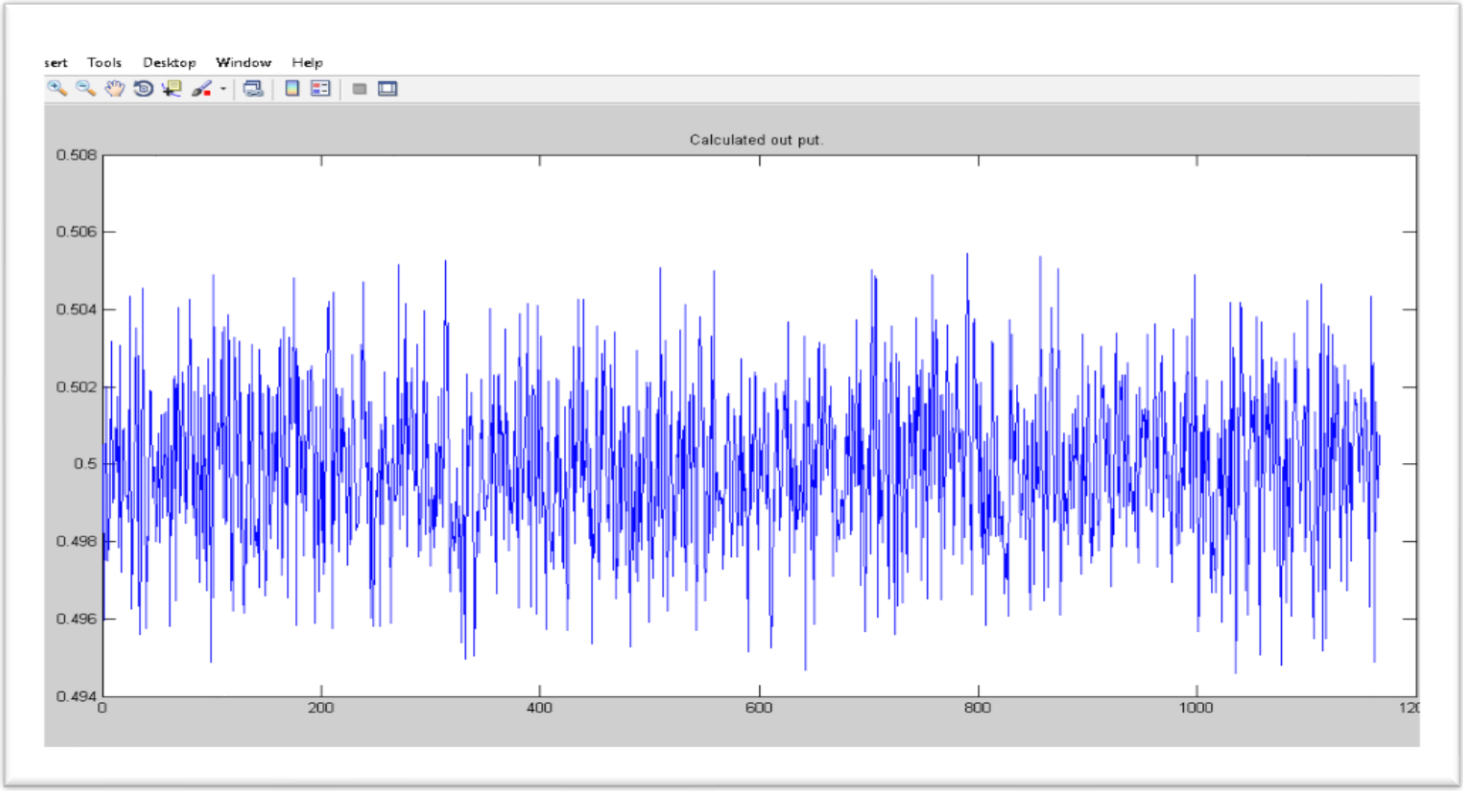

Figure 13: Trained Neural Network output for test Computer1 (neural network model)

4.2.2 TestComputer1 Recorded Error

Some error rates are recorded in the process of training. As common sense the error should be as small as possible, otherwise, the neural network is not converged, the following error rate graph recorded for all trained dataset, as it observed in the graph the algorithm is well converged. Figure 14 shows the error rate of the training algorithm for the experiment of computer one.

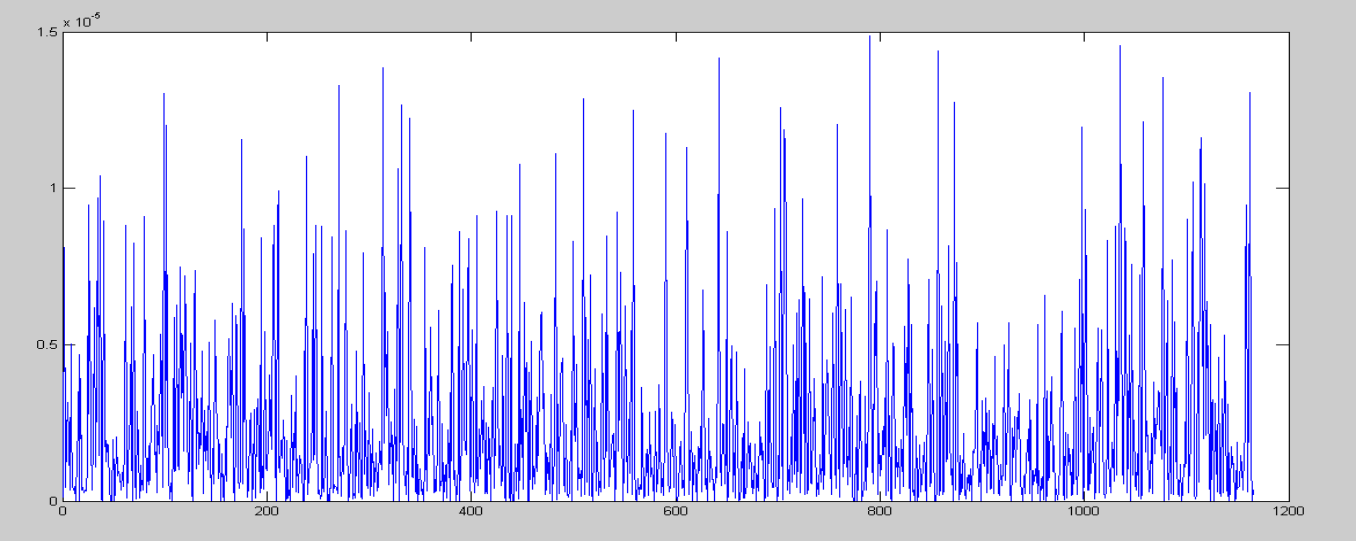

Figure 14: Error Rate of Test computer1

Figure 14 shows the error is very small with four decimal digit values, so the error is as good as we can go to the next step, which is testing. After the training process is completed the next job will be to test some random data by using this algorithm.

4.2.3 Tests Conducted in TestCOmputer1

The test is conducted by eight different normal application programs executed in this computer and eight other different malware injected programs in the same computer, the parameters are chosen to evaluate using the trained algorithm collected. The following result recorded for an individual process. The test result for malware programs and normal programs are presented separately as follows.

Table 1: Algorithm evaluation Test result for malware injected programs

No	Name	Evaluation value	Evaluation error	Categorized as
1	malware 1 (infinite mathematical operation performer)	0.6442	0.0016	malware
2	malware 2 (just start listen to port)	0.382	0.00318	malware
3	malware 3 (generate zombie process and each process display output)	0.901	0.0805	malware
4	malware 4 (start and copy the file in the current directory infinitely)	0.4270	0.0017	malware
5	malware 5 (send message to specific port continuously)	0.5091	$1.43x10^{-5}$	Normal Program
6	malware 6 (malware that searches and all kind of files inside a folder)	0.6552	0.0120	malware
7	malware 7 (malware that sends message continually to make computer busy (denial of service) for another process in the same computer)	0.6041	0.0054	malware
8	malware 8 (a program that receives a message from another process which acts as server/denial of service/)	0.3989	0.0051	malware

The result shows that from the malware injected programs the model evaluated 87% of the malware programs as an “a malware suspected program” and the remaining 13% of the programs are characterized as a normal program even if they are malwares.

Table 2: Algorithm evaluation Test for malware Free Programs

No	Name	Evaluation value	Evaluation error	Categorized as
1	Ekiga	0.4862	$9.469xe^{-5}$	malware program
2	Ubuntu software center	0.4864	$9.1816xe^{-5}$	malware program
3	Phot editor	0.4995	$1.2754e^{-7}$	Normal Program
4	Salsa web mail explorer	0.5	$1.1248xe^{-9}$	Normal Program
5	Command top	0.501	$4.5107e^{-9}$	Normal Program
6	Spread Sheet Program	0.5004	7.3712e-08	Normal Program
7	Sudoku game	0.5000	1.1498e-12	Normal Program
8	remina conecter	0.5000	1.0458e-11	Normal Program

From the test programs executed to evaluate the normal programs by the trained algorithm the model able to classify 75% of the programs as a normal program, and the remaining 25% of the test data has been characterized as a malware suspected program. The best part is according to this research plan the program cannot be categorized automatically to malware and deleted from your computer. The result will be notified to the user and requested to confirm the suspected program is actually malware or normal application, if and only if the user confirms the suspected program as malware, the process will be interrupted from running and the related file will be removed from the computer. This is because some time when the computer get busy some process may run for a long time, they may consume large storage and processer time, so during evaluation they may be categorized as malware, which is false result so to save the embarrassment to delete the normal program the user needs and to remove all related file, it is necessary to confirm from

the user directly by displaying all the necessary data (CPU usage, memory usage, how long the process was been running, etc.) to help user give knowledge-based decision.

From the test programs, we can conclude for test computer 1 we have reviewed as the following.

Table 3: Computer Test Programs Test output

False Positive	**43.75%**
False Negative	6.25%
True Positive	37.5%
True Negative	12.5%

As shown in the review table the results have been concluded based on the randomly selected test program results, the result shows good hope for the future anti malware evolution which would be artificially intelligent based malware detection system.

4.3 Experiment on Experiment Computer two.

This is another test computer chosen to perform our experiment, the computer is Lenovo model, it has 2.4GHz processor speed and 8 GB amount of RAM. The dataset collected while the user is using his computer for his regular job. 30,242 (thirty-five thousand eight hundred and twelve) number of process are recorded and from this process 17,095 (seventeen thousand and ninety-five) filtered number of process are ready for training. All the dataset has been collected while the user of the computer is using his computer for regular day to day activity, so we didn't' have to run any special programs for this experiment. The data has been collected for ten continuous days.

4.3.1 ExperimentComputer2 Training Model

The following neural network model is recorded from the training.

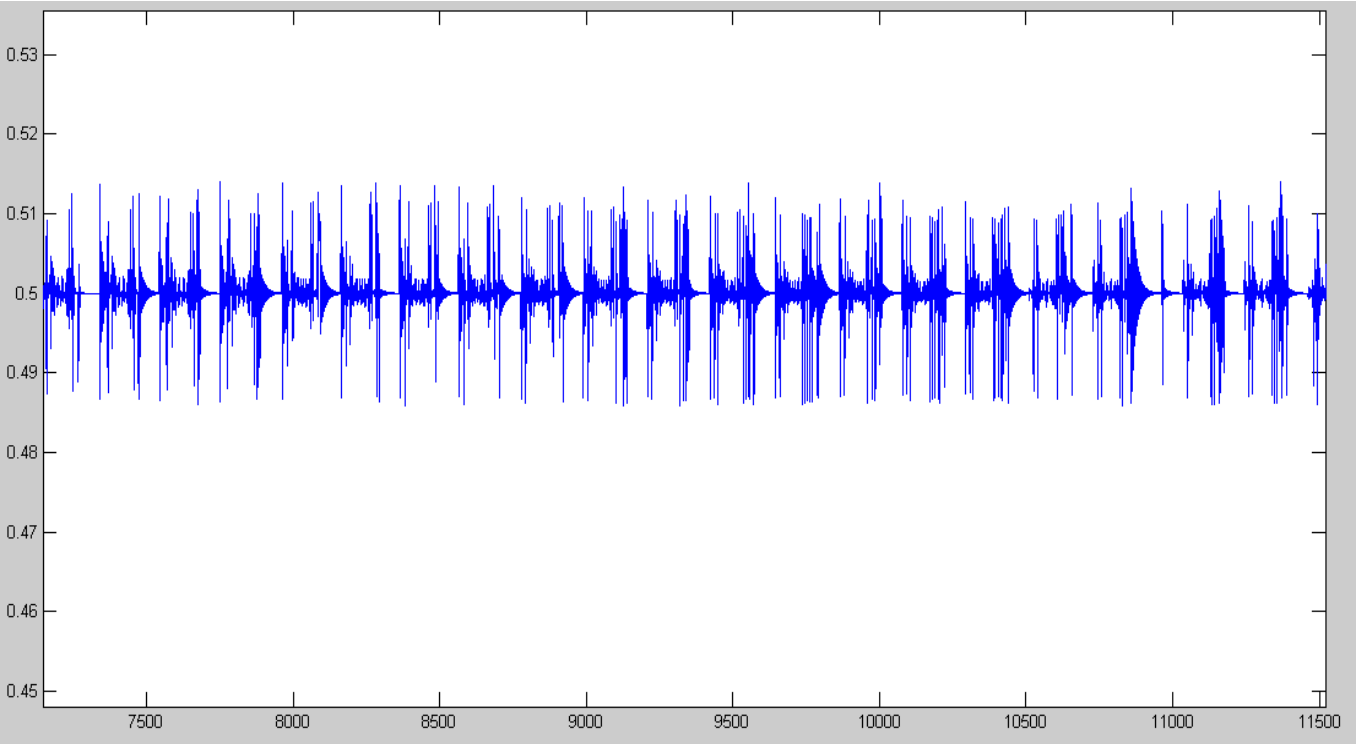

Figure 15: Neural Network Training output of TestComputer2

The training output shows a very good test dataset training result. The maximum training result recorded from the test output is 0.52 and the minimum training result recorded is 0.48. This means any output that would be gathered using the trained algorithm expected to be between these two minimum and maximum values. Results above the maximum recorded training result

and below the minimum trained output value would be categorized as a suspected malware program.

4.3.2 Experiment Computer2 Training Model Error

The following error rate recorded to make sure the error difference we may able to record and the error we may get from the actual evaluated output wouldn't make too much difference.

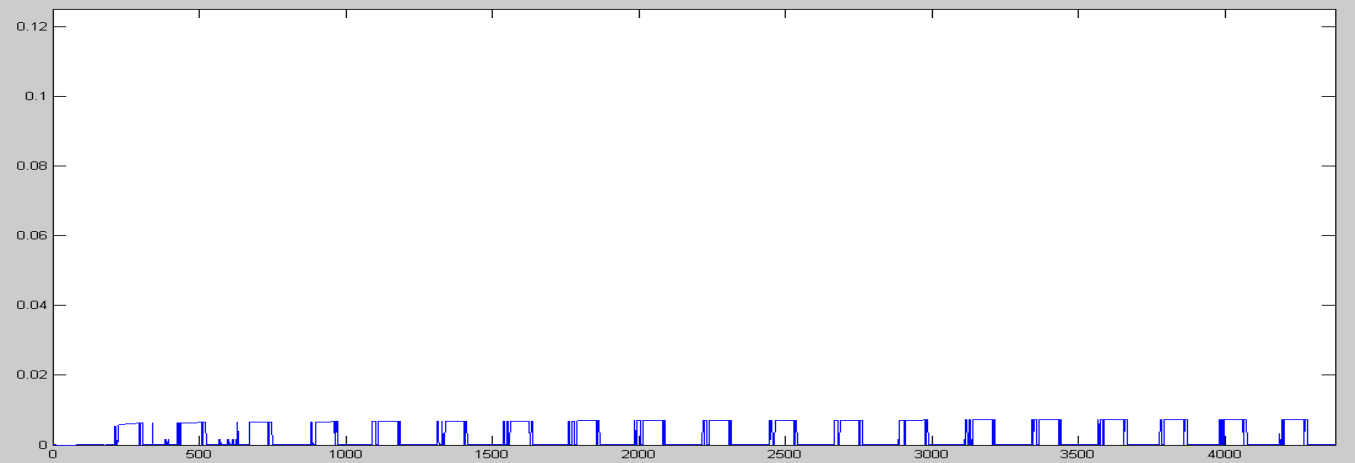

Figure 16: Error rate of neural network training of Test Computer2

The error graph for computer 2 shows the maximum error of 0.02, that means the evaluated process error is not expected to exceed this value. If the error is greater than this value there must be some problem during the data collection or during training, or the training data may be over fitted.

4.3.3 Tests Conducted on ExperimentComputer2

The test data for computer 2 is almost the same programs as the test programs used for computer1 and computer 3. The number of programs is totally 16 from these programs 8 of them are normal application programs and the remaining 8 are malware injected programs.

The test is conducted by running all programs separately in different days while the user of the computer is not aware of the running test programs. As shown in the training output and error graph, the output of computer one and computer two are completely different. This different arises from the fact that the training datasets are separate. Since these two users in the test case computer don't use the computer in the same way, we expect the test result also be different from the first test case computer results. The following output data has recorded from the test performed.

The categorization of the programs as a regular program or suspected program has been conducted by Hopfield algorithm. Hopfield algorithm use the values of numerical value recorded from the training algorithm. The training algorithm output shows the maximum training evaluation result for normal application programs is 0.52. That means any process evaluated by this algorithm is not supposed to be larger than this value if it is normal program. Otherwise it would be categorized as a malware programs. And the minimum evaluation result that would be evaluated by the training algorithm is supposed to be 0.48. The value of a process that been evaluated by the trained algorithm is not less than the given minimum value if it is the process will be categorized as a malware program. Table 4 shows the test result recorded from the test computer 2.

Table 4: Test computer 2 test results

No	Name of process	Evaluated value	Evaluated error	Categorized as
1	malware 1 (infinite mathematical operation performer)	0.6520	0.0116	malware
2	malware 2 (just start listening to port spy process)	0.2322	0.0359	malware
3	malware 3 (generate zombie process and each process display output)	0.6330	0.0116	malware
4	malware 4 (start and copy the file in the current directory infinitely)	0.3989	0.0051	malware
5	malware 5 (send message to specific port continuously)	0.5058	1.6911e-05	Normal Program
6	malware 6 (malware that searches and deletes all kind of files inside a folder)	0.4870	8.4731e-05	Normal program
7	malware 7 (malware that sends message continually to make computer busy (denial of service) for another process in	0.5488	0.0012	malware

	the same computer)			
8	malware 8 (a program that receives a message from another process which acts as server/denial of service/)	0.4380	0.0019	malware

Table 4 contains the list of malware injected programs that used to test the testcomputer2 and the trained algorithm result recorded during execution, the error rate exhibited while the program is computed and the classification of the malware as a normal program or as malware suspected programs. As shown in the table, six of the programs have been classified without error correctly and the remaining two programs are classified wrongly as a normal program even if they are actual malware injected programs.

Another test conducted in this test experiment is only by using normal programs most of them were not been running during the dataset collection process, so they are completely new to the computer. This will help us to analyze the capability of the algorithm to detect new application and categorize them correctly. The following table contains the test result recorded from the test computer2 for normal application programs.

Table 5: Testcomputer2 Result for Normal Application Programs

No	Name	Evaluated value	Evaluated error	Categorized as
1	Firefox update center	0.6520	0.0116	malware program
2	Ubuntu software center	0.4864	$9.1816xe^{-5}$	malware program
3	Phot editor	0.4995	$1.2754e^{-7}$	Normal Program
4	Salsa web mail explorer	0.5	$1.1248xe^{-9}$	Normal Program
5	Command top	0.501	$4.5107e^{-9}$	Normal Program
6	Spread sheet program	0.5004	7.3712e-08	Normal Program
7	Sudoku game	0.5000	1.1498e-12	Normal Program
8	remina conecter	0.5000	1.0458e-11	Normal Program

The results from the test computer for both normal programs and malware injected programs will be summarized as follows in the next table.

Table 6: Testcomputer2 Test Results Summary

False Positive	**37.5%**
False Negative	12.5%
True Positive	37.5%
True Negative	12.5%

The test computer2 test result summary shows the neural network algorithm classified 25% of the total test programs as a false. The remaining 75% of the programs are classified correctly. From the total programs chosen for test, the algorithm trained by the test computer2 dataset 75% of the programs has been recognized correctly by the algorithm. 25% of the programs have not been recognized correctly. For any running normal program and anti-malware programs there are 0.25 chance of probability the normal programs may be predicted as a malware. And the 0.25 probability of malware programs may run inside our computer and this algorithm may not able to suspect it as a malware process.

From this test the user may be victim in two ways by the trained algorithm. The first one is normal programs have a chance to detect normal applications as malware. The good thing is the antimalware program is not going to kill and remove the related files to this process without the user confirmation. So we are good except bothering the user to confirm the suspected program is really a malware based on the resource usage retrieved.

The second one is if the malware is running inside our computer and the trained algorithm is not able to detect it. The user will be attacked by this malware even if the anti-malware program is running.

4.4 Experiment on Experimentcomputer3

The third test computer chosen for the experiment is HP core i5 laptop which has been used by the regular user which uses Linux operating system for his day to day work activity. The dataset for training is collected for ten continues days 4 times a day which run for one hour every time since it starts running. From this computer 37,620 (thirty-seven thousand six hundred twenty) number of processes is collected for the entire data set collection period, and from this process 21,234 (twenty-one thousand two hundred thirty-four) number filtered process used directly for Neural network training.

The methodology used is the same for all three-test case computers chosen. The methodology flow is collect dataset using dataset collector program then filter some useless dataset and train the neural network using the filtered dataset.

4.4.1 ExperimentComputer3 Training Model

The following neural network training model is recorded for this experimental computer.

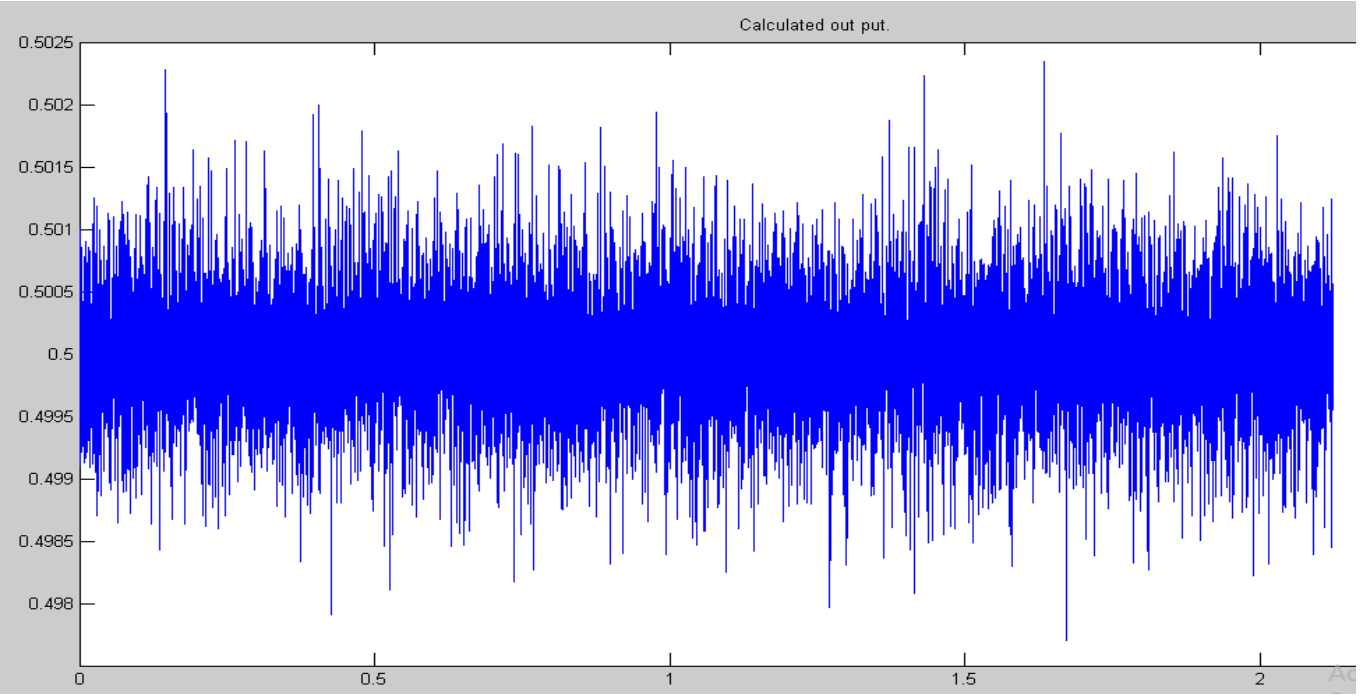

Figure 17: Testcomputer3 output

The output in Figure 17 shows the maximum evaluation result of the trained dataset for this computer would 0.503 and the minimum evaluated dataset evaluation result would be 0.475.

From this result, it can be concluded as if any process that has been evaluated by the trained neural network, the output of the evaluation would be expected to lie between these two values. If the evaluation result of the tested process value is greater than this value, the process would be suspected as a malware process. The detail resource usage will be displayed for user and confirmation would be given by the user, if the user confirms the process as a malware then the process will be deleted from the computer entirely. If the user confirms the process is not a malware the process will be registered to pass list for next time confirmation and the process will be allowed to run.

4.4.2 Experiment Coputer3 Training Model Error

In addition to the training output, the neural network error rate is important to give a decision about the process.

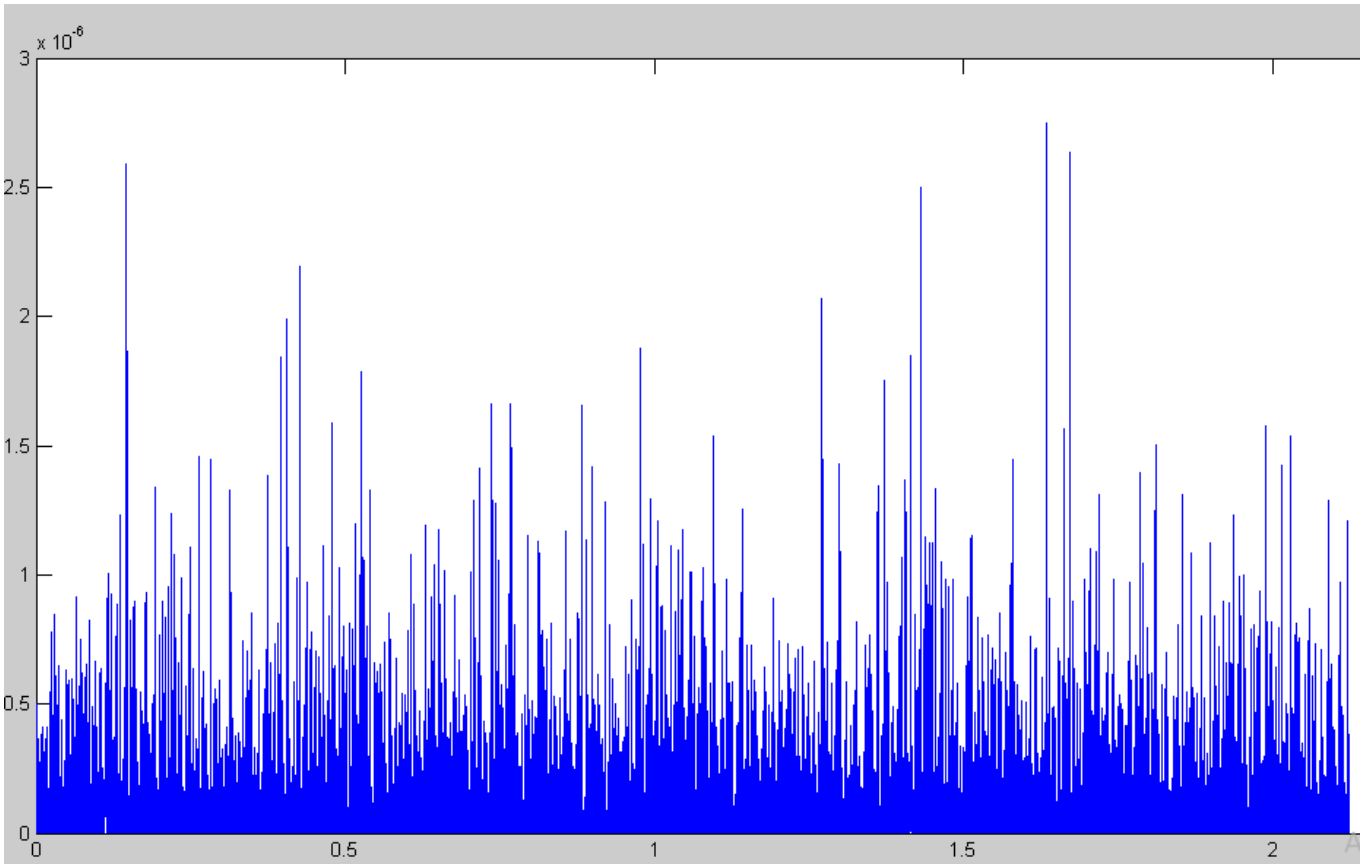

Figure 18: Test Computer 3 output Error

The recorded error graph clearly shows the recorded error is most likely constant and less than 0.0025. This result is good, but it is possible to reduce the training error rate, which has a greater contribution for the algorithm performance in terms of accuracy, if the error is as small as possible the algorithm accuracy will be high, that means the algorithm will able to detect process

correctly weather they are suspected malware process or normal application program. To reduce the training error follows two methods. The first one is to increase the training data set. The second one is to use the more sophisticated algorithm. The third one is to increase the number of layers, neurons, and iteration. But, for this research the acquired error is good to go.

The process usually evaluated by taking the training output as a bench mark to classify weather the process is suspect or not, but these errors will support how the algorithm is trained well, a well-trained algorithm exhibit a smaller error rate, which means a great accuracy.

4.4.3 Tests Conducted on ExperimentComputer2

The test computer 3 has been also tested in the same way in the previous two test computer. After neural training has been completed and achieved the best output required, and the smallest error rate as much as we can. The algorithm will get ready to test the process inside the computer and identify weather they are regular application or suspected malware programs. To test this computer, eight (8) normal application has selected which the user was not been using them during dataset phase. And other eight (8) malware injected programs which have different properties.

First lets' see the first test executed in the test computer which is the malware injected programs test. When we run this test we have to be careful, since they are malware injected programs they may destroy our computer or they may corrupt our file permanently. These kinds of program need to have isolated directory which is executed from terminal for precaution. And another program which retrieves all parameters we need about the process should be running during execution. Then we filter out that specific process and able to get all the parameters we are looking for. After completion the malware injected process should be terminated. The parameters retrieved will be evaluated by using the trained neural network. Table 7 contains the test result of the malware injected programs test result.

Table 7: Test computer 3 malware Injected Programs Evaluation Result

No	Name of process	Evaluated value	Evaluated error	Categorized as
1	malware 1 (infinite mathematical operation performer)	0.521	0.0116	malware
2	malware 2 (just start listening to port spy process)	0.401	0.0359	malware
3	malware 3 (generate zombie process and each process display output)	0.5030	0.0116	malware
4	malware 4 (start and copy the file in the current directory infinitely)	0.4994	0.0051	malware
5	malware 5 (send message to specific port continuously)	0.5058	1.6911e-05	Normal Program
6	malware 6 (malware that searches and deletes all kind of files inside a folder)	0.4870	8.4731e-05	Normal program
7	malware 7 (malware that sends message continually to make computer busy (denial of service) for another process in the same computer)	0.5488	0.0012	malware
8	malware 8 (a program that receives a message from another process which acts as server/denial of service/)	0.4380	0.0019	malware

The next table contains the test program run for normal programs in the test computer3. The previous test result is only for the test results that contain only malware injected programs. And

this one is a program that is free of any malware suspect. The process information retrieved while the user is using the computer chosen for training for a daily purpose, so the test conducted as the same environmental setup as the training dataset collection phase.

Table 8: Test Computer3 Normal Programs Test Result

No	Name	Evaluated value	Evaluated error	Categorized as
1	Firefox update center	0.523	$5.13e^{-3}$	malware program
2	Ubuntu software center	0.4870	$9.1816e^{-3}$	Normal program
3	Phot editor	0.4985	$1.2754e^{-5}$	Normal Program
4	Salsa web mail explorer	0.5012	$3.224xe^{-9}$	Normal Program
5	Command top	0.4473	$3.4107e^{-3}$	malware Program
6	Spread sheet program	0.5021	5.272e-08	Normal Program
7	Sudoku game	0.3781	9.1498e-3	malware Program
8	remina conecter	0.4871	3.0458e-5	Normal Program

The test result of the previous two tables shows the selected test programs evaluation result in the trained algorithm by the dataset collected from a similar test case computer.

In the first test result from the total test case programs, 75% of the programs have been predicted correctly. The remaining 25% of the test programs have not predicted correctly. That means if malware programs are running inside the computer, there are 0.75 probabilities the algorithm may capture this malware and there is 0.25 probabilities the malware may pass unsuspected.

For starters this is a good result, if we able to increase the number of dataset we used for training and apply more sophisticated algorithm we may able to achieve a better result. To increase the number of datasets we need to increase the number of days to collect the dataset, for this research the dataset has been collected only for ten continuous days. That means the algorithm has going to

predict the program's categories only based on the ten day knowledge. So, if we increase the number of days the algorithm has a better chance to learn all user activities pattern perfectly and it will able predict more precisely.

The second table contains only the normal application programs test result for test computer3 from this table we can see 62.5% of the applications have been classified correctly as a normal program. And the remaining 37.5% of the programs has been falsely classified as malware suspect programs. This is a good result since this is the first new approach to detect malwares based on user activity. But, it will be improved through time, since the programs detect once the program is malware and request confirmation from user. And if it turns out to be a normal application, the program will automatically mark this process as normal application. And for future there will be no calculation to be performed. The program will be automatically categorized as a malware application. So, the program will learn from its mistakes through time. That means after some point there may not be any normal application categorized as malware suspect since it's going to record almost all the normal application process in the special pass list file.

Table 9: Testcomputer3 Test Results Summary

False Positive	**37.5%**
False Negative	12.5%
True Positive	31.25%
True Negative	18.75%

The summary table shows that from the total test experiment conducted in testcomputer3 32.25% of the programs has been miss classified. And the remaining 68.75% of the test programs are correctly classified to their respective categories. Depending on the number of training set we have the test result is very promising.

To summarize, the experimental result conducted, we chose three different computers that owned and used by different users for a different purpose, and we collected the dataset for ten continuous days in each computer while the user is using the computer for daily activity. Collecting dataset is

a little bit hard since it totally depends on the availability of the computer owner. And it is totally in different locations. Data collection takes more than 50 days to complete the dataset for all three selected test computers. After the dataset collection is completed, we start to compile the dataset content. In the process, we are able to observe there is some process that may affect our algorithm performance. So, we decided to include filtering mechanism before we start the algorithm training, the things that we worried are some programs that have been started by the operating system and almost not use any source and they are not running by the user knowledge, so using this process as training dataset may pull back our training processes back.

After the filtration process is applied, we feed the dataset to the neural network algorithm for training, in average the training process takes one and half days for one-time training, since we decided to find the point where the algorithm going to converge, we have to run the same training programs many times by changing different parameters, like the number of layers, or the number of iterations.

After running the algorithm many times, we able to get a satisfactory converging point. The trained weight values are saved in the computer separately for each experimental computer. The training outputs for every test computer are also recorded. The saved weight value used to evaluate other test programs and record the output in the test output tables.

Generally, we can say we are able to implement the antimalware system that works not by analyzing the binary code structure of the programs, but by analyzing the user activity and resource usage by each process. We can say this is a very success full implementation and it would be a breakthrough for future work, since it has a very promising future.

Chapter Five: Result Discussion

The main objective of any researcher is to provide a new method to achieve a better result or to reduce execution time, or to increase the accuracy of the output. In malware detection area there are many researchers conducted till to day and they will continue to do so, in the experiment we presented in the previous chapter we tried to present a new approach to ward malware detection methodology and we have presented our findings in detail. In this chapter, we will try to compare the findings of some researchers with the experiment we conducted.

The comparison of this result may not be in the same area, but it will explain a lot of differences between the researches. the research are not one directional way it will also open a new eye to follow the different direction the one that we are using in the anti-malware technology systems approach.

The comparison of the researches has been made from a different perspective. The comparison point we chose for this research is the research approach they followed. The number of datasets, the behavior of the program, amount of Resource they need for training, and the final result acquired by the researchers.

The research methodology that is being followed by the researcher has been taken as one of the measurements we are going to compare. This measurement will let us know if different researchers try to follow different methods or they are following similar methods just by varying other parameters to get a better output. Since research is all about finding another way to perform the same activity to find a better output, we can measure how the researchers introduced a new thing in their research work.

The number of datasets they used for their research is another measurement we need to compare with our research work. In this perspective the process of collecting large dataset is time consuming.

The behavior of the algorithm or program is to tell whether the program will keep learning from its past or not. Most algorithms are one-time training and use it for life time. This kind of algorithm is classified as static behavior. Some algorithm will get a chance to learn from their past and not to repeat that again. This kind of algorithm behavior is dynamic behavior.

The resource usage of the algorithm during training is also another metrics we chose to use for comparison. Some algorithms are resource consuming and they are not going to be trained or tested in a personal computer they may require super computers to train.

The last measurement we chose for comparison is the output they get from their experiment. This will help to compare different approaches followed and the final result acquired. Since the final out put value is all we need to the actual usage of the research work we need these parametric metrics very much. Table 10 summarizes different research work value and approach being used with this paper research work.

No	Research title and Author	Research approach	No Dataset	Behavior	Resource required	Experimental result
1	*[Tang, 2016 #27]*	Extract the binary program file, Study the API call sequence, find a pattern that most malware use by using the deep neural network, classify based on the training output.	83 malware process log file It's a small number of process to train and classify the malwares, so it seems the output is biased by not trained well, so the prediction is going to be random	Once trained and used for life time. Or static it cannot update	Personal computer	For three different condition by varying the number of Layers 0.80, 0.96, 0.92 has been recorded The maximum is reasonable so it would be 0.96 accuracies.
2	[li, #18]	Extract the source code of the programs, teach the algorithm different codes that are already malware, try to draw a pattern for malware codes, apply two algorithms one, to extract features about the code, and the second to train the algorithm using extracted	494,021 training data set and 311,029 for testing	A static process, the algorithm will be trained ones and used for all time for all environment.	Trained and tested in average performance personal computer.	For three experiment they conducted the able to get: 92.20%TPR, 1.58FPR and an accuracy of 92.10%

		features.				
3	[wang, 2016 #28]	Extract the binary file of the program, study the API call sequence of the malwares and normal applications, try to create some pattern based on the API call sequence extracted from the binary file, by training the extracted binary files using deep learning Neural network model. Than classify the rest of the programs based on the classification output of the algorithm.	14,860 number of total datasets which categorized into five malware groups.	Static behavior the algorithm will be trained once by the acquired dataset and will be used forever.	The resource they used is not stated	In their presentation, they only presented the training performance and the test performance, the Train performance is claimed to be 99.7% and the test performance is 99.00%.
4	[tang, 2016 #26]	Receive a binary file of malwares for the dataset and the binary file will be mapped into a histogram map, then they will analyze the binary file distribution of the malware file and try to make some sort of	They used 431,926 binary files of different class malware.	The behavior of this approach is static, since they once retrieved the malware sample of	The resource they used for training and testing is not	They are able to get 67.7% of TPR with 0.1% of FPR and they are able to get the accuracy of 0.99794 under the give curve rock.

		common feature using Deep learning neural network algorithm. Based on this algorithm the files will start to be classified as malware or non-malware programs.		different class malware then they apply this dataset to train the algorithm and their algorithm not able to know if the algorithm made a mistake or not, so it is the static approach.	identif ied.	
5	Neural Netwo rk Based malwa re and Suspic ious Activit ies Detect ion Syste	Study the user behavior i.e. record the list of process the user use daily for some continuous days, extract important features to characterize process from the user's point of view, use the recorded dataset to train the neural network algorithm,	The training dataset will depend on the personal computer to be tested, i.e. it is not fixed. For this experiment, there are three different test computers for each we used 29,375 for TC1 17,095 for TC2 21,234 for TC3	The behavior of this program is dynamic, the algorithm may misclassify some of the programs if the program is	In the averag e capaci ty of Person al compu ter to be used for the daily	For the three tests being run the following results has recorded. TC2 FP 43.75% FN 6.255 TP 37.5% TN 12.5% TC2 FP 37.5%

	m Based on User Activit y	after training detect any running process inside the computer, evaluate using the trained algorithm and classify the process as malware or normal application.		miss classified as malware the user needs to confirm the detected process is actually a malware program, in that case, the program will able to learn if it makes mistake, that makes this approach to have dynamic behavior.	activit y.	FN 12.5 % TP 37.5 TN 12.5% TC3 FP 37.5% FN 12.5% TP 31.25% TN 18.75%

Table 10: Different Research Work Comparison with this paper work

The comparison table between this research work and the other five research papers shows that almost all the five-research presented here follow a similar approach to classify or detect malwares, the difference arises the algorithm they use and the parameters they are going to train for the Neural network. The major difference between this research and the previously presented research works is that in this a new method to detect malwares without extracting features from the binary file of the malwares is introduced.

The result recorded from the experiment is also very promising. Even if the accuracy acquired from other researcher is very high it shows they recorded from 0.8 - 0.99 accuracy. We are able to get the accuracy of 0.75 which is a very good result for this paper foundation. The number of datasets used to train the algorithm is small for all test computers. This is because the dataset has been collected only for ten continuous days. That means the proposed method is giving the decision based on ten days of the user to program interaction knowledge

- The other reason also the programs that are running inside this computer are not guaranteed they are 100% free from any kind of malware this will lead us the model might be corrupted at the beginning so it cannot give us the precise decision.
- The test conducted in this research cannot accept feedback from the user and able to correct the mistakes, but when the implementation is final the program will have a chance to correct mistakes, which may lead to producing a very high accuracy for the next round. If we able to create a perfect environment to collect the dataset for every training computer there will be a high probability the result acquired may be better than this, and if we manage to increase the number of days the dataset is going to be collected from ten days to one month or more the final result will be improved well.

Chapter Six: Conclusion and Recommendation

Conclusion

This paper presented anew methodology to hunt down malware program from your computer without consuming too much resources of the computer. The main advantage of this research is the algorithm developed in this research is not going to attack and kill all suspected malware programs like the one we usually encounter in day to day anti-virus usage. In most anti-malware programs they have to run alone, if there is another antimalware program running in the same computer than they will kill each other. But in this paper the finding shows that it doesn't matter what kind of programs are running in your computer, if the user confirms ones the program is not dangerous, the anti-malware program wouldn't bother to take action. So you can able to use more than one anti-malware programs in one computer at the same time.

This research work introduced a new method for malware and suspicious activity detection systems. We tried to show that human activity could be learned and applied to protect the individual personal computers from any kind of attack even without concerning if the malware is old or the newly arrived one. We are able to apply Artificial neural network concept for something unusual kind of data, usually artificial neural network algorithms are used in image and/or wave processing, but in this research work we are able to show that the neural algorithm is also applied to learn human and computer interaction behavior, if we able to collect appropriate parameters and enough dataset.

In this research, we are able to get an average accuracy of 0.746 or 74.6% malware detection rate. Other researchers exhibited a large accuracy result from 0.8 to 0.96 in different researches. but by applying a new methodology we can say this is the good breakthrough for future works that may be conducted in this area, specially the objective of the anti-malware program is to protect individual Personal computers from any sort of dangerous attack that may slow down the performance of the computer or those malwares that destroy the files and important resource of the computer. We can say this research can be used as a system administrator for individual computers and protect the individual computer from any kind of malware successfully.

Future Work and Recommendation

The future work for this research would be to test on many computers under different environment and justify the proposed methodology will be used to meet its basic objective which is to protect any individual personal computer from any sort of malware attack by studying the user activity. To test the algorithm by applying the suggested methodology to increase the accuracy acquired in this research work.

The Research works presented in this paper used a very lousy algorithm and a very small number of data set, for future work we will suggest increasing the number of days the dataset is going to be collected which may lead to having a great accuracy.

The algorithm used in this paper is every simple feedforward neural algorithm I suggest to apply more sophisticated re enforcement learning algorithm, I believe they will give a better result, and make the algorithm intelligent by itself.

This research work has been done basically to protect individual personal computers from any kind of attack without analyzing the code structure or need to update the malware signature database daily, I recommend if this research would be applied for servers and supercomputers it might give better protection then the other anti-malwares being used.

Reference

1. ‘An introduction to neural computing’ [Book: Aleksander, Igor, and Morton, Helen: 1990: Chapman & Hall London: Volume: 3]
2. ‘Deep learning-based feature selection for intrusion detection system in the transport layer’ [Generic: Aminanto, Muhamad Erza, and Kim, Kwangjo: 2016: volume 1]
3. ‘A Deep Learning Approach for Network Intrusion Detection’ [Jornal article: Quamar Niyaz, Weiqing Sun, Ahmad Y Javaid, and Mansoor Alam: 2011: volume 1]
4. ‘Combining restricted boltzmann machine and one side perceptron for malware detection’ [Conference Proceedings; Benchea, Răzvan, and Gavriluţ, Dragoş Teodor;2014]
5. ‘Computer security: art and science’ [Book; Bishop, Matt, publisher: Addison-Wesley Professional; 2003, volume 1]
6. ‘How virus softwar works’ [webpage; https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques/ on May 4, 2017, Tech Target private company]
7. ‘malware classification on time series data through machine learning’ [Jornal Article; de Almeida, Diogo Moutinho; 2016]
8. ‘Comparison deep learning method to traditional methods using for network intrusion detection’ [Conference Proceedings; Dong, Bo, and Wang, Xue, Conference Name: Proc. IEEE ICCSN]
9. ‘malware evolution’ [Web Page; east, LastLine lab, June 8, 2018; https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/]
10. ‘An introduction to neural networks’ [Book; Gurney, Kevin; 2014; CRC press]
11. ‘Neural networks and learning machines’ [Book; Haykin, Simon S, Haykin, Simon S, Haykin, Simon S and Haykin, Simon S; 2009; Pearson Upper Saddle River publisher]
12. ‘MtNet: a multi-task neural network for dynamic malware classification’ [Conference Proceedings; Huang, Wenyi and Stokes, Jack W; 2016; Springer publisher]
13. ‘Commo malware’ [web page; inc., Avast software, on November 26, 2018, https://www.avast.com/c-malware]
14. ‘malware Statistics’ [web page; institute], Av-Test independent security; on November 21 2018; https://www.av-test.org/en/statistics/malware/]

15. ‘A deep learning approach for network intrusion detection system’ [Conference Proceedings; Javaid, Ahmad, Niyaz, Quamar, Sun, Weiqing and Alam, Mansoor; conference name: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS); Publisher: ICST (Institute for Computer Sciences, Social-Informatics]
16. ‘deep learning for zero-day flash malware detection’ [Conference Proceedings; Jung, Wookhyun, Kim, Sangwon and Choi, Sangyong; Conference Name: 36th IEEE symposium on security and privacy]
17. ‘Intrusion detection system using a deep neural network for in-vehicle network security’ [Journal Article; Kang, Min-Joo and Kang, Je-Won %J PloS one; volume 6]
18. ‘A hybrid malicious code detection method based on deep learning’ [Journal Article; Li, Yuancheng, Ma, Rong and Jiao, Runhai %J methods; volume 5]
19. ‘An introduction to computing with neural nets’ [Journal Article; Lippmann, Richard %J IEEE Assp magazine; volume 4]
20. ‘Core is creeper and reaper’ [web page; MEtcalf, John; cited on November 21, 2018; http://all.net/books/virus/part2.html]
21. ‘Cyber-attack -hack security’ [web page; Pagliery., Virginia Harrison and Jose; on may 4 2017; http://money.cnn.com/2015/04/14/technology/security/cyber-attack-hacks-security/index.html]
22. ‘An introduction to the modeling of neural networks’ [Book; Peretto, Pierre; 1992; Publisher: Cambridge University Press; volume 2]
23. ‘Network security: private communication in a public world’ [Book; Perlman, Radia, Kaufman, Charlie, and Speciner, Mike; 2016; Publisher: Pearson Education India]
24. ‘History of malware’ [web page; Saxe, plc., gdatasoftware; cited on June 6, 2017; https://www.gdatasoftware.com/seccuritylabs/information/history-of-malware/]
25. ‘Deep neural network-based malware detection using two-dimensional binary program features’ [Conference Proceedings, Saxe, Joshua and Berlin, Konstantin, Conference Name: Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on; Publisher: IEEE]
26. ‘Deep learning approach for network intrusion detection in software defined networking’ [Conference Proceedings; Tang, Tuan A, Mhamdi, Lotfi, McLernon, Des, Zaidi, Syed Ali

Raza and Ghogho, Mounir; 2016; Conference Name: Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference; Publisher: IEEE]

27. 'malware detection with the deep neural network using process behavior' [onference Proceedings; Tobiyama, Shun, Yamaguchi, Yukiko, Shimada, Hajime, Ikuse, Tomonori and Yagi, Takeshi 2016; Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual; Publisher: IEEE]

28. 'A multi-task learning model for malware classification with useful file access pattern from API call sequence' [Journal Article; Wang, Xin, and Yiu, Siu Ming %J arXiv preprint arXiv; 2016]

29. 'Common malware Types and cyber security' [web page; Zeltser, Neil DuPaul Lenny; cited on May 3, 2018; https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101/]

Index

Code to collect a dataset

```
#include<sys/stat.h>
#include<string.h>
#include<stdio.h>
#include<errno.h>
//#include<signal.h>
//function definition
void readfromfile(const char*fileName);
void writetofile();
long int fsize(const char*filename);
int search_in_File(const char *str);
int qantifyPath(const char *str);
int quantifyUser(const char *str);
int quantifyTime(const char *str);
int quantifyStatus(const char *str);
//void nextsampleschedule(int signal);
//main function bodies
 void main()
{
int count=0;
printf(" Dataset Collecter program Is Runing now.\n");
printf(system("date"));
while(count<4)
{
writetofile();
printf("writetofile function executed");
readfromfile("parameter1.txt");
printf("read from file function executed\n");
printf("The %d roound dataset collection is Compleated Sucessfully. \n",(count+1));
sleep(720);
//wait(10000);
count++;
printf(system("date"));
}
printf("dataset collection for 1 hour has been compleated successfully");
}
//function to read cpu and memory usage
void writetofile()
{
printf("data process parameters are being retreived and saved to file.\n");
system("top -n 1 -b >parameter1.txt");
system("lsof -i >parameter2.txt");
}
// function to read the content of the above function from file
void readfromfile(const char*fileName)
{
char buff[255];
FILE *fp,*fwr,*qfr;
fp=fopen(fileName,"r");
fwr=fopen("NwPara.txt","a+");
qfr=fopen("qparameter.txt","a+");
//printf("going to line number 6\n");
for(int i=0;i<=6;i++)
{
fgets(buff, sizeof(buff), fp);
}
 while (fgets(buff, sizeof(buff), fp) != NULL)
  {
    buff[strlen(buff) - 1] = '\0';
//*****************tokenize the data retreived from file*****************
  char *token;
    char *rest = buff;
const char *nn[12];
```

```
 int n=0,temp;
long int sz=0;
   while ((token = strtok_r(rest, " ", &rest)) &&
n<13)
     {
         nn[n]=token;
        // printf("%s\n", nn[n]);
         n++;
}
//printf("processing the parameters retreived
from process\n");
sz=fsize(nn[11]);//file size of the runing process
temp=search_in_File(nn[11]);//searches if this
process is using internet
int ppp=qantifyPath(nn[11]);
int uuu=quantifyUser(nn[1]);
int ttt=quantifyTime(nn[10]);
int sss=quantifyStatus(nn[7]);
//printf("value of path when quantified: %d\n\n
value of sername : %d\n\n",nnn,mmm);
//printf("writing the quantified parameters to the
dataset list.\n");
fprintf(fwr,"%s %s %s %s %s %s %s %s %ld
%d\n", nn[8],nn[4],nn[5],nn[6], nn[1], nn[10] ,
nn[11], nn[7],sz,temp);
fprintf(qfr,"%s %s %s %s %d %d %d %d %ld
%d\n", nn[8],nn[4],nn[5],nn[6], uuu, ttt ,ppp,
sss,sz,temp);
//printf("\n after the first line\n");

//remember to write a new line to the file
//*************************************
  }
// close both opende file. e
  fclose(fp);
  fclose(fwr);
  fclose(qfr);
}
// function to gate the file size of the program
long int fsize(const char*filename){
struct stat st;
if(stat(filename,&st)==0)
return(st.st_size);
else
return 0;
}
// search if the proces is comunicating through
internnet.
int search_in_File(const char *str){
char fname[]="parameter2.txt";
      FILE *fp;
      char temp[512];
      if((fp = fopen("parameter2.txt", "r")) ==
NULL) {
         printf("file does not exist\n");
                return(-1);
      }

      while(fgets(temp, 512, fp) != NULL) {
                if((strcmp(temp, str)) == 0) {
                        return 1;
                }
      }
}

int qantifyPath(const char *str)
{
int vall,rtrn;
vall=strcmp(str,"/lib/*");
if(vall==0)
rtrn=1;
else
vall=strcmp(str,"/opt/*");
if(vall==0)
```

```
rtrn=1;
else
rtrn=0;
return rtrn;
}
int quantifyUser(const char *str)
{
      int val=0;
//result = strcmp(example1, example2);
int result=strcmp(str,"tegegn");
if(result==0)
      val=3;
else
result=strcmp(str,"root");
 if(result==0)
      val=2;
 else
   result=strcmp(str,"gdm");
      val=1;
 return val;
}
int quantifyTime(const char *str)
{
int n,values;
n=strcmp(str,"0:59:59");
if(n<=0)
  values=3;
   else
    {
 n=strcmp(str,"1:59:59");
     if(n<=0)
       values=2;
       else
         {
         n=strcmp(str,"5:59:59");
         if(n <=0)
          values=1;
          else
                values=0;
}
}
return values;
}
int quantifyStatus(const char *str)
{
int n;
if(strcmp(str,"R")==0)
n=3;
else if(strcmp(str,"S")==0)
n=2;
else if(strcmp(str,"D")==0)
n=1;
else
n=0;
return n;
}
```