National College of Ireland

Accurate Detection of Malicious Code in PDF Files using Machine Learning.

MSc Academic

Cyber Security

Student Name: Kiran Hassan Shivashankar

Student ID: x18184987

School of Computing

National College of Ireland

Supervisor: Prof. Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kiran Hassan Shivashankar
Student ID:	x18184987
Programme:	MSc in Cyber Security **Year:** 2019- 2020
Module:	Academic Internship
Supervisor:	Prof. Ross Spelman
Submission Due Date:	17/08/2020
Project Title:	Accurate Detection of Malicious Code in PDF Files using Machine Learning.

Word Count: 4916 **Page Count** 15 pages

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: Kiran Hassan Shivashankar

Date: 17 August 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):	

Accurate Detection of Malicious Code in PDF Files using Machine Learning.

Kiran Hassan Shivashankar

18184987

Abstract

The 21st century has been the age of technology. The continuous growth in technology has also forced growth in the field of cybersecurity. Cyber-criminal is continuously researching for new ways to perform cyber-crime. PDF files are very popular in the current age. Most of the official documents to learning materials are now been circulated and read in PDF format. Due to its high flexibility in features PDF format has become very popular. Due to these reasons, cyber-criminals are now been using PDF files to exploit systems and perform cybercrime. PDF file formats are being used by individuals who have no knowledge about computer security. So, they are very vulnerable to any form of PDF-based cyber-attack. This thesis aims at developing a system to detect malicious code in PDF files using machine learning algorithms. The thesis will be performing a comparative study of the algorithm to find the algorithm that gives the best results when it comes to PDF malware detection.

Keywords: PDF, Malware, Adaboost, Logistic Regression, Decision Tree.

1. Introduction

There has been a great boost in the field of IT sector. The technology has evolved enough to share electronic documents. Data is shared across the internet through video files, images, and documents in a large number. PDF (Portable Document Format) is a format used by many people for reading documents. Most of the official documents of organizations are shared in pdf format. Students and college professionals use pdf format of general documentation. PDF format is being used widely around the world due to its wide range of flexible and versatile range of functionality. PDF is being used by many people who may or most of the time are not technical people. People have this wrong understanding that PDF file being just a document and being sponsored by Microsoft must have no security issues.

Due to these features of PDF and its popularity it has become a great tool for hackers to target the users for information and money. Hackers are smart enough to embed scripts in these PDF files which can self-execute as soon as these documents are opened. PDF formatted documents can be used by attackers to perform deadly cyber-attacks ranging from server-side to client-side attacks. It gives a great window or a breach for the attacker to infiltrate targets computer as the applications used by the users are not very up to date applications most of these PDF files executing applications are downloaded from the torrent and hence have no protection. One of the most famous applications to execute PDF files is Adobe Reader [1]. Most people act carelessly and do not download this application from the responsible company websites rather than use to visit torrents to save money at the cost of security. Attackers can exploit the vulnerabilities present in the application and can also make use of other advance features present in Adobe reader document like /Launch using which some specific scripts can be run which will trigger some OS level events. Often JavaScript is being used by the attackers whose sole job is to deviate execution flow to malicious code. Using these attribute of PDF file in mind I have conducted my research in this field and have decided to use machine learning to tackle the problem of PDF file detection rather than going with traditional if-else statements based logics. The dataset was obtained from contagion [2]. Based on the literature review and a tool called PDFID values for a set of features will be extracted. These extracted features are then used for generating model. The set of algorithms is then applied to the extracted feature values and finally, a model is generated. This model will then be used for predicting whether a pdf file is malicious or not.

Can Adaboost algorithm machine learning technique enhance the accuracy of detecting malicious code in pdf files?

PDFiD [3]: Attackers can make use of PDF files for triggering malicious scripts. PDFiD is a tool developed by Didier Stevens in python to analyze PDF files for any malicious content.

PDF structure for a file:

A PDF file is nothing but a collection of objects. The structure defines the hierarchy of these objects, a manner these objects are placed. It defines how these objects are accessed in what manner and in what order. The structure is mainly composed of four parts:

- Header: represent the number of the version used.
- Body: It is the largest part consisting of all the objects and the information that is to be shown.
- (CBT) Cross reference table: It consists of the address of every object in the structure. These addresses are represented as an entry in the table.
- Trailer: The location of CRT and information on the root object are stored here [4].

According to [5] 15 out of 28 desktop PDF viewer application are vulnerable for malicious scripts which are capable of changing the content of digital signature documents.

A research team working with CERT-Bund (Computer Emergency Research Team of Germany) for the sake of research was successful to break the digital signature of the PDF for PDF viewer application. They were also successful in breaking of digital signing on 21 PDF application out of 22 [6].

2. Literature Review

This section will discuss about the literature related to PDF malware detection. We discuss different research papers studied by me for developing this thesis. Their pros and cons will be discussed and finally, a literature review conclusion will be decided.

2.1 Machine Learning

It is a branch of AI which can predict the outcome of a problem by learning its features. Machine learning can learn by observing the data fed to it. It tends to analyze the key features in the dataset which has the highest say in the final output to be achieved. By analyzing these key features and their impact on the output value concerning the input of those features, machine learning tries to build a logic of its own also known as a model. Based on this learning it predicts the output for a input. Machine learning problems can of two types classification or regression [7]. Machine learning has basically two main approaches:

1. Supervised Machine learning algorithm.
2. Unsupervised Machine learning algorithm.

2.1.1 Supervised Machine Learning

Supervised learning involves both the input and output values. The dataset given to machine learning algorithms has values for both input features and the output feature. Machine learning algorithms try to find a relation between the input features and output features. In supervised learning, the algorithms know the output and it must learn the path to reach that output [8].

2.1.2 Un-Supervised Learning

Un-supervised learning consists of only input features and no output features. In un-supervised learning, the algorithms do not know about the final answer or the correct output. The algorithm tries to find an existing pattern within the given input and then reaches a conclusion.

Most of the researches conducted up till now has focused more on the static and dynamic way of PDF malware detection. The research conducted by [7] and [8] presented a method of embedded malware code in word document through static analysis using ngram and introduced novel dynamic run-time test that shows assertion but also remains limited due to the size of malcode. The research showed better results, but the problem was that these kinds of methods were not very suitable for PDF files but were especially focused on other formats like exe, docs. Methods like Heap spraying, AES encryption, and obfuscation can conceal the identity of malware according to the research of [9] and [10]. These exploiting methods are performed using embedded JavaScript code in PDF files. Therefore, rese archers mainly targeted JavaScript code in PDF file.

The system developed by [11] known as SBMDS to detect malware present in PDF files made use of PE features. The system SBMDS works directly on PE files of windows. PE file format is a file format used by all different applications of windows. The majority of malware are found to be in these PE files. An interpretable string format is created for each PE file. Any compressed PE files are decompressed using third party libraries. SVM algorithm along with bagging was used to determine whether a given PE (Portable Executable) file is malicious or not. The system SBMDS works directly on PE files of windows. A model was build using Bagging based SVM algorithm. The size of the dataset used was 9838. Out of which 2,320 PDF files were benign executables. SBMDS achieves high performance in scalability, generalization, efficiency, and effectiveness.

[12]developed a system to detect malware in PDF. An Origami tool was used to extract features from the given PDF file. The job of this too to analyze a PDF file and extracted the value of the given features from them. Features or objects related to JavaScript were extracted using parser used in this model. Features like:

- Eval_length: Malicious scripts can make use of this JavaScript function to dynamically interpret code. Calculate the length of the longest string passed to this function.
- max_string: It can be used to find the length of the longest string present in the file. Generally, malware developed with shell code have longer string size as compared to strings.
- string count: The method can be used for counting the number of strings defined in the script. Usually, the attacker divides the string into multiple parts for obfuscating the malware scripts.
- Replace: replace () is actually a JavaScript function. Which is often used by attackers for obfuscating JavaScript code.
- Substring: Another JavaScript function which is used to manipulate strings in JavaScript.

AdaBoost algorithm was used for generating a model based on the features developed by the parsers. The results obtained by implementation Adaboost algorithms was 87%. The number of false positive and true negative were very high for this research.

[13] Developed a system to detect malware in PDF files. The system developed made use of path features to detect malware. A combination of tags such as Open Action/JS has been used to detect malware. The methods used in the research done by [13] were quite effective they were able to detect malware with good accuracy. The problem with this approach is that it is very difficult to extract the path features as it is not possible to define every structural path manually. This implies that hand-crafted features maybe not working when new types of samples emerge.

The model developed by [14] made use of various JavaScript based features to detect malware in a PDF file. The JavaScript objects, their lexical form, objects, methods, keywords, and attributes were used. A machine learning model was build based on these features. The software was named PJScan. Translation of js code into lexical format was done and tokens from JavaScript were extracted. Their work has significance in deciphering obfuscated parts and an in-depth analysis of the contents of JavaScript. However, their work output has an accuracy of 95%, which is relatively lower than previous methods even in smaller samples.

The [14] researcher designed a malicious pdf detection model against malware attack with contains JavaScript. In this research a feature set was created which consist of structures and metadata like file

size, keywords and encoding methods along with content features such as object name, keywords, and readable strings in JS. Malware detection model was designed using black-box type models with the structure and content features so that the risk of malware attack will be reduced as small changes are robust for machine learning algorithms. For testing, the data set was created consist of benign files consist of multiple JS and for malicious documents the malware was injected. The model was tested on multiple malicious files mainly pdf and it was found that the Random forest machine learning algorithm a group of decision tree algorithm gives a good result.

[15]made use of data mining methods to detect malware and different types of vulnerability in a PDF file. Predictive functionality was used to detect vulnerability. The Proposed system developed was TSEM (Topically Supervised Evolution Model).

[16]made use of SDLC software life cycle to detect PDF based malware. The author also discussed about the cross-site scripting attacks and its use on creating PDF based malware.

[17]discussed about XSS classification. They created a set of features for detecting PDF based malware by extracting only 12 features of pdf files. Naïve Bayes and SVM algorithm were used for making the prediction that had high rate of asymptotic errors.

[18]Developed a model to detect malicious PDF files which makes use of JavaScript to build malicious scripts. This research made use of features like keywords, encoding methods and file size, and other features like object name. Black box type of modelling was used for the development of the system. This reduces the risk of any malware as small changes are robust for machine learning algorithms.

The [19] authors in their research concluded that the main reason for failure of detection of Cross-site scripting attacks are a high range of false positive values. Conducted analyses of different static methods. The researchers used machine learning and pattern recognition techniques to detect malwares in PDF.

In this research, the authors [20] has classified the different inputs and the sinks which may trigger security attacks. Then, the author classified the different sanitization techniques which are commonly utilized to inputs to prevent security problems. Then, they have used data mining techniques to forecast the SQL Injection and Cross site scripting exposures in web applications. Initial researches show that that our proposed attributes are vital signs of such exposures. It has accumulated the fixed code characteristics that describe these categorization programs for every vulnerable fall in a web program. Susceptibility estimate models are then constructed using the accumulated information and the exposure knowledge of each dig. In their initial experiments, these prototypes expected around 85% of SQLI and XSS exposures in various web products.

Author Gurpreet Kaur, Bhavika Pande, Gargi Bhagat and Shashank Gupta in their paper [21] proposed a XSS sanitization mechanism to prevent malicious JavaScript and links to run on HTML5 applications. In this author suggested a defensive mechanism in which proposed architecture will work in two phases, online and offline phase. In offline phase the proposed solution calculates the features extracted from JS code and store it in its repository. Similarly, in online phase the proposed architecture works in two stages. The first stage extract Java Script code from URL and compare it with java script code of webpage and the second stage, compares the features extracted from JS of web with the JS store in repository and if any anomaly found the architecture sanitize the corresponding JavaScript code. Thus, in concluding author states that the proposed solution on virtual HTML5 web application provides security against untrusted JavaScript code and links with very less false positive and negative rate and in the future, they will deploy this model as a Chrome extension on the Cloud Platform. However, all these XSS prevention technologies are for web-based applications, but what if an attack happened using a payload or malicious file the solution is not provided by them.

3. Research Methodology

We conduct a PDF structure analysis of PDF documents downloaded from Contagion Malware Dump and identify the discrepancies between benign and malicious PDF documents. A virtual isolated environment is created to find out how the malicious documents work and how JS is manipulated.

Before applying any algorithm on the dataset or the input PDF file features must be extracted from the PDF file so that they can be processed by the algorithm for generating an output.

In the research conducted by [21] they made use of PDFiD for extracting features from the PDF files. A set of 21 features were selected which can be commonly found in a malicious PDF file according to Cuan et el. Features like /JS indicating the presence of JavaScript in the file. /OpenAction indicating an automatic action to be performed. These features are very suspicious to be found in the PDF files and can be a sign of malicious behavior. PDFiD performs a scan on the input PDF file. The PDF file consists of many features. The job of the PDFiD is to extract the required set of features.

```
PDFiD 0.2.1 CLEAN_PDF_9000_files/rr-07-58.pdf
 PDF Header: %PDF-1.4
 obj                    23
 endobj                 23
 stream                  6
 endstream               6
 xref                    2
 trailer                 2
 startxref               2
 /Page                   4
 /Encrypt                0
 /ObjStm                 0
 /JS                     0
 /JavaScript             0
 /AA                     0
 /OpenAction             0
 /AcroForm               0
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 0
 /EmbeddedFile           0
 /XFA                    0
 /Colors > 2^24          0
```

Figure 1: Output of PDFiD

Further, these features are categorized according to components of JS language and further it is developed into semantic feature set deriving readable strings. On this semantic feature a machine learning model will be decided that will distinguish between malicious and benign pdf file.

The algorithms used for the development of this thesis:

1. AdaBoost
2. Logistic Regression
3. Decision Tree

Adaboost algorithm:

Intuition of AdaBoost algorithm is quite like the random forest algorithm. Random forest is a

collection of trees, each tree acts on its own, makes their own decision based on the dataset give. Similar process takes place in Adaboost algorithm. The key difference between a random forest and Adaboost algorithm is that the result of the random forest consists of the average of all the results obtained by different trees. In the case of AdaBoost algorithm, the decision trees are replaced by stumps. Stumps are trees with only one split, that is one decision to make. The stumps present in Adaboost are of different weights. The result of each stump contributes differently to the final result. The selection or the ordering of features in these stumps is done on basis of their true positive and false positive rate[1].
The total error is used for determining the Amount of Say in the final classification.

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

The total error is just a value obtained by average of False positive and True positive calculated by a stump.

Logistic Regression:

Logistic regression is a supervised form of classification-based algorithm. Logistic regression is used to perform classification which is of binary in nature. The target variable or the dependent variable in the case of logistic regression are dichotomous in nature. The classification is completely binary in nature i.e. the output is in the form of 1 and 0. Mathematically logistic regression predicts the value of the dependent variable "Y" based on the input variable "X". For example, to detect whether a website is phishing or not. These kinds of problems are binary level problems.

Types of classification problems in Logistic regression:

Logistic regression is generally used for solving binary level problems but other types of problems like Multinomial or ordinal level problems can also be solved using logistic regression.
Multinomial: Multinomial type of classification can have more than one type of final classes to be classified in. For example, rather than having a binary classification of only two classes, there can be three classes like 'Type1', 'Type2', 'Type3'[2].

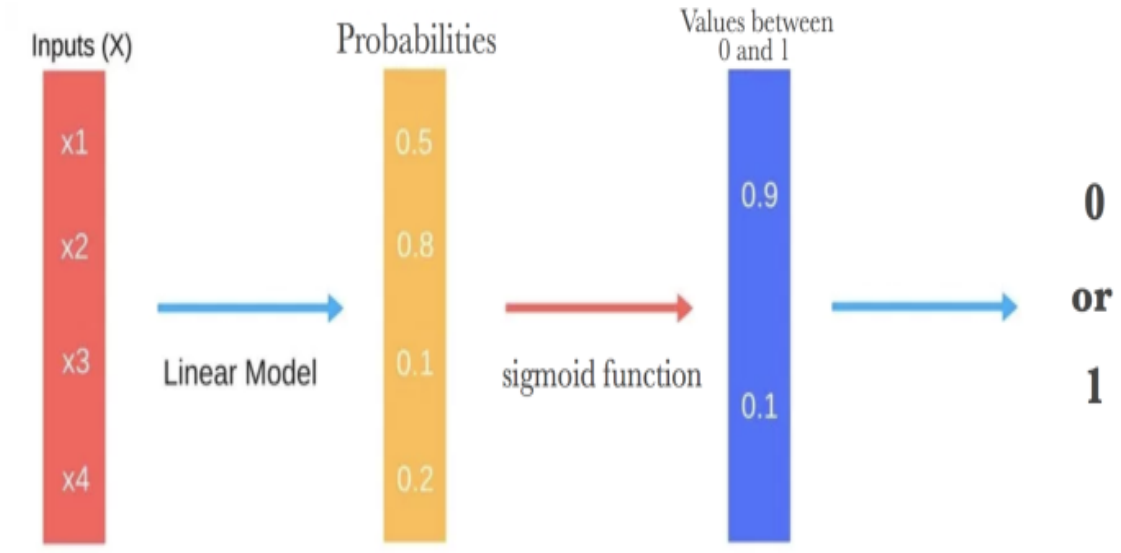

Figure 2: Steps involve in Logistic regression

[1] https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe
[2] https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

Decision Trees:

Decision tree are supervised form of machine learning algorithm. Decision tree as the name suggest is a tree-based algorithm. A decision algorithm has multiple branches. Each leaf node represents a decision to make. Any Boolean function can be represented in a decision tree[3].
In the beginning the entire dataset is considered as root node. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model. Records are distributed based on the measures of attributes in a recursive manner. The attributes are ordered as internal nodes or roots on the basis of the statistical values obtained[4].

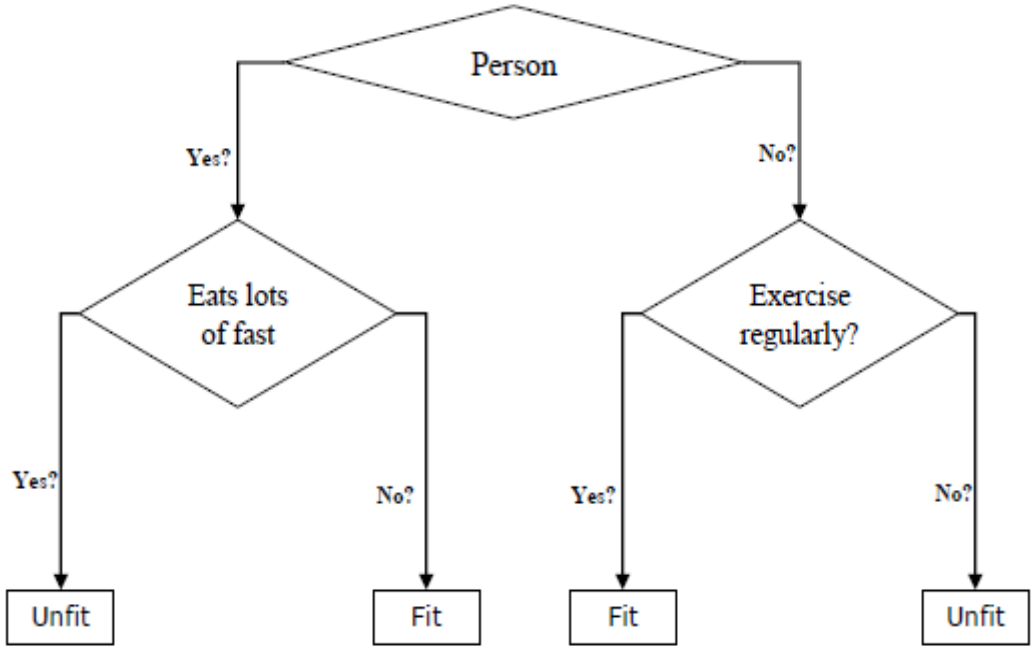

Figure 3: Decision Tree

If we consider the example of a decision tree above, we can understand the working of a decision tree. The classification to be made is whether the person is fit or not. At each node of the tree a decision is made and based on the decision made next criteria is evaluated.

4. Design Specification

This section will consist of detailed discussion of the methods used for developing this thesis. This section will discuss about the algorithms and other functional and non-functional tools required for the development of this thesis.

Many features are considered to characterize the pdf document. The motivation of the project is providing a robust classification algorithm that accurately detect the malicious code in the adversarial samples and base samples. In this approach, we have analyzed and identified the structure and content of the PDF document on the method and tool to analyze. The JavaScript is deciphered to generate the set of features that consist of attributes such as operators, constants, types, and readable strings as well as recorded content. In addition, these extracted features are categorized by JS language components and build them further into a semantical feature set deriving readable strings. A model for machine learning

[3] https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4
[4] https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/

is trained decide on this semantic function to differentiate between malicious and benign pdf file.

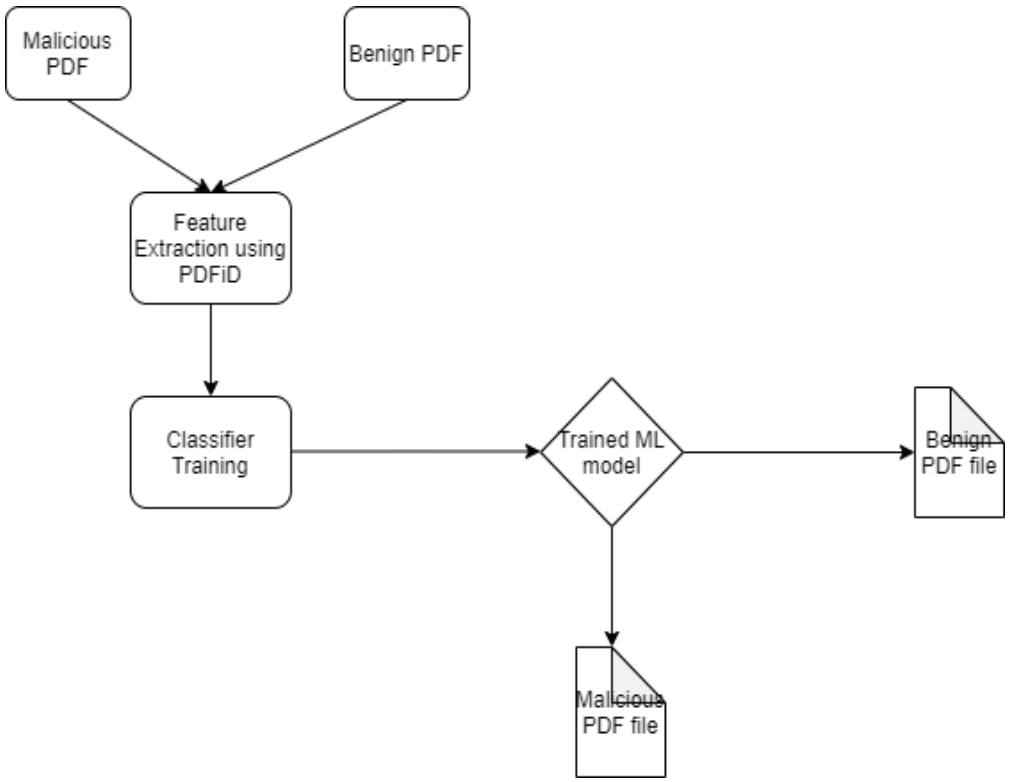

Figure 4: Process Model

Step-1: The dataset of Malicious pdf files and Benign pdf are collected from the Contagio Malware dump.

Step-2: PDFiD tool is used to extract the features of pdf files such as keywords, JavaScript components that sanitizes malicious as well as benign pdf file.

Step-3: A classification model is developed with a robust machine learning algorithm.

Step-4: The classification model is then trained based on the features extracted from Step 2 to distinguish between the pdf files.

Step-5: Then the result we differentiate the malicious and benign file by indicating ✓ for non-malicious pdf file and ✗ for malicious pdf files.

5. Implementation

This section will discuss the implementation of algorithms used for developing this thesis. The section will also describe the architecture and the workflow of the model developed.

Dataset: The dataset for this thesis is obtained from Contagio [22]. The Dataset is divided into two parts. 80% is used for training purposes while 20% used for the testing purpose.

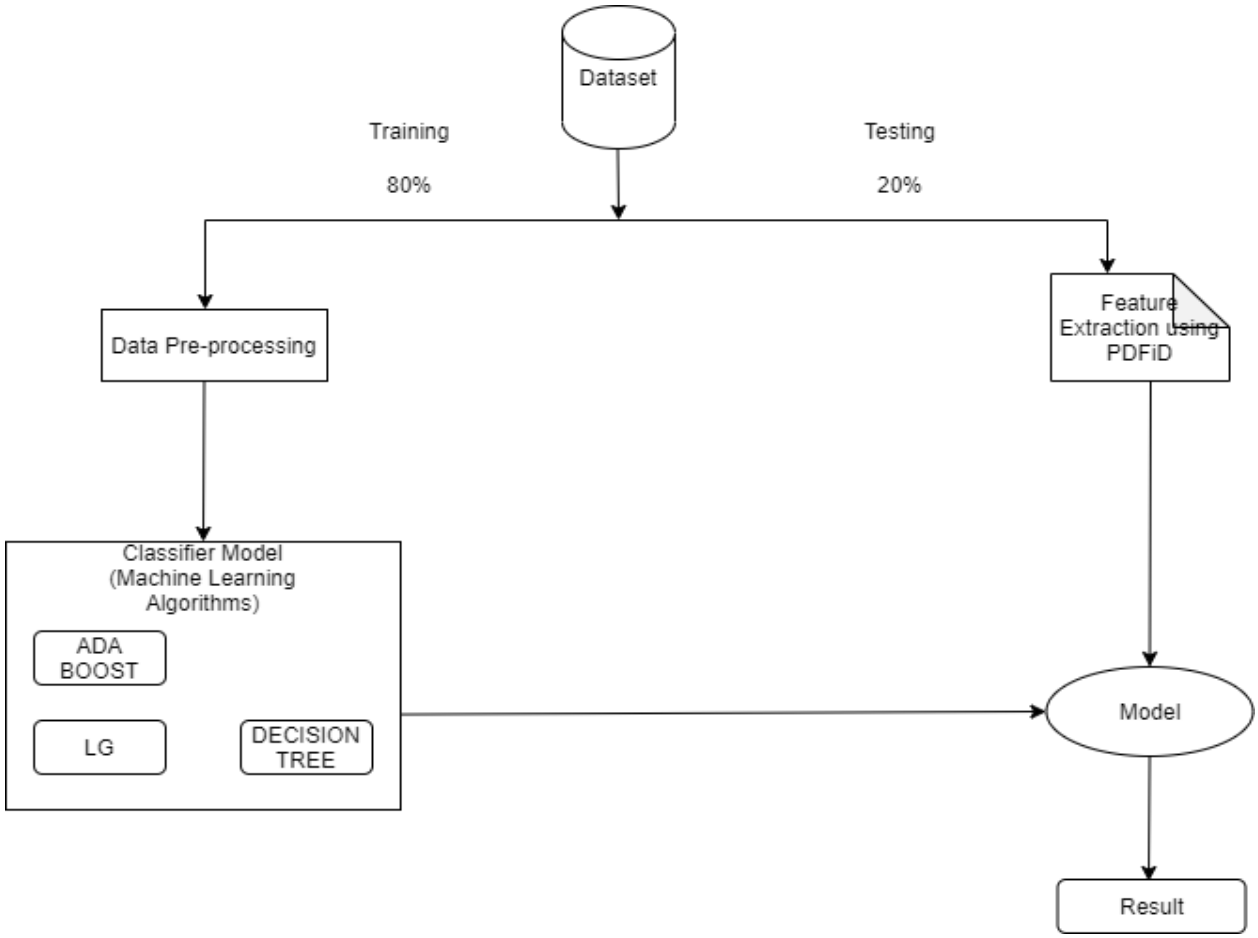

Figure 5:Detection Workflow

Training Phase:

Here the training part of the dataset is used for the generation of the machine learning model. The training dataset is first pre-processed. During pre-processing the garbage values and the NULL values are removed. The un-necessary features are removed from the dataset. After the pre-processing is done. The machine learning algorithms are trained on the dataset. Based on the algorithm and the dataset used a model is developed. This model is then used for making predictions.

```
results = {}
print("\nNow testing algorithms")
for algo in algorithms:
    clf = algorithms[algo]
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    results[algo] = score
    #print("%s : %f %%" % (algo, score*100))
    y_pred = clf.predict(X_train)
    pred=clf.predict_proba(X_test)
    print( algo + ":Confusion Matrix: ", confusion_matrix(y_train, y_pred))
    print ( algo + ":Accuracy : ", accuracy_score(y_train,y_pred)*100)
    print(algo + ":Test set")
    y_pred = clf.predict(X_test)
    print(algo + ":Confusion Matrix: ", confusion_matrix(y_test, y_pred))
    print (algo + ":Accuracy : ", accuracy_score(y_test,y_pred)*100)
```

Figure 6:Code Snippet of training the Classifier model

Testing Phase:

Here the testing dataset is used. The dataset is first pre-processed. These feature values are then sent to the model developed in the training phase. The model then makes a prediction whether the PDF file is malicious or not.

```
col=['obj','endobj','stream','endstream','xref','trailer','startxref','/Page','/Encrypt','/ObjStm',
'/JS','/JavaScript','/AA','/OpenAction','/AcroForm','/JBIG2Decode','/RichMedia','/Colors > 2^24','res']

lst=[]
obj = x['obj']
endobj = x['endobj']
stream = x['stream']
endstream = x['endstream']
xref = x['xref']
trailer = x['trailer']
startxref = x['startxref']
page = x['/Page']
encrypt = x['/Encrypt']
objstm = x['/ObjStm']
js = x['/JS']
javascript = x['/JavaScript']
aa = x['/AA']
openaction = x['/OpenAction']
acroform = x['/AcroForm']
jbig2decode = x['/JBIG2Decode']
richmedia = x['/RichMedia']
colors_gt_2_24 = x['/Colors > 2^24']
res=0

col=['obj','endobj','stream','endstream','xref','trailer','startxref','/Page','/Encrypt','/ObjStm',
'/JS','/JavaScript','/AA','/OpenAction','/AcroForm','/JBIG2Decode','/RichMedia','/Colors > 2^24','res']
lst.append(obj)
lst.append(endobj)
lst.append(stream)
lst.append(endstream)
```

Figure 7:Code snippet of feature extraction

PDFiD is used for extracting the necessary feature values. A total of 21 features are selected from the benign pdf documents and malicious pdf files as well.

```
algorithms = {

    "LR": LogisticRegression(),

    "DecisionTree": tree.DecisionTreeClassifier(max_depth=10),

    "AdaBoost": ske.AdaBoostClassifier(n_estimators=100),

}
```

Figure 8:Code snippet of the implementation of Machine Learning algorithms

The extracted features is then fed into the classifier model in order to validate the accuracy of detecting the malicious files. Then the outcome of the model that has higher accuracy in detecting is stored into classifaction.pkl file. The model predicts whether the PDF file is benign or malicious file based on the result stored is the classification.pkl file.

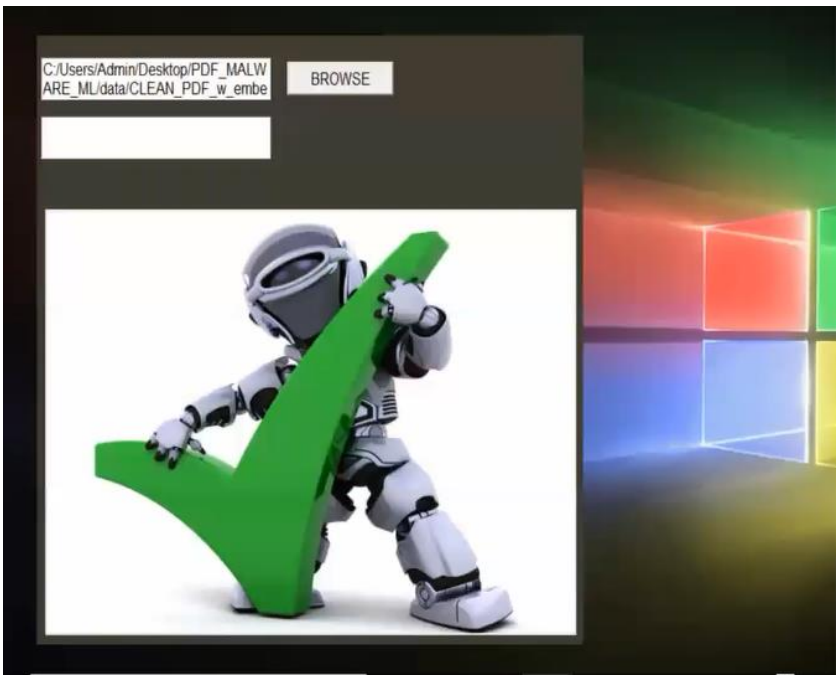

Figure 9: Snapshot of the output of the proposed prototype

6. Evaluation

This section will discuss the results obtained after the implementation of this thesis. The result of each algorithm will be discussed in terms of accuracy, precision, recall, f1-score.

Performance Measure:

Accuracy: Accuracy denotes how well the system has made the prediction. It is denoted by the Number of correct classifications divided by the total number of classifications[5].

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Confusion Matrix:

The confusion matrix is actually a representation of all the classification done by the model in terms of measures. It represents the actual state of the model[6].

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

Confusion Matrix

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision and recall values[7]

Results obtained by the algorithms:

We first extracted the structural and content features from the pdf files. Precision Recall and F-measure were evaluated in this class. 108 benign pdf files and 120 malicious pdf files that included JavaScript were taken from Contagio Malware Dump out which 25 benign pdf files and 21 malicious files were used for testing and training the model.

[5] https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/
[6] https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
[7] https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

Algorithm	Class	Precision	Recall	F-Measure
ADA Boost	Benign	0.96	1	0.98
	Malware	1	0.95	0.98
Logistic Regression	Benign	0.96	0.92	0.94
	Malware	0.91	0.95	0.93
Decision Tree	Benign	1	0.92	0.96
	Malware	0.91	1	0.95

Table 1: Machine Learning Algorithms detection performance

Class	Algorithms	Benign	Malware
Benign	ADA Boost	25	0
	Logistic Regression	23	2
	Decision Tree	23	2
Malware	ADA Boost	1	20
	Logistic Regression	1	20
	Decision Tree	0	21

Table 2: Classification results of base samples and malicious pdf files

We developed model using Logistic Regression that posed an improvement as described in Table1 and Table 2 describes the evaluation results of the classifying benign sample files and the malware samples performed by the training model. ADA Boost algorithm improvised the performance of the model while, Logistic Regression and Decision Tree were still open to adversarial attacks with the accuracy of 93.47% and 95.65% respectively. ADA Boost can be considered robust in an adversarial attack. Despite of using variety of features Decision Tree could not be improved. ADA Boost has the highest accuracy of 97.82% as compared to any other algorithm. Logistic Regression has the least accuracy. As ADA Boost algorithm shows the best results. We can say that that algorithm with a boosting and tree-based approach tends to perform well with this kind of problem, which is detecting malicious code in PDF.

Result comparison:

	Accuracy	F1-Score
Our Research Prototype	97.82%	0.98
[11]	Not mentioned	0.90
[14]	95%	Not mentioned
[12]	87%	Not mentioned

7. Conclusion and Future Work

The main of this research prototype is built using the machine learning approach to identify any existence of malicious code in PDF files. The dataset was divided into training and testing phase. The dataset then goes through pre-processing and feature extraction. This thesis has made use of PDFiD for extracting features from a PDF file. According to the result, ADABoost tends to give the highest accuracy as compared to any other algorithm. The decision tree and random forest algorithm also produced good accuracy. While the accuracy of Naïve Bayes and Logistic regression was lower than others. This shows that tree and boosting based algorithm tends to produce higher accuracy when detecting malware in PDF files. By performing the comparative study of algorithms in this thesis, this literature is successful in finding the types of machine

learning algorithm which can boost the accuracy of malware detection in PDF. The system developed also produces the best accuracy as compare to the literature discussed according to the result section of this thesis.

As this model has shown that boosting and tree-based approach tends to perform well when detecting malware, in future a hybrid form of learning can be used to improve the detection of malicious code in PDF files. The hybrid form of leaning can be a neural network or stacking approach that can consist of a combination of both tree and bagging based algorithms.

References

[1] P. Stokes, "Malicious PDFs | Revealing the Techniques Behind the Attacks," 2019. [Online]. Available: https://www.sentinelone.com/blog/malicious-pdfs-revealing-techniques-behind-attacks/.

[2] "contagio," 2013. [Online]. Available: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html.

[3] "didierstevens," 2020. [Online]. Available: https://blog.didierstevens.com/programs/pdf-tools/.

[4] "pdfid Package Description," 2020. [Online]. Available: https://tools.kali.org/forensics/pdfid.

[5] C. Cimpanu , "New 'Shadow Attack' can replace content in digitally signed PDF files," 2020. [Online]. Available: https://www.zdnet.com/article/new-shadow-attack-can-replace-content-in-digitally-signed-pdf-files/.

[6] C. Cimpanu, "Researchers break digital signatures for most desktop PDF viewers desktop PDF viewers," 2019. [Online]. Available: https://www.zdnet.com/article/researchers-break-digital-signatures-for-most-desktop-pdf-viewers/.

[7] L. W. Stolfo, S. Stavrou, A. Androulaki and K. , "A study of malcodebearing documents," 2007.

[8] S. Khayam and F. M, "Embedded malware detection using markov ngrams," 2008.

[9] "PDF reference, adobe portable document format version 1.7," 2006. [Online].

[10] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious pdf files detection," 2012.

[11] Y. Yanfang, L. Chen, W. Dingding and T. Li, "SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging," *Springer,* 2019.

[12] M. Mahajan, A. Agarwal and S. Dabral, "Malicious PDF Files Detection Using Structural and Javascript Based Features," *IEEE,* 2017.

[13] I. Corona, D. Maiorca, D. Ariu and G. Giacinto, "Detection of malicious pdf-embedded javascript code through discriminant analysis of api references," 2014.

[14] N. Šrndic and P. Laskov, "Detection of malicious pdf files based on hierarchical document structure," 2013.

[15] M. A. Williams, R. C. Barranco and N. M. Sheikh, "A vulnerability analysis and prediction framework," *IEEE,* 2020.

[16] S. N. Bukhari and M. A. Dar, "Reducing attack surface corresponding to Type 1 cross site scripting attacks using secure development life cycle practices," *IEEE,* 2018.

[17] A. E. Nunan and S. Eduardo, ""Automatic classification of cross-site scripting in web pages using document-based and URL-based features," *IEEE*, 2012.

[18] A. Reum Kang, "Malicious PDF Detection Model against Adversarial," *IEEE*, 2019.

[19] A. W. Marashdiha, ""Web Application Security: An Investigation on Static Analysis with other Algorithm to detect Cross- Site Scripting," *IEEE*, 2019.

[20] L. K. Shar and H. B. K. Tan, "Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization pattern," *IEEE*, 2013.

[21] G. Kaur, B. Pande, G. Bhagat and S. Gupta, "Defense Against HTML5 XSS Attack Vectors: A Nested Context-Aware Sanitization Technique," in *IEEE*, Noida, India, 2018.

[22] B. Cuan, A. Damien, C. Delaplace and M. Valois, "Malware Detection in PDF Files using Machine Learning," *scitepress*, 2018.