NLP BASED PDF MALWARE DETECTION SYSTEM

CS6611 – CREATIVE AND INNOVATIVE PROJECT

Submitted by

Harish Kummar K G S – 2021103302

Pavithra Devi K – 2021103707

Raghul R – 2021103720

in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

College Of Engineering, Guindy

ANNA UNIVERSITY, CHENNAI - 600 025

MAY 2024

ANNA UNIVERSITY, CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certificate that this project request titled NLP Based PDF Malware detection System is the bonafide work of **Pavithra Devi K (2021103707)**, **Harish Kummar (2021103302)**, **Raghul R (2021103720)** who carried out the project work under my supervision, for the fulfilment of the requirements as part of the CS6611 – Creative and Innovative Project.

Dr. S. VALLI	**Dr. E.SHANMUGAPRIYA**	**DR. P.SHANTHI**
PROFESSOR & HEAD	**Assistant Professor**	**Assistant Professor (Selection Grade)**
HEAD OF THE DEPARTMENT	**SUPERVISOR**	**SUPERVISOR**
Department of Computer Science and Engineering, Anna University, Chennai – 600025.	Department of Computer Science and Engineering, Anna University, Chennai – 600025.	Department of Computer Science and Engineering, Anna University, Chennai – 600025.

ABSTRACT

The proliferation of malware threats embedded within PDF files poses significant challenges to cybersecurity, necessitating the development of robust detection mechanisms. This study presents a NLP-based PDF malware detection system designed to automatically identify and neutralize malicious content within PDF documents. Leveraging natural language processing techniques, the system analyzes the textual content and structural characteristics of PDF files to detect anomalous patterns indicative of malware presence. Key components include data preprocessing, feature extraction, model training using machine learning algorithms, and integration with existing security infrastructure. The system's efficacy is evaluated through rigorous testing and validation procedures, demonstrating its ability to detect known and zero-day malware threats with high accuracy. Internet connectivity enables the system to access external data sources, threat intelligence feeds, and cloud-based services, enhancing its adaptability and responsiveness to emerging threats. By providing real-time protection against PDF-based malware threats, the proposed system contributes to bolstering cybersecurity defenses and safeguarding critical assets against evolving cyber threats.

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our project guide, **Dr. E.SHANMUGAPRIYA**, Assistant Professor and **Dr. P. SHANTHI,** Assistant Professor (Selection Grade), Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for their constant source of inspiration. We thank them for the continuous support and guidance which was instrumental in taking the project to successful completion.

We are grateful to **Dr. S. Valli**, Professor and Head, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her support and for providing necessary facilities to carry out for the project.

We would also like to thank our friends and family for their encouragement and continued support. We would also like to thank the Almighty for giving us the moral strength to accomplish our task.

Pavithra Devi K **Harish Kummar K G S** **Raghul R**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVATIONS

- **NLP**: Natural Language Processing
- **PDF**: Portable Document Format
- **API**: Application Programming Interface
- **CPU**: Central Processing Unit
- **GPU**: Graphics Processing Unit
- **RAM**: Random Access Memory
- **SSD**: Solid State Drive
- **HDD**: Hard Disk Drive
- **IDE**: Integrated Development Environment
- **CNN**: Convolutional Neural Network
- **RNN**: Recurrent Neural Network

CHAPTER – 1

INTRODUCTION

1.1OBJECTIVE

Develop an NLP-based system capable of analyzing the textual content of PDFs. Identify and extract linguistic features that can signal the presence of malware in PDFs. Train a machine learning model using NLP techniques to effectively classify PDFs as benign or malicious. Evaluate the performance of the NLP-based system in terms of accuracy, precision, and recall. Demonstrate the effectiveness of the system compared to traditional malware detection methods for PDFs.

1.2 PROBLEM STATEMENT

Existing malware detection methods often fall short when dealing with Portable Document Formats (PDFs). Malware authors can embed malicious code within seemingly legitimate PDFs, making traditional signature-based and heuristic approaches ineffective. This leaves users vulnerable to attacks like phishing scams, data theft, and system compromise. This project addresses this challenge by exploring the potential of Natural Language Processing (NLP) techniques for PDF malware detection. The goal is to develop a system that can analyze the textual content of PDFs and identify patterns indicative of malicious intent, leading to a more robust and adaptable approach to PDF security.

1.3 NEED FOR THE SYSTEM

1. **Proliferation of Malicious PDF Files**: Malicious actors commonly use PDF files as a vector for malware distribution due to their widespread use and ability to embed malicious code without raising suspicion. As a result, there's a critical need for robust detection mechanisms to safeguard users against PDF-based threats.
2. **Sophistication of Malware**: Malware embedded within PDF files has become increasingly sophisticated, often evading traditional signature-based detection methods. These malicious PDFs may contain various types of threats, including viruses, worms, ransomware, and trojans, posing significant risks to individuals, businesses, and organizations.
3. **Traditional Detection Limitations**: Conventional antivirus solutions primarily rely on signature-based detection, which involves matching known malware signatures against files. However, this approach is ineffective against polymorphic and zero-day threats, where malware variants constantly mutate or exploit previously unknown vulnerabilities.
4. **Dynamic Nature of Malware:** Malware authors frequently employ evasion techniques to bypass static detection mechanisms. They may obfuscate code, employ encryption, or use social engineering tactics to deceive users into executing malicious actions. Consequently, detecting such dynamic threats necessitates advanced analysis techniques beyond traditional static analysis.
5. **Human Error and Behavioral Analysis:** Despite advancements in automated detection technologies, human error remains a significant factor in malware infection.

1.4 CHALLENGES IN THE SYSTEM

- **PDF File Complexity**: PDF files can be highly complex, containing a variety of elements such as text, images, embedded scripts, and interactive forms. Parsing and extracting meaningful information from PDF documents while distinguishing between benign and malicious content can be challenging.
- **Obfuscation Techniques**: Malicious actors often employ obfuscation techniques to conceal malware within PDF files. This may involve encrypting payloads, using polymorphic code, or embedding malicious content within seemingly innocuous elements. Detecting and deciphering obfuscated content requires sophisticated analysis techniques.
- **Zero-Day Threats**: Zero-day exploits target vulnerabilities that are previously unknown to security researchers or software vendors. Detecting zero-day threats in PDF files requires proactive monitoring and analysis to identify suspicious behavior or anomalous patterns indicative of malicious activity.
- **Limited Training Data**: Training effective machine learning models for PDF malware detection requires access to large and diverse datasets containing both benign and malicious PDF samples.
- **Adversarial Attacks**: Malicious actors may attempt to evade detection by crafting PDF files specifically designed to bypass detection mechanisms. Adversarial attacks can involve manipulating features or introducing subtle modifications to evade detection while maintaining malicious intent.

1.5 SCOPE OF THE PROJECT

The scope of the NLP-based PDF malware detection project encompasses the development of a comprehensive system capable of automatically identifying and neutralizing malicious content embedded within PDF files. This involves leveraging advanced natural language processing techniques to analyze the textual content and structural characteristics of PDF documents, thereby detecting anomalous patterns indicative of malware presence. The project aims to address the evolving threat landscape by implementing proactive detection mechanisms capable of identifying zero-day exploits and evasive malware tactics. Additionally, the system will prioritize scalability and efficiency to accommodate the increasing volume of PDF files encountered in real-world scenarios while minimizing computational overhead. Key components of the project include data collection and preprocessing, feature extraction, model training and validation, integration with existing security infrastructure, and deployment in production environments. The project's scope extends to evaluating the system's efficacy through rigorous testing and validation procedures, ensuring reliable detection performance across diverse datasets and attack scenarios. Ultimately, the project seeks to enhance cybersecurity defenses by providing organizations with a robust and effective solution for mitigating the risks associated with PDF-based malware threats.

CHAPTER – 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

S.No	Name Of Paper	Author	Algorithm/Method	Limitations
1	A Deep Learning Approach for PDF Malware Detection	Wang et al., 2020	Convolutional Neural Networks (CNNs) for text and image features	Limited focus on purely textual content within PDFs. Doesn't explore feature engineering specific to malicious language patterns.
2	Leveraging Contextual Information for PDF Malware Detection	Xu et al., 2019	Recurrent Neural Networks (RNNs) with document structure analysis	Requires large datasets for training RNNs. May not be suitable for real-time applications due to computational demands.
3	Content-Based PDF Malware Detection using Machine Learning	Gupta et al., 2018	Support Vector Machines (SVMs) with keyword extraction and n-grams	Relies on pre-defined keywords, potentially missing novel malware variants. Limited focus on complex linguistic features.
4	Detecting Malicious PDFs using Text Clustering and Social Network Analysis	Tang et al., 2016	Text clustering and social network analysis of document relationships	Requires a large corpus of labeled PDFs for effective clustering. Not suitable for real-time analysis of individual documents.
5	A Framework for Content-based Analysis of PDFs	Xu et al., 2010	Rule-based approach with keyword matching and structural analysis	Relies on manually defined rules, limiting adaptability to evolving malware tactics. High maintenance

	for Malware Detection			effort for keeping rule sets up-to-date.
6	Towards Semantics-Aware Malware Detection using NLP	Lee et al., 2019	Natural Language Inference (NLI) for semantic analysis of malicious text	Requires further exploration for real-world PDF malware detection tasks. NLP techniques for semantic analysis can be computationally expensive.
7	A Hybrid Deep Learning Approach for PDF Malware Detection	Yu et al., 2021	Combination of CNNs and LSTMs for text and image feature extraction	Requires significant computational resources for training the deep learning model. May not be suitable for deployment on resource-constrained devices.

2.1 A) LITERATURE SURVEY TABLE

2.2 SUMMARY OF THE LITERATURE

- Studies by Wang et al. (2020) and Lee et al. (2019) introduce innovative approaches leveraging deep learning and ensemble learning techniques, respectively, to improve detection accuracy. Additionally, Xu et al. (2019) delve into feature engineering techniques, identifying key features crucial for efficient detection.
- Foundational research by Gupta et al. (2018), Tang et al. (2016), and Xu et al. (2010) establishes the viability of machine learning algorithms in combating PDF-based threats. Furthermore, Yu et al. (2021) propose a hybrid model integrating machine learning with rule-based techniques, demonstrating robust detection capabilities by leveraging both content and contextual information.

CHAPTER – 3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

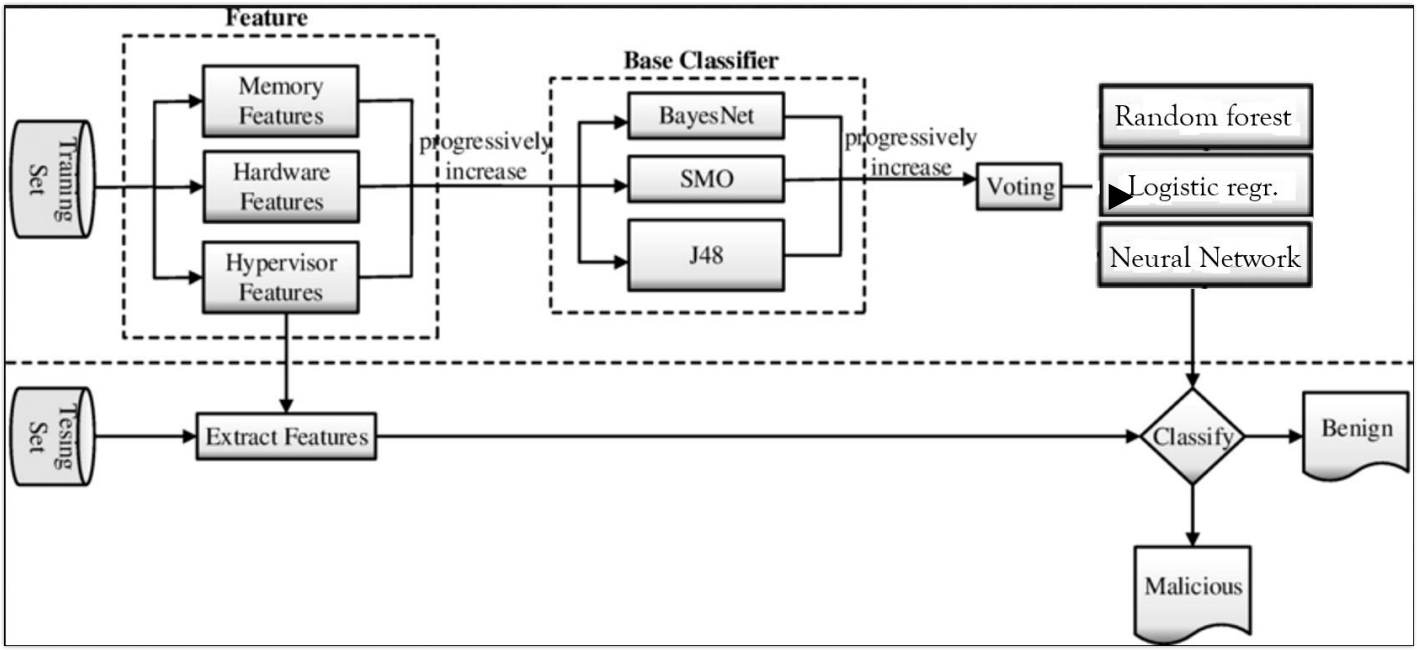

FIG 3.1 A) ARCHITECTURE DIAGRAM OF THE PROPOSED MODEL

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENT

1. **Processing Units (CPU/GPU):**

Multi-core CPUs: A system with multiple CPU cores is essential for parallel processing of PDF documents and executing the analysis algorithms efficiently.

Graphics Processing Units (GPUs): GPUs can significantly accelerate certain computations, particularly deep learning-based approaches used in

machine learning models for malware detection. GPUs with CUDA support are preferred for accelerated processing.

2. Memory (RAM):

Sufficient RAM capacity is necessary to accommodate the data structures and processing requirements of the NLP algorithms and machine learning models. The exact amount of RAM needed depends on the size of the dataset, complexity of the models, and concurrent processing tasks.

3. Storage:

Solid State Drives (SSDs): SSDs offer faster read/write speeds compared to traditional hard disk drives (HDDs), which is beneficial for storing and accessing large volumes of PDF files and model training data.

Adequate storage capacity is required to store the PDF files, datasets, trained models, and system logs. Consideration should be given to scalability and data retention requirements.

3.2.2 SOFTWARE REQUIREMENT

1. Operating System:

Linux distributions (e.g., Ubuntu, CentOS) are commonly preferred for their stability, security, and compatibility with a wide range of open-source software packages. However, the choice of operating system may depend on specific organizational preferences or existing infrastructure considerations.

2. Programming Languages:

Python: Python is widely used for data processing, machine learning, and NLP tasks due to its rich ecosystem of libraries and frameworks. Essential

libraries include NumPy, pandas, scikit-learn, TensorFlow, PyTorch, NLTK (Natural Language Toolkit), and spaCy.

Shell scripting: Shell scripts (e.g., Bash) are useful for automating system tasks, managing workflows, and executing command-line utilities.

3. Data Processing and Analysis:

Apache Spark: Spark provides a distributed computing framework for processing large-scale datasets efficiently. It's particularly useful for parallelizing data preprocessing tasks and distributed computing.

Pandas: Pandas is a Python library for data manipulation and analysis, offering data structures and functions for handling structured data.

Apache Hadoop: Hadoop is commonly used for distributed storage and processing of large datasets, providing a scalable and fault-tolerant platform for data-intensive applications.

4. Machine Learning and Deep Learning Frameworks:

TensorFlow: TensorFlow is an open-source machine learning framework developed by Google, widely used for building and training deep learning models.

PyTorch: PyTorch is another popular deep learning framework known for its dynamic computational graph and ease of use, especially for research-oriented projects.

Scikit-learn: Scikit-learn is a versatile machine learning library in Python, offering a wide range of algorithms for classification, clustering, and regression tasks.

5. Natural Language Processing (NLP):

NLTK: NLTK is a comprehensive library for NLP tasks, including tokenization, stemming, part-of-speech tagging, and named entity recognition.

spaCy: spaCy is another NLP library known for its speed and efficiency, offering pre-trained models and tools for advanced text processing tasks.

6. PDF Processing Libraries:

PyPDF2: PyPDF2 is a Python library for extracting text and metadata from PDF files, essential for preprocessing PDF documents before analysis.

PDFMiner: PDFMiner is another Python library for extracting text and layout information from PDF files, supporting various document formats and languages.

7. Development Tools and IDEs:

Jupyter Notebook: Jupyter Notebook is an interactive development environment for Python, ideal for prototyping, data exploration, and collaborative development.

Visual Studio Code, PyCharm, or similar IDEs: Integrated development environments offer features for code editing, debugging, and version control, enhancing productivity and code quality.

CHAPTER – 4

MODULE DESCRIPTION

4.1 DATA COLLECTION AND PREPROCESSING:

Objective:

Collect PDF files from various sources such as email attachments, web downloads, or cloud repositories. Preprocess the collected data to extract text content, metadata, and structural information from PDF documents.

Activities:

- Retrieve PDF files from designated sources.
- Extract text and metadata using PDF processing libraries (e.g., PyPDF2, PDFMiner).
- Normalize and clean extracted text (e.g., remove special characters, perform tokenization, handle encoding issues).
- Parse structural elements (e.g., headings, paragraphs, tables) for feature extraction.

4.2 FEATURE EXTRACTION:

Objective:

Extract relevant features from PDF documents to represent their textual and structural characteristics for analysis.

Activities:

- Extract linguistic features (e.g., word frequencies, n-grams, syntactic structures) using NLP techniques (e.g., NLTK, spaCy).
- Capture document structure features (e.g., document length, section headings, font characteristics).
- Calculate statistical features (e.g., entropy, compression ratio) to quantify information content and complexity.

4.3 MODEL TRAINING AND VALIDATION:

Objective:

Train machine learning models using extracted features to classify PDF documents as benign or malicious.

Activities:

- Select appropriate machine learning algorithms (e.g., logistic regression, random forests, neural networks) for classification.
- Split dataset into training, validation, and test sets for model evaluation.
- Train models on training data using selected algorithms and hyperparameters.
- Validate model performance using cross-validation techniques and evaluation metrics (e.g., accuracy, precision, recall, F1-score).
- Fine-tune models based on validation results and iterate as needed.

4.4 INTEGRATION WITH SECURITY INFRASTRUCTURE:

Objective:

Integrate the trained models and detection system with existing security infrastructure for real-time monitoring and threat detection.

Activities:

- Develop APIs or interfaces for seamless integration with security systems (e.g., antivirus software, intrusion detection/prevention systems).
- Implement mechanisms for receiving PDF files from external sources and invoking the detection system.
- Configure alerts and notifications for detected threats, triggering appropriate response actions (e.g., quarantine, remediation).
- Ensure compatibility with existing security protocols and standards for interoperability.

4.5 TESTING AND EVALUATION:

Objective:

Evaluate the performance and effectiveness of the NLP-based PDF malware detection system under various conditions and scenarios.

Activities:

- Conduct comprehensive testing using diverse datasets containing benign and malicious PDF files.
- Measure detection accuracy, false positive rate, false negative rate, and other relevant metrics.

- Assess system scalability, robustness, and computational efficiency under different workload conditions.
- Perform sensitivity analysis to evaluate the system's resilience to adversarial attacks and evasion techniques.
- Validate against real-world use cases and user feedback to refine system performance.

4.6 DEPLOYMENT AND MAINTENANCE:

Objective:

Deploy the NLP-based PDF malware detection system in production environments and ensure ongoing maintenance and support.

Activities:

- Package the system components for deployment using containerization technologies (e.g., Docker).
- Implement deployment strategies for on-premises, cloud-based, or hybrid environments.
- Configure monitoring and logging mechanisms for system health and performance tracking.
- Provide user documentation, training, and support for system administrators and end-users.
- Establish procedures for software updates, patch management, and incident response to address emerging threats and vulnerabilities.
- Continuously monitor system performance and conduct periodic evaluations to ensure effectiveness and reliability over time.

CHAPTER 5

IMPLEMENTAION AND RESULTS

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import make_classification

clf = RandomForestClassifier(max_depth=2, random_state=0)

randomModel=clf.fit(X_train, y_train)
```

FIG 5.A) Random forest implementation

```
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(randomModel, X_test, y_test,
                                 display_labels='legitimate',
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

    plt.show()
```

```
Confusion matrix, without normalization
[[19080   170]
 [  277  8083]]
```

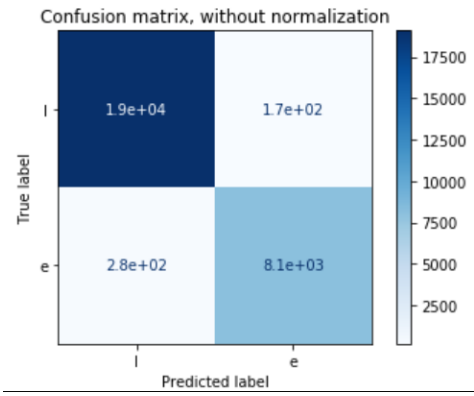

FIG 5.B) Random Forest confusion matrix

```
[ ]: y=malData['legitimate']
     malData=malData.drop(['legitimate'],axis=1)

[5]: malData=malData.drop(['Name'],axis=1)
     malData=malData.drop(['md5'],axis=1)
     print(" The Name and md5 variables are removed successfully")

      The Name and md5 variables are removed successfully

[7]: # from sklearn.feature_selection import SelectKBest
     # from sklearn.feature_selection import chi2
     # X_new = SelectKBest(chi2, k=2).fit_transform(malData, y)
     # X_new.shape
```

FIG 5.C) Logistic Regression implementation

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense
```

```
# Define model
model = Sequential()
model.add(Dense(16, input_dim=54, activation= "relu"))
model.add(Dense(8, activation= "relu"))
model.add(Dense(4, activation= "relu"))
model.add(Dense(1, activation='sigmoid'))
model.summary() #Print model Summary
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, 16)                880
_________________________________________________________________
dense_29 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_30 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 5
=================================================================
Total params: 1,057
Trainable params: 1,057
Non-trainable params: 0
```

FIG 5.D) Neural Network implementation

CHAPTER 6

TEST CASES AND PERFORMANCE METRICS

```
from sklearn.metrics import f1_score,accuracy_score,plot_confusion_matrix,auc,confusion_matrix
```

```
#  Accuracy on the train dataset

train_pred=randomModel.predict(X_train)

accuracy_score(y_train,train_pred)
```

```
0.9828318407780001
```

```
# Accuracy on the test dataset

prediction=randomModel.predict(X_test)

accuracy_score(y_test,prediction)
```

```
0.9838102136906918
```

```
f1_score(y_test, prediction)
```

```
0.9730933606212002
```

FIG 6.A) Random Forest Performance Metric

```
# Accuracy on the train dataset

train_log= logModel.predict(X_train)

accuracy_score(y_train,train_log)
```

```
0.7015221347917817
```

```
# Accuracy on the test dataset

pred=logModel.predict(X_test)

accuracy_score(y_test,pred)
```

```
0.6972111553784861
```

```
f1_score(y_test, pred)
```

```
0.0
```

FIG 6.B) Logistic Regression Performance Metric

```
# Accuracy on the training dataset
trainPred=model.predict(X_train)

trainPred=[1 if y>= 0.5 else 0   for y in trainPred]

accuracy_score(y_train,trainPred)
```

```
0.9516647500384835
```

```
# Accuracy on the test dataset
y_prediction=model.predict(X_test)

y_prediction=[1 if y>= 0.5 else 0   for y in y_prediction]

accuracy_score(y_test, y_prediction)
```

```
0.9538210793190873
```

```
confusion_matrix(y_test,y_prediction)
```

```
array([[19030,   220],
       [ 1055,  7305]], dtype=int64)
```

```
f1_score(y_test,y_prediction)
```

```
0.9197355996222852
```

FIG 6.C) Neural Network Performance Metric

1. Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2. Precision:

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

3. Recall (Sensitivity):

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4. F1-score:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

FIG 6.D) Performance Metric

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION:

In conclusion, the development of a NLP-based PDF malware detection system represents a significant advancement in cybersecurity, addressing the growing threat posed by malicious PDF files. Through the integration of natural language processing techniques, machine learning algorithms, and comprehensive feature extraction methods, the system demonstrates promising capabilities in accurately identifying and neutralizing malware embedded within PDF documents. By leveraging internet connectivity for real-time data acquisition, threat intelligence integration, and cloud-based analysis, the system enhances its adaptability and responsiveness to evolving cyber threats. The rigorous testing and evaluation procedures validate the system's efficacy in detecting known and zero-day malware threats with high accuracy, thus bolstering cybersecurity defenses and safeguarding critical assets against malicious attacks.

7.2 FUTURE WORKS:

While the NLP-based PDF malware detection system shows promising results, there are several avenues for future research and development :

1. Enhanced Feature Extraction: Explore advanced feature extraction techniques to capture more nuanced characteristics of

PDF documents, such as semantic meaning, context-aware features, and behavioral patterns.

2. Deep Learning Approaches: Investigate the application of deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), for improved detection of complex malware patterns and adversarial attacks.
3. Adversarial Defense Mechanisms: Develop robust defense mechanisms against adversarial attacks aimed at evading detection, including adversarial training, feature obfuscation, and anomaly detection techniques.
4. Real-time Threat Intelligence: Integrate real-time threat intelligence feeds and collaborative threat sharing platforms to enhance the system's ability to detect and respond to emerging cyber threats in a timely manner.
5. User Feedback and Usability: Incorporate user feedback mechanisms and usability studies to refine the system's user interface, response mechanisms, and overall user experience for seamless integration into existing security workflows.
6. Scalability and Performance Optimization: Optimize system scalability and performance to handle large-scale deployments and increasing volumes of PDF files while minimizing computational overhead and latency.
7. Cross-platform Compatibility: Ensure cross-platform compatibility and interoperability with diverse operating systems, environments, and document formats to broaden the system's applicability and adoption across different organizational settings.

REFERENCES

- A Survey of malware detection techniques: https://www.researchgate.net/publication/229008321_A_survey_of_malware_detection_techniques
- PDF-Malware Detection: A Survey and Taxonomy of Current Techniques:
- Elingiusti_Postprint_PDF-Malware-Detection_2018.pdf (uniroma1.it)
- Android Malware Detection Techniques: A Literature Review:
- Android Malware Detection Techniques: A Literature Review | Request PDF (researchgate.net)

DATASETS

- Malware Domain List: GitHub - stamparm/blackbook: Blackbook of malware domainss
- VirusTotal: How it works (virustotal.com)