Opcode Frequency Based Malware Detection Using Hybrid Classifiers

MSc Academic Internship

Cyber Security

Arun Manoharan Kollara

Student ID: X18212204

School of Computing

National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Arun Manoharan Kollara		
Student ID:	18212204		
Programme	MSC CYB…	**Year:**	2019-2020
Module:	Academic Internship		
Supervisor:	Ross Spelman		
Submission Due Date:	17/08/2020		
Project Title:	Opcode Frequency Based Malware Detection Using Hybrid Classifiers		
Word Count:	3464 **Page Count** 16		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:	Arun Manoharan Kollara
Date:	17/08/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Abstract:

The world we live in is the world of technology. Almost every sector in business or organizations makes use of computers for storing information. Some information is private and sensitive to the users. It may include business ideas, government papers, bank account passwords etc. The malicious software is programmed by cyber-criminals to get a hold of this information. This can result in huge losses for the organization or an individual. There are traditional anti-viruses available in the market. However, the complexity and variety of malware are increasing day-by-day. They can bypass the traditional signature-based anti-malware. Research has been focused on Machine learning to find a solution for this advanced malware. There are research conducted to detect a malware executable using opcodes. Opcodes are a part of machine level language that instructs the processor hardware what functions to perform. This thesis makes use of a machine learning-based hybrid algorithm to boost the accuracy of malicious file detection. The thesis developed makes use of opcode-based features.

Introduction

Recent years have contributed largely to technological advancements. Technology has evolved a lot with the growth of the internet. Computer has found a place in the life of every individual. Let it be business, education, banking, etc every department has gone online. People even pay money using a computer-based system. In today's age of technology, cybersecurity is a very serious concept to look after. Apart from companies providing services using the internet. There is also a community that seeks to commit crimes on the internet. Like stealing money or information. They are known as cyber-criminals. Malware is an all-inclusive term. It is composed of various malicious programs like worms, viruses, rootkit, Trojan, etc. Malicious codes are scripts written by hackers with an intent to commit a cyber-crime. Malware can damage a computer system in multiple ways. It can damage files stored on computers. It can crash an OS, corrupt a system, or can even steal sensitive data. There are tech companies that specialize in creating anti-malware systems. Most of these anti-malware systems or antiviruses function on the signature-based concept. Which means it works on a list basis. The incoming file is compared with files it stores in its database. As technology is developing day by day. Cybercriminals are also elevating their skills. These cybercriminals are coming with advanced malware that can obfuscate codes. Which involves register renaming, instruction replacement, etc. The modern malware is capable of tempering the signature of its file using which signature-based anti-viruses can be evaded. Such malware displaying metamorphic abilities consist of metamorphosing engine are termed as metamorphic malware. It is very difficult to detect such kind of malware. This thesis aims of using a machine-learning algorithm to detect malware using opcode-based features. Opcodes are bytes of information that are executed at the machine level. Opcodes are used to initiate hardware instructions. This thesis makes use of opcode-based features on which machine learning will be performed. Ensemble-based Hybrid algorithms will be used for making predictions. Hybrid algorithms are algorithms that consist of multiple models or classifiers working parallelly, in stacked mode, or an iterative order.

Research Question:

- Can we boost the accuracy of malware detection using opcode frequency by leveraging hybrid classification algorithm, and verifying the same by conducting a comparative study of classification algorithms?

Structure of the Document:

Literature Review:

This section will discuss about the literature and different research papers studied by me before developing this thesis.

Research Methodology:

This section will discuss about the concept used for developing this thesis. It will discuss about the algorithms used and different flow designs.

Implementation:

This section will contain the architecture of the thesis developed and will discuss about the proposed system.

Results:

This section will contain the results obtained by the implementation of different algorithms. It will contain a comparison table for proposed system and previous system.

References:

It will contain the references of the literature and different survey papers used by me for developing this thesis.

Literature Review:

This section of the thesis will discuss about the literature studied by me to come up with this thesis. The section will contain a brief discussion of different concepts used in previous literature to detect malware in the system.

Detecting malware is a difficult task to perform. To avoid the malware analysis and detection, more and more malwares adopts measures such as: Shell Code [1], Polymorphism and Metamorphism which make the analysis and detection of malware very difficult. B.Rad and M.Masrom [2] made use of opcode frequency-based histograms to classify viruses. To understand the distance between each executables eucledian distance was used.

LIU Wu, REN Ping, LIU Ke, DUAN Hai-Xin [3] developed a system to identify malware in executable files. The system was completely focused on using the BOS (Behaviour Operation Set) and MBF features to detect the malware. BOS are basic operations used to perform smaller changes in the system. The operations include creating or deleting a file etc.

Four kinds of BOS were defined by Wu et al in their research.

1) Process Action (PA)
2) File Action (FA)
3) Registry Action (RA)
4) Network Action (NA)

They obtain an accuracy of about 97%. The result seems to be good. But the problem with the results is the high number of false positives. These false positives are found to increase with a small increase in the dataset.

In Flow-based Malware Detection Using Convolutional Neural Network author has [4] used dataset of Stratosphere IPS for developing malware detection system. The algorithms used by them were C-NN, Support Vector Machine and Random Forest. A set of 35 features were

used based on the research of [5] for making prediction. The accuracy obtained was 80%. According to the research paper “analysis of Malware Behavior: Type Classification using Machine Learning” [6] the research has also led to high false positive which has been over shadowed by the accuracy.

T.Y. Wang, C.H. Wu and C.C. Hsieh [7] made use of PE to detect malware in executable files using Support Vector Machine algorithm. The overall accuracy obtained was about 98.6%. The problem was that the accuracy for Trojan and backdoor detection were very low.

The system developed by Huicong Loi, Aspen Olmsted [8] was completely focused on detection of backdoors. The main concept used for backdoor detection was to look for any open connections to foreign domains. The DNS of these domain were than checked in the blacklist generated by the system.

In the research paper “Analysis of Malware Behavior: Type Classification using Machine Learning” the author has [6] made use of cuckoo sandboxing for analyzing malware behaviour in exe files. Random Forest algorithm was used as classifier. The accuracy obtained was 98%. The problem with the system was that it could only detect four kinds of attacks. According to the results the number of false positives were also very high.

M. Nar, A. G. Kakisim and N. Çarkac [9] developed a system to detect malware files using control flow graphs which used the concept of pattern matching. A CFG was generated based on the mail program yielding the corresponding annotated CFG for the analysis of malware-based samples. S. Sogukpinar, I. Traore and R. N. Horspool [10] came up with a concept of using sliding window of difference in control flow weight. The system made use of differences in the opcodes distribution. Carkaci et el [11] made use of MAIL structure in-order to map the opcodes to a new feature space in-order to gain the frequency of the opcodes. Iskandar et al. [12] also used control flow graph by adding it with the behavioural feature of the malware to develop an intelligent system for detection of malware.

Runwal et al. [13] developed a system to detect malware based on the graph developed by opcodes. The concept includes constructing a weighted directed opcode graph using sequence of opcodes gathered from executable files.

J. Zhang, Z. Qin and H. Yin [14] made use of Convolution Neural Network to detect malware whose inputs are malware of known binaries. The model detects whether the binary code is malicious.

In [8] the author has conducted a research on the heuristic-based analysis of malwares. Their research concluded that most of the heuristic way detection produced a high false positive rate. For most of the cyber-crime related to data theft to occur the hacker must have control over the internet connection which in-turn can be used by hackers to control the victim systems [15]. They followed blacking of DNS to check for the presence of Backdoor malwares.

Akshay Kapoor and Sunita Dhavale [16] developed a system to detect presence of malware in an executable file with the help of instructions based CFG. Based on the system developed by [17] they made use of n-grams as features which is widely utilized in natural language processing for building a classifier. Feature extraction was done with the help of sliding window. The selection of features was done with the help of Bi-normal separation. Bi-normal separation can be used to measure the importance of features.

$$BNS(f_i) = \left| F^{-1}\left(\frac{C_p}{C_f}\right) - F^{-1}\left(\frac{C_N}{C_{f^{-1}}}\right) \right| \quad (2)$$

where F^{-1} is the inverse cumulative probability function of standard normal distribution.

$\left(\frac{C_p}{C_f}\right)$: True positive rate for a feature.

$\left(\frac{C_N}{C_{f^{-1}}}\right)$: False positive rate for the feature.

The input files were disassembled into multiple instruction which was then organized into CFG. Each node of these CFG contains an instruction. The algorithm used by [16] were Naïve Bayes, SVM and Random Forest.

Research Methodology:

This section of the thesis will discuss about the algorithms used, software and hardware requirements for developing this thesis.

Requirements:

The entire code is developed on Ubuntu 14.04, using Python 2 programming language. We have used Pipenv based Python virtual environment for the development, this helps us to maintain the multiple versions of the same library on the system for project dependencies. Additionally, installing dependent libraries also becomes easy as we maintain the Pipfile which consists of the names of the libraries along with their corresponding version number.

Dataset:

The size of the dataset is about 3000 executable files. 2000 legitimate files were obtained from online free software sources like Source Forge[1], PortableApps[2] and Softsonic[3]. 1000

[1] https://sourceforge.net/
[2] https://portableapps.com/
[3] https://en.softonic.com/

malware containing files are downloaded from VirusShare[4]. All the files in the dataset are 32bit PE files.

Proposed System:
The proposed system will make use of various machine learning algorithms to predict whether a PE file is malicious or not. The system makes use of Hybrid (Voting) algorithm with an intension to boost the overall accuracy of the system. Hybrid algorithms are combination of two or more number of models.

Voting classifier is a classifier which consist of two or more number of classifiers working together. Each classifier operates independently on the dataset and produces a result. Based on majority, the final output is selected.

In my proposed system Voting classifier is composed of (Random forest, Adaboost and xgboost). The output obtained by the voting classifier will be the best of these three algorithms.

So, this system aims at using an ensemble-based hybrid algorithm to produce better results as compared to general machine learning algorithm.

Design

The dataset contains files which are malign and benign. The system developed by us uses opcodes as features for building the model. Objdump command is used for extracting opcodes from the exe files. Objdump command in Linux can be used to dis-assemble an exe file into its opcodes. Then these opcodes are transformed into CSV file for building machine learning model.

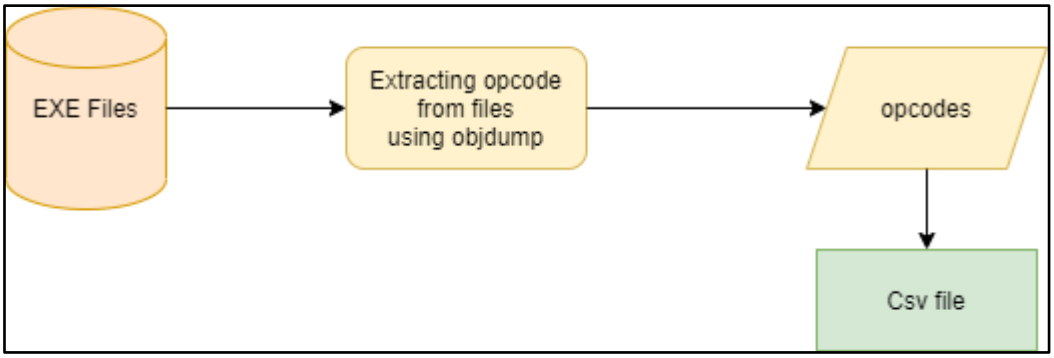

Fig 3.1 CSV File Generation

The CSV file is then divided in the ratio of 8:2. 80% is used for training the model while 20% is used for testing the model. The CSV files consist of opcodes as input independent features and a dependent feature with value ‘0’ and ‘1’ signifying non-malicious and malicious files.

[4] https://virusshare.com/

The training dataset is subjected to pre-processing for removing and null values or garbage data. Then machine learning algorithms are applied to the dataset for training the model.

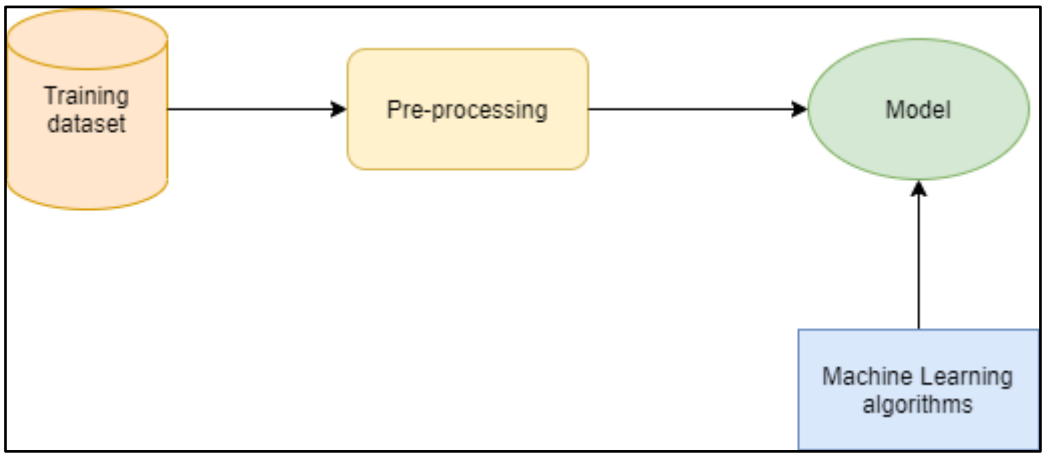

Fig 3.2 Training Phase

Testing data is subjected to pre-processing. The dataset is then fed to the model developed in training phase. Based on the training done the models predicts whether the file is legitimate or not.

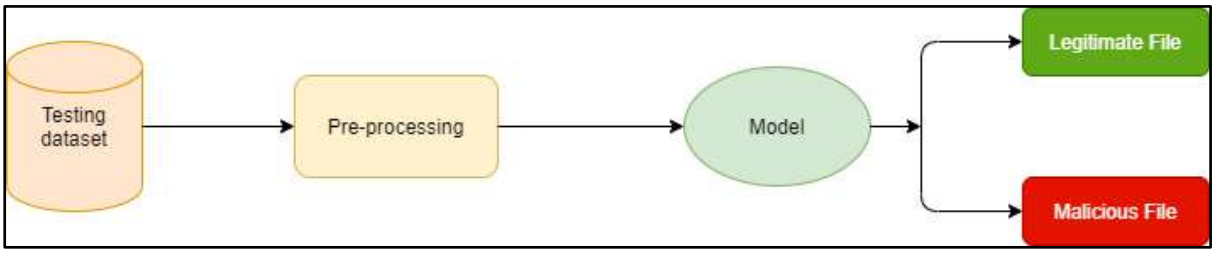

Fig 3.3 Testing Phase

The dataset contains PE files. Using Objdump command the executables are dis-assembled into opcodes. These opcodes are then converted into CSV file. This CSV files are treated as dataset. The dataset is then divided into 8:2 part. 80% is used for training while 20% is used for testing purpose. The training then is then pre-processed. All the null values and garbage values are removed during this process. This pre-processed and clean data is then subjected to machine learning algorithms. The machine learning algorithms used here are (Random Forest, Adaboost, xgboost and Voting Classifier). The 20% testing data is then given to this model for making predictions. Based on its training the model makes prediction whether the given exe file is malware or not.

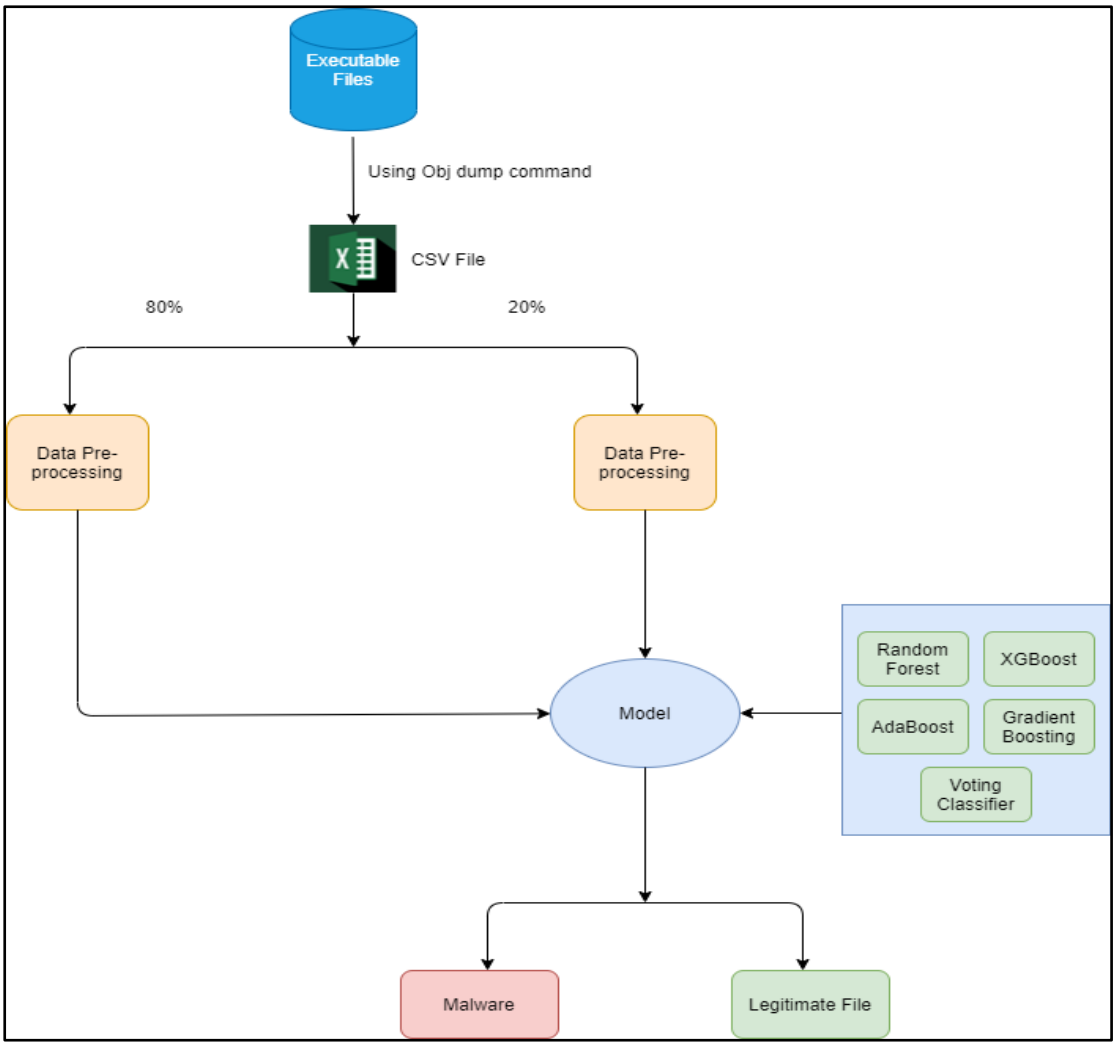

Fig 3.4 Architecture

Implementation

We first start by creating the dataset for malware and benign files then. We used a list of opcodes to scan the Objdump output, and used this as a reference for creating the frequency list for each PE file.

```
opcodes_list = ['mov', 'push', 'call', 'pop', 'cmp', 'jz', 'lea', 'test', 'jmp',
 'add', 'jnz', 'retn', 'xor', 'and', 'bt', 'fdivp', 'fild', 'fstcw', 'imul',
 'int', 'nop', 'pushf', 'rdtsc', 'sbb', 'setb', 'setle', 'shld', 'std', '(bad)']
```

Fig 4.1 Opcode List

Then we scan through dataset folder for PE files, and for each PE file we run Objdump command to dump the opcodes into a file, which is later used for extracting opcodes and creating the frequency list of each opcode present in the binary.

```
for i in os.listdir(PATH)[:1000]:
    row = [0]*(len(headers)-1)
    print i
    file = os.path.join(PATH, i)
    # row[0] = file
    if os.path.isfile(file):
        row[0] = file
        call('objdump -M intel -D ' + file[2:] + ' > hello.txt', shell=True)
        with open('hello.txt', 'r') as f:
            for l in f:
                if ':' in l:
                    # print l
                    for i, opcode in enumerate(opcodes_list):
                        if opcode in l:
                            row[i+1] += 1
        row.append('?')
        with open("check.csv", "ab") as csv_file:
            writer = csv.writer(csv_file, delimiter=',')
            writer.writerow(row)
```

Fig 4.2 Opcode Extraction

Once the features are extracted separately for malware and benign, we load them in our project script using the pandas python library. We then split the total dataset into training (80%) and testing (20%) dataset. Then we train our AdaBoost, RandomForest and XGBoost classifiers that we imported from sklearn python library with training dataset.

```
benign_df = pd.read_csv('./Bening.csv')
malign_df = pd.read_csv('./Malware.csv')

benign_df['res'] = 0
malign_df['res'] = 1

# concatenating dataframe of benign and malware
df = pd.concat([benign_df, malign_df])
df.drop(df.columns[[0, 0]], axis=1, inplace=True)
X = df.drop(['res', 'labels'], axis=1)
y = df['res']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

AdaBoost(X_train, y_train, X_test, y_test)
XGBoost(X_train, y_train, X_test, y_test)
GBoost(X_train, y_train, X_test, y_test)
RandomForest(X_train, y_train, X_test, y_test)

# Voting Classifier
ada = AdaBoostClassifier(n_estimators=400, random_state=42)
rf = RandomForestClassifier(max_depth=70, max_features='auto', n_estimators=400)
xg = XGBClassifier(max_depth=5, n_estimators=100)

Voting(X_train, y_train, X_test, y_test)

if len(sys.argv) > 1:
    filename = sys.argv[1]
    result(filename)
```

Fig 4.3 Project Logic Flow

After training the dataset we define the Voting classifier with the 3 estimator algorithms that we trained, and we have used soft voting which will take the best of 3 results for the prediction. Now, we test our trained Voting hybrid model with the filename that we pass during the command line invocation of the script. We pass the filename to a function called “results” which will perform the testing using the Voting algorithm to predict if it belongs to malware or benign class.

```
def result(filename):
    shutil.copy(filename, "./data/samples")
    opcode_generate.generate_opcode()
    df = pd.read_csv('./check.csv')
    df = df.drop(['name','labels'], axis=1)
    estimators = [('ADA', ada), ('RF', rf), ('XGB', xg)]
    vc = VotingClassifier(estimators, voting='soft')
    vc.fit(X_train, y_train)
    prediction = vc.predict(df)
    print(prediction[len(prediction) - 1])
    for files in os.listdir("./data/samples/"):
        os.remove("./data/samples/"+files)
    if (prediction[len(prediction) - 1] == 0):
        print(filename+" is Bening File")
    else:
        print(filename+" is Malware File")
```

Fig 4.4 Prediction Logic

Evaluation

Introduction:

This section will discuss about the results obtained after the implementation of this thesis. and will analyse and compare the results with other models.

Evaluation Criteria:

Classification Accuracy: It is the ratio of number of correct predictions to the total number of input samples

Precision: It is the number of true positive results divided by the sum of true positive results and false positive results predicted by the classifier

Recall: It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive)

Result Discussion:
From the results obtained we can see that Voting classifier has produced the best results as compared as to the other classifiers with an accuracy of 95.77%.

Below table depicts the output

Algorithm	Accuracy	Precision	Recall	F Measure
AdaBoostClassifier	95	0.94	0.98	0.96
XGBClassifier	93.22	0.93	0.97	0.95
GradientBoostingClassifier	92.33	0.91	0.97	0.94
RandomForestClassifier	95.66	0.92	0.99	0.95
VotingClassifier	95.77	0.95	0.98	0.96

Table 5.1 Prediction Performance

Results

Introduction:

This section will discuss about the results obtained after the implementation of this thesis. and will analyze and compare the results with other models.

Algorithm	Accuracy	Precision	Recall	F Measure
AdaBoostClassifier	95	0.94	0.98	0.96
XGBClassifier	93.22	0.93	0.97	0.95
GradientBoostingClassifier	92.33	0.91	0.97	0.94
RandomForestClassifier	95.66	0.92	0.99	0.95
VotingClassifier	95.77	0.95	0.98	0.96

Table 5.1 Evaluation

Result Discussion:

From the results obtained we can see that Voting classifier has produced the best results as compared as to the other classifiers.

Conclusion & Future Work:

Conclusion:

As per the expectation set earlier in the paper, the thesis came out flying colours in identifying the malicious binaries by extracting opcodes and training different algorithms/models in machine learning. The product has been successfully developed by feeding executable files as input and extracting opcodes from them using Objdump command on Linux platform. These opcodes were treated as features for the machine learning model. From the result section it is evident that Voting classifier which is an ensemble-based hybrid algorithm has produced better results compared to other machine learning algorithms. Hence, we can conclude that a hybrid algorithm increases the accuracy when it comes to detecting malwares using opcode. From the comparative study of algorithms, we have found that ensemble algorithms like random forest and voting classifier are more accurate compared to other algorithms. The evaluation section concludes the accuracy of hybrid algorithm.

Future Work:

In future other hybrid models can be used for producing better results. The system developed only supports files which are of 32-bits. The system has not been tested against stripped PE files, against packed PE files and against PE files with size greater that 35KB. In future these things can be implement using a better dataset. The proposed solution take only one gram into account but with the power of machine learning, Deep Learning algorithms bi gram and tri gram opcode similarity analysis can be done.

References

[1] Y.-s. and . I.-k. Kim, "PE File Header Analysisbased Packed PE File Detection Technique," *IEEE,* 2008.

[2] B. Rad and M. Masrom, "Opcodes istogram for classifying metamorphic portable executable malware," *IEEE,* 2012.

[3] L. Wu, R. Ping, L. Ke and D. Hai-xin, "Behavior-based Malware Analysis and Detection," *IEEE,* 2011.

[4] M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song and C. Park, "Flow-based Malware Detection Using Convolutional Neural Network," *IEEE,* 2018.

[5] G. G. M. Stiborek and J. Zunino, "An empirical comparison of botnet detection methods.” computers & security," *IEEE,* 2014.

[6] R. S. Pirscoveanu, S. S. Hansen, T. M. T. Larsen, M. Stevanovic, J. M. Pedersen and A. Czech, "Analysis of Malware Behavior: Type Classification using Machine Learning," *IEEE,* 2015.

[7] T.-Y. Wang , C.-H. Wu and C.-C. Hsieh , "Detecting Unknown Malicious Executables Using Portable Executable Headers," *IEEE,* 2009.

[8] H. Loi and A. Olmsted, "Low-cost Detection of Backdoor Malware," *IEEE,* 2017.

[9] M. Nar, A. G. Kakisim and N. Çarkac, "Analysis and Comparison of Opcode-based Malware detection approah," *IEEE,* 2018.

[10] S. Sogukpinar, I. Traore and R. N. Horspool, "Sliding window and control flow weight for metamorphic malware detection.," *Springer,* 2015.

[11] N. Carkaci and I. Sogukpinar, "Frequency based metamorphic malware detection.," *IEEE,* 2016.

[12] M. Eskandari and S. Hashemi, "A graph mining appraoch for detecting unknown malwares.," *ResearchGate,* 2012.

[13] N. Runwal and R. M. Low, "Opcode graph similarity and metamorphic detection," *ResearchGate,* 2012.

[14] J. Zhang, Z. Qin and H. Yin, "Malware varient detection using Opcode image recogniton with small training set.," *IEEE,* 2016.

[15] P. Chen, L. Desmet and C. Huygens, ""A Study on Advanced Persistent Threat," *Springer,* 2014.

[16] A. Kapoor and S. Dhavale , "Control Flow Graph Based Multiclass Malware Detection Using Bi-normal Separation," *ResearchGate,* 2016.

[17] Y. Ding, W. Dai and S. Yan, "Control Flow-based Opcode Behavior Analysis for Malware Detection," *ResearchGate,* 2014.