Contents

[DECLARATION i](#_Toc74316774)

[BONAFIDE CERTIFICATE ii](#_Toc74316775)

[ACKNOWLEDGEMENTS iv](#_Toc74316776)

[ABSTRACT v](#_Toc74316777)

[CHAPTER 1 vi](#_Toc74316778)

[**INTRODUCTION:** vi](#_Toc74316779)

[CHAPTER 2 ix](#_Toc74316780)

[**RELATED WORKS:** ix](#_Toc74316781)

[**THEORETICAL BACKGROUND** x](#_Toc74316782)

[**MALWARE TAXONOMIES:** x](#_Toc74316783)

[**Adware:** x](#_Toc74316784)

[**Backdoor:** x](#_Toc74316785)

# DECLARATION

I hereby declare that the thesis entitled “**MALWARE DETECTOR USING DATA MINING TECHNIQUES**” submitted by me to the Department of Computer Science, Central University of Tamil Nadu in partial fulfillment of the requirements for the award of the degree of Master of Computer Science of Central University of Tamil Nadu is a bonafide work carried out by me under the supervision of **Dr. R. Saranya**, Department of Computer Science, Central University of Tamil Nadu.

I further declare that the work reported in this thesis has not been submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

**Date: 24.05.2021 Signature of the candidate**

Beulah Evanjalin A

**Place: Chennai Name of the student**

# BONAFIDE CERTIFICATE

This is to certify that the project report entitled **“MALWARE DETECTOR USING DATA MINING TECHNIQUES”** submitted by Beulah Evanjalin A (**P191304**) of II year / IV semester during the academic year 2019-21 in partial fulfillment of the requirement for the award of the degree of Master of Computer Science of Central University of Tamil Nadu is a record of bonafide work done carried out by her under my guidance.

This work has not been submitted elsewhere for the award of any other degree to the best of our knowledge

**GUIDE**

**HEAD OF THE DEPARTMENT**

Submitted for the University Viva-Voce Examination held on 17. 05. 2021

**INTERNAL EXAMINER(S)**

# 

# ACKNOWLEDGEMENTS

I have immense pleasure in expressing our sincerest and deepest sense of gratitude towards our guide **Dr. R. Saranya** for the assistance, valuable guidance and co- operation in carrying out this Project successfully. I have developed this project with the help of Faculty members of our institute and I am extremely grateful to all of them. I also take this opportunity to thank Head of the Department Prof. P.V.S.S.R Chandra Mouli for providing the required facilities in completing this project.

I am greatly thankful to our parents, friends and faculty members for their motivation, guidance and help whenever needed.

ABSTRACT**:**

Malware, which is a kind of software construct that are intentionally created for malicious activity like data destruction, stealing someone's personal and sensitive data, causing major damage to the system or the data available in the system or authorizing someone's system via network. So, a wide range of Antivirus constructs are also been developed over the past few decades in order to detect and remove malware that are existing in the system. However, the detection rate would be measured only using the best’s way of defence criteria. One such criteria is by applying data mining and machine learning techniques which are used to classify the behaviour of the executable file (either as a benign or a malicious) by extracting some sequence of available features from the executable files.

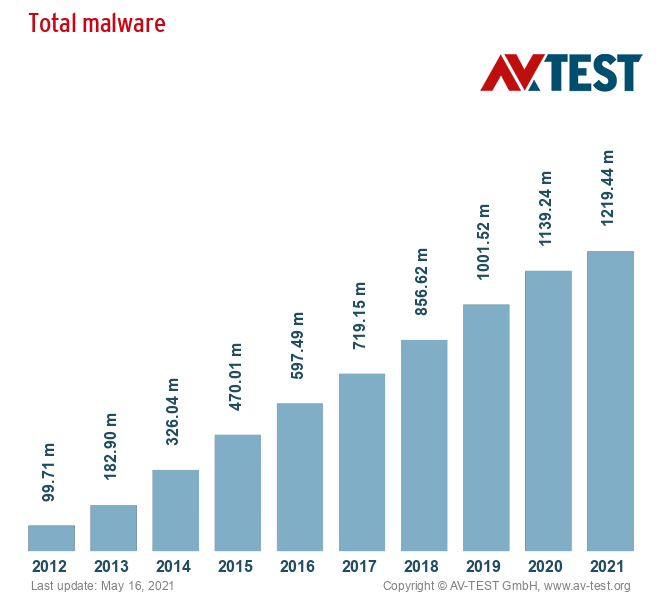
The purpose of this project is to model a detector which classifies the software binaries based on static malware analysis method by using character string extracted from the executable as the feature. This project also addresses various facets like data representation, feature selection, feature compression, classification algorithms (with classifiers like Random Forest classifier, Decision Tree classifier, K-Nearest-Neighbours were evaluated), as well as evaluation metrics with active learning. And the dataset used for this project consisted of 991 malware software binaries files and 428 benign files of different formats.

# CHAPTER 1

## **INTRODUCTION:**

A brief view on the histories of malwares provides us an insight about the presence of malicious constructs that has been travelling with us since from the day the computer has evolved. During 1970s, the first malware was documented on ARPANET named, the Creeper virus [1] which was a self-replicating program with the message "I'm the creeper, catch me if you can!” From this data onwards till now, a massive fight against the malware has never ended and has getting increased each new day.

Also malware threats have evolved as large as Stars in the Sky (that is, large volume or numbers of malwares with different varieties and functionalities [2].



In order to cope with this malicious software’s cyber security professionals and other research people need to constantly revamp their defence techniques. Specifically, anti-virus software providers are responsible for all such preventions and detections nowadays.

Conventionally, anti-virus industries were pinning their hopes on a signature based methods and heuristic based methods. A signature is basically an algorithm which uniquely identifies a certain type of malware whereas heuristics are set of protocols that are dogged by cyber security professionals after the analysis of the malware's behaviour. And both of these approaches need the malware to be investigated before implementing the protocols and heuristics.

The objective of malware analysis is to provide information about the given software binaries and there exist two kinds of analysis, namely,

1. Static analysis and
2. Dynamic analysis.

Static analysis is about examining the information in order to provide clues about the code's behaviour without any executions or installations. And on the other hand, dynamic analysis is about examining the code's behaviour by actually executing or installing it in order to check what will it be doing and how it will affect the system.

With fast technological improvements over the past few decades, malware detection and prevention has become one of the major requirements nowadays. Also during the last decade, machined learning based malware detections have triggered a great shift in improving anti-virus engines. Evidence is that there arises the number of studies published over the Internet in the last few years on malware detection that leverage machine learning techniques. According to dimensions.ai [3], the number of research papers published under the topic 'malware detection' in the year 2021 is 3382, and 9983 in the year 2020 which is approximately 2680.78% increase from early 2000s. This symbolically implies that there exist huge demands for cyber security professionals who are skilled with data science techniques. Also, the traditional watchmen monitoring system becomes obsolete nowadays especially when data science techniques emerged in finding threat patterns in huge volumes of data.

# CHAPTER 2

## **RELATED WORKS:**

[4] A survey was presented on malware detection which uses static method of malware analysis. In addition they examined some features in the software binaries and provided some exclusive techniques to dodge the malware.

[5] Here, in this article they provided the study on both static and dynamic analysis. However, this study doesn't provide any description on feature taxonomy and data mining techniques that was used.

[6] In this study, they categorized their works like

* The target test that they tried to achieve
* Different features that they extracted from Portable Executable file (PE files) and
* Machine learning algorithms that they used.

[7] In this paper, they provided the taxonomy for machine learning based malware detection by addressing some kinds of features and their selection techniques.

[8] Here, they proposed the extraction methods based on PE headers, DLLs and API functions and methods which based on Naive Bayes, J48 Decision Trees, and Support Vector Machines. And they achieved the highest overall accuracy with the J48 algorithm (99% with PE header feature type and hybrid PE header and API function feature type, 99.1% with API function feature type).

## **THEORETICAL BACKGROUND**

This part provides the background knowledge that is necessary to understand the malware detection under the usage of the machine learning methods.

## **MALWARE TAXONOMIES:**

Malware is a kind of software construct that are intentionally created for malicious activity like data destruction, stealing someone's personal and sensitive data, causing major damage to the system or the data available in the system or authorizing someone's system via network. Depending on the purposes, a malware can be divided into various categories.

### **Adware:**

Adware is basically software that downloads advertisement onto user’s computers whenever they are online. This malware is used as a source of revenue generator as it displays advertisements on users screen ace or the screen. They may also tend to slow down the user computer and affect the users browsing experience.

### **Backdoor:**

A backdoor can be described as a malware type created by the developers as means to access various content on a users’ computer in an unauthorized access mode. Malicious codes are normally inserted during installation to create these backdoors for unauthorised remote access for malicious activities.

**Bot:**

A bot is described as a self-propagating malware that is designed to perform various repetitive tasks on the internet. They as also known as web crawlers that are used to gather passwords of users online and also used to launch DDos attacks. Bots are self-propagating because they possess wormlike capabilities. Bots are part of a botnet, a network of interconnected devices, which are controlled using command and control (C&C) software.

**Downloader:**

Acts like a Trojan that installs its self onto a device and once internet connection is established it downloads other malicious software onto the computer it has attached itself to.

**Launcher:**

A launcher is also known as a loader that primarily is used to load various malware. The launcher operates in such a way that launched malware are concealed from the user. This is done by setting up various things to prevent users from noticing the malwares that have been launched.

**Ransom ware:**

Ransom ware can be described as the type of malicious software that prevents and restrict user access to their computer system. This is often done by holding users at ransom after encrypting important information on the users system.

**Rootkit:**

Rootkit is described as a malware that is designed with the sole intent of protecting other malware programs from detection. .

**Spyware:**

Like the name suggests spywares secretly keep tabs on various user activities on their computer systems. They often keep track on passwords and collects sensitive information without permission from a victim's computer. Key-loggers and password gravers are examples of spywares.

**Trojan:**

A Trojan is a type of malicious software that is often disguised as a legitimate software. Upon downloading it installs its self on to the system and damages files and creates various backdoors for unauthorized access to the computer.

**Virus:**

A virus can be described as malicious software that can propagate itself from device to device.

**Worm:**

A type of virus that exploits a systems vulnerabilities and spreads throughout the system by self-replication. The mere existence of a worm on your computer is enough to cause damage. This is because worms do not need to attached themselves to any program to cause damage.

**MALWARE ANALYSIS:**

Malware analysis is the process of dissecting malware in order to examine its workings, origin and sometimes author details and the impact that it potentially causes. And the objective of malware analysis is to provide information about the given software binaries. There are two kinds of analysis in general, namely, Static analysis and dynamic analysis. Static analysis is about examining the information in order to provide clues about the code's behaviour without any executions or installations. And on the other hand, dynamic analysis is about examining the code's behaviour by actually executing or installing it in order to check what will it be doing and how it will affect the system.

**STATIC ANALYSIS:**

Static analysis is about examining the information in order to provide clues about the code's behaviour without any executions or installations. Usually, this analysis method checks whether a file is malicious or benign with the information it has about its functionality. This can also be used to implement simple signatures sets. Some of the basic approaches of static analysis are

1. Finding series of strings or characters:

Casting about for the strings in the software construct provides us the information about its functionality. Strings extracted from the binary usually contains file path references of the modified files or executable that are accessed, URLs to which the program accesses, domain names, IP addresses, attack commands, names of Windows dynamic link libraries (DLLs) loaded, registry keys, and so on.

1. Analyse PE file headers and sections:

The PE file headers contain metadata about the file itself.

1. Searching for packed/encrypted code:

Packing and encrypting the software constructs paves a way to make it more difficult to analyse. They usually contain very few strings and higher entropy compared to legitimate programs.

1. Disassembling the program:

Disassemble refers to translating the machine code into an assembly language. It loads up the executable file into a disassembler to discover the things that the program has done so far.

**MACHINE LEARNING APPROACHES:**

Signature based malware detectors works really well on the malwares that are previously discovered by anti-virus industries. But it has some drawbacks over the malwares that are capable to change its state and signatures over some period and also for malwares that are newly created. However, heuristics-based detectors (a detector that detects the virus by examining code for suspicious behaviour) are not advisable since they provided a lot of false positive and false negative values [9]. So the solution to this problem would relied on using heuristic based detection in combination with machine learning approaches which provides us the higher accuracy during malware detection.

Hence, over the past few decades, there was a great increase in the research side of machine learning especially it has provided a great solution to tackle the classification problems involved in malware detection.

**BUILDING THE DETECTOR:**

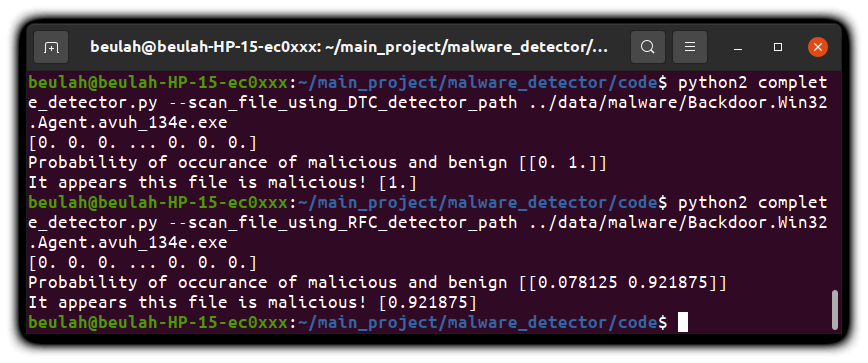
Extracting all the features and applying machine learning algorithms to all of those features will eventually run into memory issues and it may even take very long hour to train the data. This project searches for the series of strings or characters in the software constructs in order to get information and to train the model with it. So the goal is to simply encounter the available unique strings in each of those software binaries using regular expression and storing those features within a dictionary based data structure. It won’t be appropriate for the hardware to deal with thousands and thousands of unique features during the training process. So it is required to compress the extracted features to a vector with some constant dimensions. After features extraction, the vector is ready of the training. Later, passing the feature vector (that is, X) and the label vector (that is, y) into the detector's fit method to train the data and using Python's pickle module to save the detector for future use.

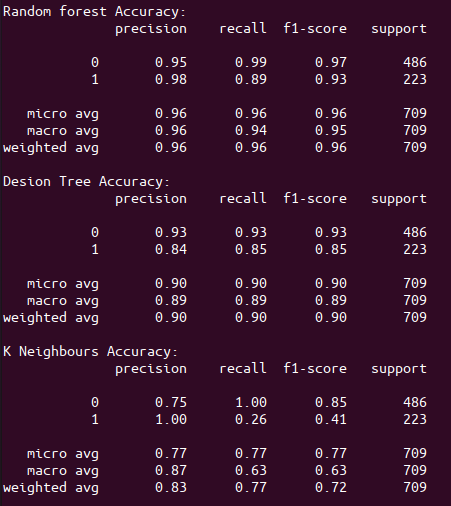
**CONCLUSION:**

In the year of increasing malware and its complexity and its types, its detection and prevention have become the ultimate goal for every anti-virus industries as well cyber security researches. The objective of this project is to detect malicious malware from benign ware and conceived all the challenges in proposing a malware detector and yielded accuracy over 90%. This model can also be fine-tuned by working with some more features for a better learning and accuracy results.

**APPENDIX – A**

**Screen shots:**

****

****

**APPENDIX – B**

**Sample Code:**

import os

import sys

import pickle

import argparse

import re

import numpy

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.feature\_extraction import FeatureHasher

from sklearn import metrics

def get\_string\_features(path, hasher):

# extracting string values from the binary file using regular expressions

chars = r" -~"

print(chars)

minLength = 5

stringRegularExpression = '[%s]{%d,}' % (chars, minLength)

print(stringRegularExpression)

fileObject = open(path)

data = fileObject.read()

pattern = re.compile(stringRegularExpression)

strings = pattern.findall(data)

print(strings)

# storing string features in dictionary form

stringFeatures = {}

for string in strings:

stringFeatures[string] = 1

print(stringFeatures)

# hashing the features

hashedFeatures = hasher.transform([stringFeatures])

print(hashedFeatures)

# data munging -> in order to get the feature array

hashedFeatures = hashedFeatures.todense()

print(hashedFeatures)

hashedFeatures = numpy.asarray(hashedFeatures)

print(hashedFeatures)

hashedFeatures = hashedFeatures[0]

print(hashedFeatures)

# returning hashed string features

print("Extracted {format(len(stringFeatures)} strings from {path}")

print(hashedFeatures)

return hashedFeatures

def scan\_file(path):

# scaning a file to determine whether it is malicious or benign

if not os.path.exists("saved\_detector.pkl"):

print ("Detector haven't trained yet! Train it before scanning the file.")

sys.exit(1)

with open("saved\_detector.pkl") as saved\_detector:

trainedModel, hasher = pickle.load(saved\_detector)

print(trainedModel)

print(hasher)

labels = get\_string\_features(path, hasher)

print("Probability of occurance of malicious and benign", classifier.predict\_proba([labels]))

resultProbability = trainedModel.predict\_proba([labels])[:,1]

# if the user specifies malware\_paths and benignware\_paths, train a detector

if resultProbability > 0.5:

print ("The given file is malicious!",resultProbability)

else:

print ("The given file is benign.",resultProbability)

def get\_training\_data(benign\_path, malicious\_path, hasher):

def get\_training\_paths(directory):

targets = []

for path in os.listdir(directory):

targets.append(os.path.join(directory, path))

return targets

malicious\_paths = get\_training\_paths(malicious\_path)

benign\_paths = get\_training\_paths(benign\_path)

X = [get\_string\_features(path, hasher) for path in malicious\_paths + benign\_paths]

y = [1 for i in range(len(malicious\_paths))] + [0 for i in range(len(benign\_paths))]

return X, y

def train\_detector(benign\_path, malicious\_path, hasher):

# training the detector on the specified training data

def get\_training\_paths(directory):

targets = []

for path in os.listdir(directory):

targets.append(os.path.join(directory, path))

return targets

maliciousDataPaths = get\_training\_paths(malicious\_path)

benignDataPaths = get\_training\_paths(benign\_path)

X = [get\_string\_features(path, hasher) for path in maliciousDataPaths + benignDataPaths]

y = [1 for i in range(len(maliciousDataPaths))] + [0 for i in range(len(benignDataPaths))]

# Random Forest Classifier

RandomForestClassifier = RandomForestClassifier(64)

RandomForestClassifier.fit(X, y)

pickle.dump((RandomForestClassifier, hasher), open("RandomForestClassifier\_saved\_detector.pkl", "w+"))

# Decision Tree Classifier

DecisionTreeClassifier = DecisionTreeClassifier(64)

DecisionTreeClassifier.fit(X, y)

pickle.dump(( DecisionTreeClassifier, hasher), open(" DecisionTreeClassifier\_saved\_detector.pkl", "w+"))

DecisionTreeClassifierData = tree.export\_graphviz(DecisionTreeClassifier, out\_file=None)

graph = graphviz.Source(DecisionTreeClassifierData)

graph.render()

# K Neighbors Classifier

KNClassifier = KNeighborsClassifier(n\_neighbors=3)

KNClassifier.fit(X, y)

pickle.dump((KNClassifier, hasher), open("KNeighborsClassifier\_saved\_detector.pkl", "w+"))

def cv\_evaluate(X,y, hasher):

# use cross-validation to evaluate our model

import random

from sklearn import metrics

from matplotlib import pyplot

from sklearn.model\_selection import KFold

X, y = numpy.array(X), numpy.array(y)

fold\_counter = 0

for train, test in KFold(2,shuffle=True).split(X):

training\_X, training\_y = X[train], y[train]

test\_X, test\_y = X[test], y[test]

RFClassifier = RandomForestClassifier(64)

RFClassifier.fit(training\_X,training\_y)

pred\_y = RFClassifier.predict(test\_X)

print("Random forest Accuracy: ")

print(metrics.classification\_report(test\_y, pred\_y))

DTClassifier = tree.DecisionTreeClassifier()

DTClassifier.fit(training\_X,training\_y)

pred\_y = DTClassifier.predict(test\_X)

print("Desion Tree Accuracy: ")

print(metrics.classification\_report(test\_y, pred\_y))

KNClassifier = KNeighborsClassifier(3)

KNClassifier.fit(training\_X,training\_y)

pred\_y = KNClassifier.predict(test\_X)

print("K Neighbours Accuracy: ")

print(metrics.classification\_report(test\_y, pred\_y))

**REFERENCES**:

[1] https://web.archive.org/web/20061016141708/http://www.viruslist.com/en/viruses/encyclopedia?chapter=153310937

[2] https://www.av-test.org/en/statistics/malware/

[3] https://app.dimensions.ai/discover/publication

[4] Z. Bazrafshan, H. Hashemi, S.M.H. Fard, A. Hamzeh

A survey on heuristic malware detection techniques

The 5th Conference on Information and Knowledge Technology (May 2013), pp. 113-120

https://www.scopus.com/record/display.uri?eid=2-s2.0-84888622633&origin=inward&txGid=85070b2300d704e9f64d06bc1e409273

[5] A. Souri, R. Hosseini

A state-of-the-art survey of malware detection approaches using data mining techniques

Human-centric Computing and Information Sciences, 8 (1) (Jan 2018), p. 3, 10.1186/s13673-\_018-\_0125-\_x

https://www.scopus.com/record/display.uri?eid=2-s2.0-85045289472&origin=inward&txGid=ab0a7198de053be374019e46a657c821

[6] Y. Ye, T. Li, D. Adjeroh, S.S. Iyengar

A survey on malware detection using data mining techniques

ACM Comput. Surv., 50 (3) (Jun 2017), 10.1145/3073559

41:141:40

https://dl.acm.org/doi/10.1145/3073559

[7] A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer

Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-Of-The-Art Survey. Information Security Technical Report 14 (1) (2009), pp. 16-29

https://www.sciencedirect.com/science/article/abs/pii/S1363412709000041

[8] Baldangombo, Usukhbayar & Horng, Shi-Jinn. (2013). A Static Malware Detection System Using Data Mining Methods. International Journal of Artificial Intelligence & Applications.

4.10.5121/ijaia.2013.4411. https://www.researchgate.net/publication/255787076\_A\_Static\_Malware\_Detection\_System\_Using\_Data\_Mining\_Methods

[9] https://ecommons.udayton.edu/maics/2016/Saturday/3/