



Computer and System Engineering Department

Harnessing Deep Learning for Evasive PDF Malware Detection

**Graduation project is submitted to Computer and System Engineering
Department in Partial fulfilment of the requirements for the Degree of
Bachelor of Computer and System Engineering**

BY

Shady Esmael Hrajy

Mostafa Mamdouh Sayed

Osama Nasrelden Abdelaleem

Abdallah Abdelmoty Ahmed

Mohamed Ahmed Mohamed

SUPERVISED BY

Prof. Medhat Awadallah & Prof. Gamal Abdelfadel

2024

Breakdown of Individual Contributions:

Student	Contributions
Shady Esmael Hrajy (Team leader) (Communications)	<ul style="list-style-type: none">• Project Management• Web designer (UI/UX)• Embedded systems SW developer
Mostafa Mamdouh Sayed (Communications)	<ul style="list-style-type: none">• Deep Learning
Osama Nasrelden Abdelalem (Computers)	<ul style="list-style-type: none">• Deep Learning
Abdallah Abdelmoty Ahmed (Computers)	<ul style="list-style-type: none">• Web development (Back-end)
Mohamed Ahmed Mohamed (Computers)	<ul style="list-style-type: none">• Web development (Front-end)

Acknowledgment:

First, we would like to thank Allah for the generosity, success, and knowledge He has bestowed upon us, and that without Him we would not have reached where we are now.

We would like also to thank the profs supervising the project **Prof: Medhat Awadallah** and **Prof: Gamal Abdelfadel** for their guidance, support, and knowledge. We were honored to have them supervise our graduation project, and we ask Allah to reward them.

We would also like to thank our friends for their support, opinions that pushed us to develop the project for the better and standing with us throughout the journey. We ask Allah to grant them success in their graduation projects.

We are grateful to the faculty members and staff of the department for their knowledge, wisdom, patience, and kindness. Their dedication and hard work have provided us with an enriching and rewarding academic experience.

Thank you all for your support and encouragement. May Allah bless you all and guide us to the path of righteousness.

Abstract:

The prevalence of sophisticated cyber threats, particularly evasive malware, necessitates the development of advanced security solutions. Our project aims to tackle this issue by integrating artificial intelligence, web development, and embedded systems to create a comprehensive malware detection system. The primary function of this project is to detect evasive malware hidden within PDF files using deep learning techniques.

The project is divided into three main parts:

Artificial Intelligence and Malware Detection: At the heart of our project lies a deep learning model designed to analyze and detect malware within PDF files. Using a dataset comprising both benign and malicious PDFs, the model is trained to recognize subtle patterns indicative of malware. This high-accuracy detection mechanism ensures that even the most evasive malware is identified and flagged. The model employs various techniques, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to achieve optimal performance.

Web Development: To make our malware detection system accessible, we developed a user-friendly web interface. This website allows users to upload PDF files for analysis. Upon submission, the files are processed by our deep learning model, and the results are displayed in real-time. The web interface is designed with a focus on usability and security, ensuring a seamless and safe user experience. It incorporates secure upload protocols and provides detailed feedback on the analysis results, including the type of detected malware, if any.

Embedded Systems for Secure Communication: The final component of our project involves the use of microcontrollers to facilitate secure file transmission. A microcontroller, acting as a transmitter, sends the PDF file to our web-based detection system. After the analysis, the system sends a response back to the transmitter. If the file is deemed safe, it is forwarded to a receiver microcontroller; otherwise, it is deleted. This setup ensures a secure communication link, preventing the transmission of malicious files. The embedded systems are programmed to handle real-time communication and file transfer, providing a robust solution for secure data exchange.

Contents

Acknowledgment:	2
Abstract:	3
List of Figures	9
List of Tables	11
Chapter 1: Introduction	13
1.1.Problem statement:	13
1.2.Examples of some attacks using evasive pdf malware:	15
1.3. Project objective:	16
Chapter 2: Previous and Related Solutions	18
2.1. Currently solutions:	18
2.1.1. Traditional Antivirus Software:.....	18
2.1.2. Heuristic and Behavioral Analysis:	18
2.1.3. Sandboxing:	18
2.1.4. Machine Learning and AI:.....	18
2.1.5. Network Traffic Analysis:	19
2.1.6. Endpoint Detection and Response (EDR):	19
2.1.7. Cloud-Based Security Services:	19
2.1.8. Combined Approaches:	19
2.2. Related work:.....	19
Chapter 3 : Proposed System	23
3.1. Artificial Intelligence Component:.....	24
3.1.1. Data Collection and Preprocessing:.....	24
3.1.2. Model Training:	24
3.1.3. Model Evaluation and Optimization:	24
3.2. Web Development Component:	24
3.2.1. User Interface (UI):.....	24
3.2.2. Backend Integration:.....	24

3.2.3. Real-Time Feedback:.....	25
3.3. Embedded Systems Component:.....	25
3.3.1. Microcontroller Configuration:	25
3.3.2. Secure Communication Protocol:.....	25
3.3.3. Response Handling:	26
3.4. System Integration:.....	26
Chapter 4 : Deep Learning	28
4.1. Abstract:.....	28
4.2. Keywords :.....	29
4.3. Introduction:	29
4.4. Background:.....	31
4.5. Convolutional Neural Networks (CNNs):.....	31
4.5.1. Convolutional Layer:	31
4.5.2. Pooling Layer:.....	33
4.5.3. Fully Connected Layer:	33
4.5.4. Transfer Learning:	34
4.5.5. EfficientNet:.....	36
4.6. Malware Detection Techniques:.....	37
4.6.1.Traditional Malware Detection Techniques	38
4.6.1.Advanced Techniques Using Machine Learning:	39
4.7.Literature Review:	42
4.8.Methods:	49
4.8.1. Data Collection:	49
4.8.2. Data Pre-processing:	51
4.8.3. Model Design and Configuration:	53
4.8.4. Model Training and Validation:	54
4.8.5. Model Evaluation:	55
4.8.6 Results Analysis:.....	58

4.8.7. Model Training and Validation Performance:.....	59
4.9. Detailed Classification Reports:.....	61
4.10. Evaluation and Discussion:	63
4.11. Conclusion:.....	65
4.11.1. Summary of Objectives and Accomplishments:	65
4.11.2. Detailed Findings and Implications:.....	66
4.11.3. Challenges and Future Directions:	67
Chapter 5 : Web Development.....	70
5.1. Abstract:.....	70
5.2. Introduction:	70
5.3. System Overview:	70
5.3.1. User Interaction:	70
5.3.2. Functional Components:.....	70
5.3.3. Non-Functional Requirements:.....	71
5.3.4. System Architecture:	71
5.3.5. Technologies and Tools:.....	71
5.3.6. Testing and Validation:.....	71
5.4. Analysis and Requirements	72
5.4.1. System Requirements	72
5.4.2. Use Cases	73
5.4.3. Use case diagram	75
5.4.4. Sequence diagram	76
5.4.5. ERD Diagram	79
5.4.6. Flow Chart Diagram:	80
5.4.7. System Architecture.....	82
5.4.8. Used Technologies and Tools.....	85
5.4.9. Front-end flow-chart.....	89
5.4.10. Snapshots for the results:	90

5.4.11. Testing.....	91
5.5.Communication with hardware prototype.....	95
5.5.1 Receiving PDF from MCU	95
5.5.2. Preprocessing the PDF File	95
5.5.3. Sending the Response to the MCU.....	95
Chapter 6 : Embedded System.....	97
6.1. Introduction:	97
6.2. Problem Statement:	98
6.3.Proposed Solution:.....	99
6.4.Our System:	101
6.4.1 Hardware Used:	102
6.4.2. Implementation:	103
Chapter 7 : User Interface.....	109
7.1. Key Features:	109
7.1.1.File Upload Mechanism:.....	109
7.1.2.Real-Time Feedback:	109
7.2.Result Display:	109
7.3.User Authentication:.....	109
7.4.Responsive Design:	109
7.5.Design Elements:	110
7.5.1.Clean and Modern Layout:	110
7.5.2.Intuitive Navigation:	110
7.5.3.Accessibility:	110
7.5.4.Security Indicators:	110
7.6.User Workflow:	110
7.7.Snapshots:	111
7.7.1. Logo:	111
7.7.2. Header :	112

7.7.3. Section one of the home page.....	113
7.7.4. Section two of the home page.....	113
7.7.5. Section three of the home page.....	114
7.7.6. Section four of the home page.....	114
7.7.7. Section five of the home page	115
7.7.8. Footer	115
7.7.9. Home dark mode.....	116
7.7.10. Registration page	116
7.7.11. Registration page dark mode	117
7.7.12. Purchasing page	117
7.7.13. Purchasing page dark mode	118
7.8.Tooles and software used:	118
7.8.1. Software:	118
Chapter 8: Conclusion and Future Work.....	120
8.1. Conclusion:	120
8.2. Future Work:.....	121
8.2.1. Deploy the Website on a Cloud Service:.....	121
8.2.2. Testing Various Architectures:	122
8.2.3. Feature Interpretation and Extraction:	122
8.2.4. Multi-Modal Approaches:	122
8.2.5. Continuous Learning and Adaptation:.....	123
8.2.6. Integration with Existing Cybersecurity Systems:	123
References	125

List of Figures

Figure 1 : Evasive Malware Growth	14
Figure 2 : Cybersecurity Market From 2009 To 2015	14
Figure 3 : The malware that hack Linus tech tips channel	15
Figure 4 : system block diagram for safe communication link scenario	23
Figure 5 : System block diagram for web application scenario.....	23
Figure 6 : MCU ESP 8266	25
Figure 7 : A depiction of a convolutional layer in graphic form.	32
Figure 8 : A typical CNN design.....	34
Figure 9 : Inductive transfer visualization.	35
Figure 10 : EfficientNet-B7 performance	37
Figure 11 : The conversion process of evasive PDF malware files into images.....	51
Figure 12 : Examples of benign and malicious images	52
Figure 13 : Confusion Matrix for binary class classification.....	56
Figure 14 : The training loss, accuracy, precision, and recall graphs.	60
Figure 15 : Confusion matrix for the validation set.	61
Figure 16 : Confusion matrix for the test set.	62
Figure 17: Use Case Diagram	75
Figure 18 : User's Register Sequence diagram.....	76
Figure 19 : User's Login Sequence diagram.....	76
Figure 20: User's Payment Sequence diagram.....	77
Figure 21: Upload PDF Sequence diagram.....	78
Figure 22: ERD diagram.....	79
Figure 23: Flow Chart diagram.....	80
Figure 24: System Architecture diagram.....	82
Figure 25: Front-end flow chart.....	89
Figure 26: Upload without login.....	90
Figure 27: Upload with login and the pdf is benign.....	90
Figure 28: Upload with login and the pdf is benign.....	91

Figure 29: Upload with login but out of trials.	91
Figure 30: HTTP protocol.....	92
Figure 31: Embedded system block diagram.....	101
Figure 32: Node MCU esp8266 pin mapping.....	102
Figure 33: HTTP Request.....	103
Figure 34: Decision making.....	105
Figure 35: SPI wiring.....	106
Figure 36: Logo.....	111
Figure 37: Header.....	112
Figure 38: Section one of the home page.....	113
Figure 39: Premium Malero.....	113
Figure 40: Why Malero.....	114
Figure 41: who are we.....	114
Figure 42: Contact us.....	115
Figure 43: Footer.....	115
Figure 44: Home dark mode.....	116
Figure 45: Registration page.....	116
Figure 46: Registration page dark mode.....	117
Figure 47: Purchasing page.....	117
Figure 48: Purchasing page dark mode.....	118
Figure 49: Figma.....	118
Figure 50: Photoshop.....	118
Figure 51: Illustrator.....	118
Figure 52: Flaticon.....	118
Figure 53: Freepic.....	118

List of Tables

Table 1 : Summary of related models' performance.	48
Table 2 : Performance metrics for the best model.....	60
Table 3 : Classification report for the validation data.....	61
Table 4 : Classification report for the test data.....	62
Table 5 : Function Requirements.....	72
Table 6 : Non-Function Requirements.....	72
Table 7 : Testing User Sign-Up	92
Table 8 : Testing user-login.....	93
Table 9 : Testing uploading file.....	93
Table 10 : Testing uploading file.....	94



Chapter (1)

INTRODUCTION

Chapter 1: Introduction

1.1.Problem statement:

The proliferation of malware poses a significant threat to individuals and organizations, with evasive malware presenting a particular challenge. Traditional antivirus solutions often struggle to keep pace with the evolving tactics of cybercriminals, necessitating more sophisticated detection methods. This project aims to address this critical issue by leveraging deep learning techniques to detect evasive malware in PDF files, enhancing cybersecurity measures.

The primary objective is to develop an integrated system that combines AI, web development, and embedded systems to detect and mitigate the threat of evasive malware. The project encompasses three main components: Artificial Intelligence, which uses deep learning algorithms to analyze PDF files for signs of malware; Web Development, which creates a website where users can upload PDF files for malware scanning; and Embedded Systems, which implement a secure communication protocol using microcontrollers. One microcontroller acts as a transmitter, sending the PDF to the website for analysis. Depending on the website's response, the transmitter will either forward the file to a receiver microcontroller or delete it if malware is detected.

The core problem addressed by this project is the need for a reliable and efficient method to detect evasive malware in PDF files. Current detection methods are often inadequate against sophisticated malware that can evade traditional antivirus software, posing significant risks including data breaches, financial losses, and compromised system integrity.

Key challenges include developing a deep learning model capable of accurately identifying evasive malware, ensuring the AI system provides rapid feedback to users without compromising detection accuracy, and designing a reliable protocol for microcontrollers to transmit and receive files securely. Our solution integrates advanced AI techniques with practical web development and embedded systems to create a comprehensive malware detection system. The deep learning model will be trained to detect various forms of evasive malware, providing a high level of accuracy. The user-friendly website will facilitate easy file uploads and deliver prompt analysis results, while the embedded system will ensure secure file transmission, verifying the safety of files before they reach their destination.

By addressing the challenge of evasive malware detection, this project aims to significantly enhance cybersecurity for individuals and organizations. The integration of AI, web development, and embedded systems provides a holistic approach to malware detection and prevention, ensuring safer digital communication and data transfer.

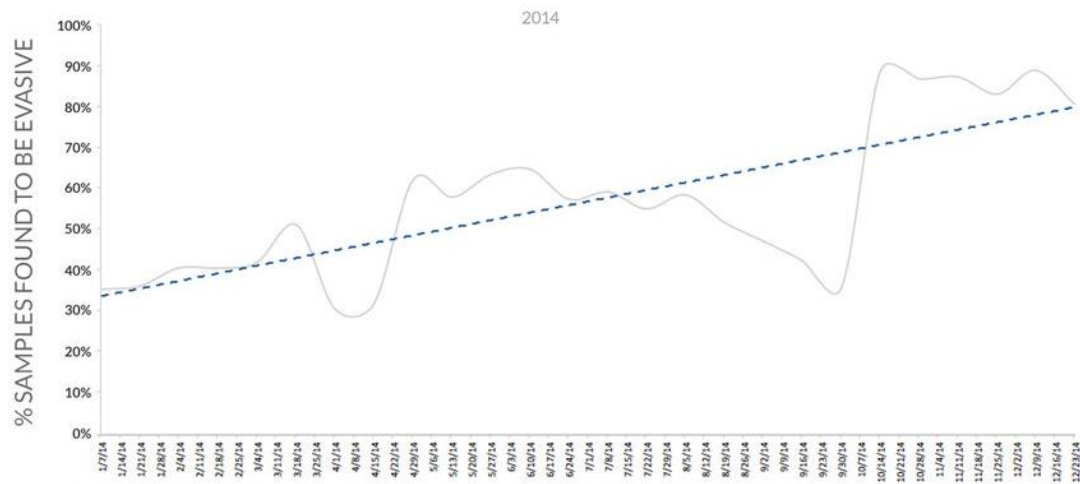


Figure 1 : Evasive Malware Growth

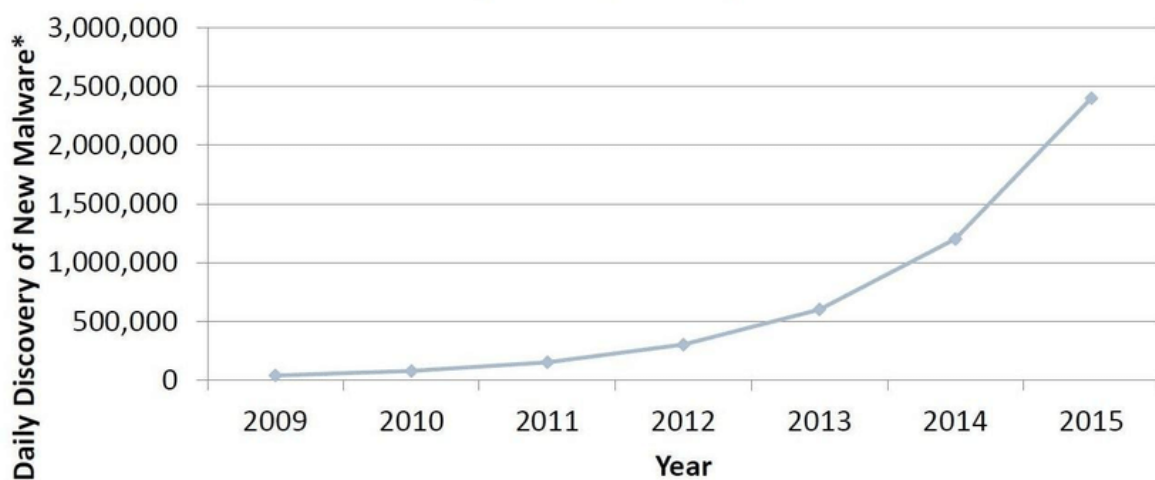


Figure 2 : Cybersecurity Market From 2009 To 2015

1.2.Examples of some attacks using evasive pdf malware:

A vulnerability in Adobe Reader that was exploited by a PDF containing malicious JavaScript to execute arbitrary code on the victim's machine.

-Operation Pawn Storm:

A cyber-espionage campaign that used spear-phishing emails with malicious PDF attachments to target government and military organizations. The PDFs exploited zero-day vulnerabilities to deliver malware.

-2017 Election Phishing Campaign:

Attackers used PDFs to target the campaigns of French presidential candidates. The PDFs contained links to malicious websites that harvested credentials or delivered malware.

-BadPDF:

A technique discovered by researchers where a seemingly benign PDF could be manipulated to download malicious content when opened in certain viewers, exploiting a feature rather than a vulnerability

-Linus Tech Tips Channel:

Linus Tech Tips, a YouTube account channel with over 15 million subscribers, was hacked and ESET was one of the few antivirus vendors to detect the Redline Stealer malware responsible for the attack.



Figure 3 : The malware that hack Linus tech tips channel

1.3. Project objective:

The objective of our graduation project is to design and implement a comprehensive system that enhances cybersecurity by detecting evasive malware in PDF files using advanced deep learning techniques. This system integrates three key components: artificial intelligence, web development, and embedded systems, each playing a crucial role in achieving our goal.

Artificial Intelligence: Our first aim is to develop a robust deep-learning model capable of accurately identifying malware in PDF files. This model will be trained on a diverse dataset of known malware and benign files, ensuring it can effectively distinguish between safe and malicious content. The use of deep learning allows the system to learn complex patterns and improve detection accuracy over time, making it a reliable solution for identifying evasive malware.

Web Development: The second component involves creating a user-friendly website that allows users to upload PDF files for malware scanning. This website will serve as the interface for the AI model, providing real-time analysis and feedback to users. When a user uploads a PDF, the website will process the file using the trained deep learning model and promptly inform the user whether the PDF contains malware. This seamless interaction ensures that users can easily and quickly verify the safety of their documents without needing specialized knowledge.

Embedded Systems: The third component of our project focuses on implementing a secure communication protocol using microcontrollers to ensure safe file transmission. In this setup, one microcontroller will act as a transmitter, responsible for sending the PDF files to the website for analysis. Based on the website's response, the transmitter will either forward the file to a receiver microcontroller if it is deemed safe or delete it if malware is detected. This secure communication link prevents the spread of malicious files and ensures that only safe documents are transmitted, achieving a secure and efficient data transfer system.

By combining these components, our project aims to provide a reliable and efficient method for detecting and mitigating the threat of evasive malware. This integrated approach not only enhances malware detection capabilities but also ensures secure and safe communication of data, significantly improving cybersecurity for users. The project addresses a critical need for advanced malware detection solutions, offering a practical and effective tool to protect against cyber threats.



Chapter (2)

PREVIOUS AND RELATED SOLUTION

Chapter 2: Previous and Related Solutions

2.1. Currently solutions:

The challenge of detecting evasive malware has prompted the development of various cybersecurity solutions, each with its own strengths and limitations. Below are some of the current approaches used to tackle this issue:

2.1.1. Traditional Antivirus Software:

Traditional antivirus software relies on signature-based detection methods, where known malware signatures are compared against files on a system. While effective against known threats, this approach struggles to detect new or modified malware that doesn't match existing signatures. Evasive malware, which is designed to avoid detection by altering its code or behavior, often bypasses these traditional defenses.

2.1.2. Heuristic and Behavioral Analysis:

Heuristic analysis attempts to identify malware by analyzing the behavior of files or programs and looking for suspicious patterns or activities. This method can detect new or unknown malware by identifying potentially harmful behavior. However, it often produces false positives and can be resource-intensive, leading to performance issues on the host system.

2.1.3. Sandboxing:

Sandboxing involves running potentially malicious files in a controlled, isolated environment to observe their behavior without risking the host system. This approach can effectively detect malware by monitoring for malicious activities in a safe setting. However, sophisticated malware can detect when it is being run in a sandbox and alter its behavior to avoid detection, reducing the effectiveness of this method.

2.1.4. Machine Learning and AI:

Recently, machine learning and AI have been employed to enhance malware detection capabilities. These techniques involve training models on large datasets of malware and benign files to identify patterns that distinguish malicious from non-malicious files. AI-based solutions can detect previously unknown malware by recognizing suspicious patterns. However, they require significant computational resources and large, well-labeled datasets to train

effective models. Additionally, adversarial attacks can be designed to fool AI models, necessitating continuous updates and improvements.

2.1.5. Network Traffic Analysis:

Some solutions focus on analyzing network traffic to detect malware communication with command and control servers or other suspicious activities. By monitoring network patterns, these solutions can identify and block malicious traffic. However, they often require complex configurations and can produce a high volume of data, making it challenging to pinpoint specific threats.

2.1.6. Endpoint Detection and Response (EDR):

EDR solutions provide continuous monitoring and analysis of endpoint activities to detect, investigate, and respond to advanced threats. EDR tools can identify suspicious activities and provide detailed insights for remediation. While powerful, they are typically complex to deploy and manage, requiring specialized knowledge and resources.

2.1.7. Cloud-Based Security Services:

Cloud-based security services offer scalable and flexible malware detection solutions, leveraging the power of the cloud to analyze and respond to threats in real-time. These services can offload the processing burden from local systems and provide up-to-date threat intelligence. However, they rely on internet connectivity and may raise concerns about data privacy and compliance.

2.1.8. Combined Approaches:

Many modern solutions combine multiple techniques to enhance malware detection and mitigation. For example, integrating signature-based detection with heuristic analysis and machine learning can provide more comprehensive protection. These hybrid solutions aim to balance the strengths and weaknesses of individual approaches, offering more robust defenses against a wide range of threats.

2.2. Related work:

The application of artificial intelligence (AI) to detect malware, particularly evasive malware in PDF files, is an active area of research. Below are some

significant related works that have explored the use of AI techniques to enhance malware detection capabilities:

1. PDFRate

PDFRate is a machine learning-based tool designed to detect malicious PDF files. It extracts various features from PDFs, such as the number of embedded JavaScript objects, the presence of suspicious keywords, and the structure of the file. These features are then used to train a classifier that can distinguish between benign and malicious PDFs. The use of machine learning allows PDFRate to adapt to new types of attacks, providing a more robust detection mechanism than traditional signature-based methods.

2. MalPDF

MalPDF leverages deep learning techniques to analyze PDF files for malware. This approach involves training a neural network on a large dataset of labeled PDFs to learn the characteristics that distinguish malicious files from benign ones. MalPDF focuses on byte-level analysis, capturing the binary representation of the PDF files to detect even subtle signs of evasion techniques used by advanced malware.

3. PDF Malware Detection Using Convolutional Neural Networks (CNNs)

Researchers have explored the use of convolutional neural networks (CNNs) to detect malware in PDF files. This method involves converting PDF files into image-like representations and using CNNs to identify patterns associated with malicious content. The advantage of CNNs lies in their ability to automatically learn features from the data, making them well-suited for identifying complex patterns in PDFs that may indicate the presence of malware.

4. Deep PDF: A Hybrid Deep Learning Approach

Deep PDF combines multiple deep learning techniques, including CNNs and recurrent neural networks (RNNs), to analyze the structure and content of PDF files. This hybrid approach aims to capture both spatial and sequential features within PDFs, enhancing the model's ability to detect evasive malware. By

leveraging the strengths of different deep learning architectures, Deep PDF can achieve high accuracy in identifying malicious files.

5. Semi-Supervised Learning for PDF Malware Detection

This approach utilizes semi-supervised learning to improve the detection of malware in PDF files. By combining labeled and unlabeled data, the model can learn from a larger dataset, enhancing its ability to generalize and detect new types of malware. This method addresses the challenge of limited labeled data, which is common in the cybersecurity domain, and helps improve detection rates for evasive malware.



CHAPTER (3)

PROPOSED SYSTEM

Chapter 3 : Proposed System

Our proposed system aims to address the challenge of detecting evasive malware in PDF files by integrating advanced deep learning techniques with user-friendly web development and secure embedded systems. The system is divided into three main components: Artificial Intelligence (AI), Web Development, and Embedded Systems, each of which plays a crucial role in achieving our goal.

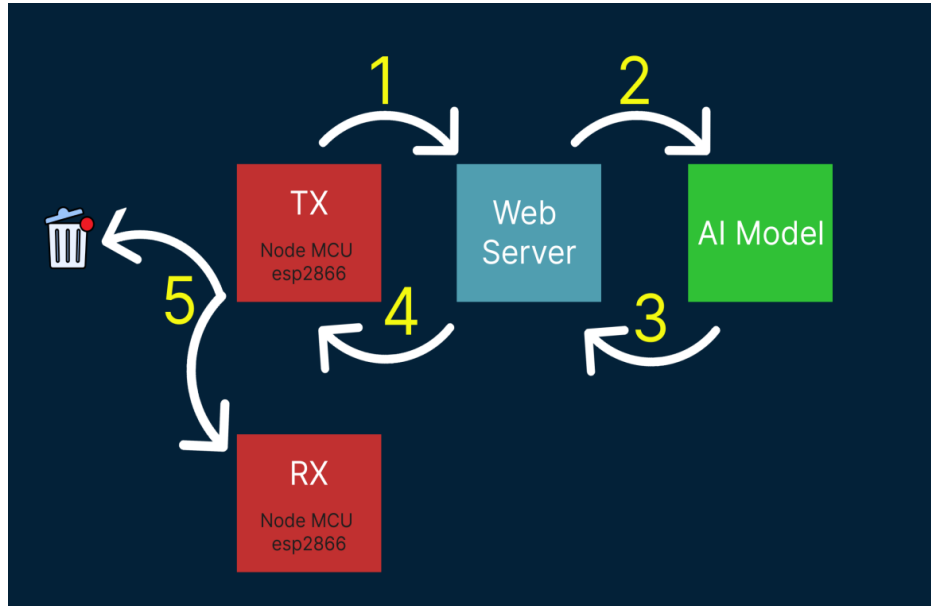


Figure 4 : system block diagram for safe communication link scenario

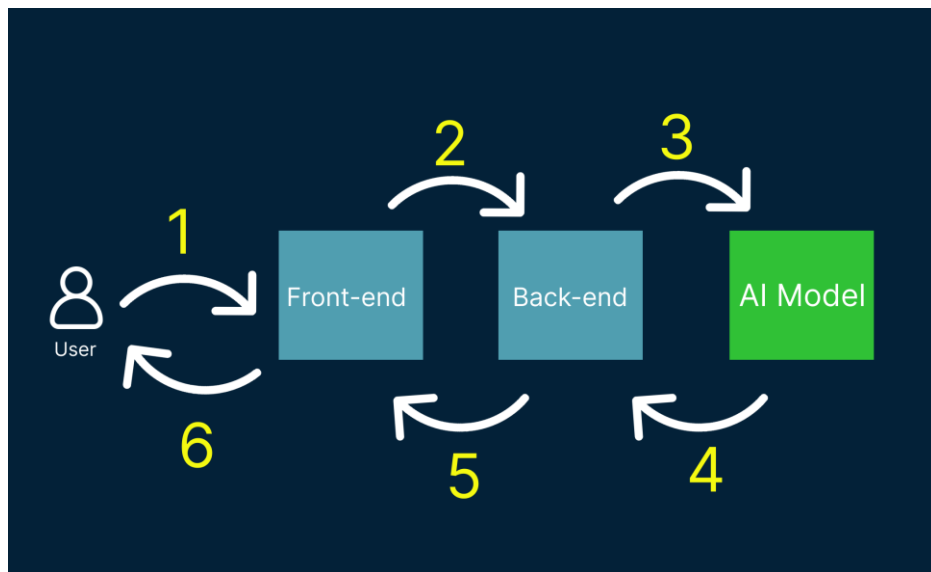


Figure 5 : System block diagram for web application scenario

3.1. Artificial Intelligence Component:

The AI component focuses on developing a robust deep-learning model to identify malware in PDF files. The key features of this component are:

3.1.1. Data Collection and Preprocessing:

We will compile a comprehensive dataset consisting of both benign and malicious PDF files. This dataset will be used to train and evaluate our deep learning model. Preprocessing steps will include feature extraction, such as analyzing the structure of PDF files, embedded scripts, and metadata.

3.1.2. Model Training:

We will employ convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to capture both spatial and sequential features within the PDF files. The model will be trained using supervised learning techniques, where labeled data will be used to teach the model to differentiate between safe and malicious PDFs.

3.1.3. Model Evaluation and Optimization:

The trained model will be evaluated using a separate validation dataset to assess its accuracy and performance. Hyperparameter tuning and optimization techniques will be applied to enhance the model's detection capabilities.

3.2. Web Development Component:

The web development component will provide a user-friendly interface for interacting with the AI model. The key features of this component are:

3.2.1. User Interface (UI):

We will develop a web-based platform that allows users to upload PDF files for malware scanning. The UI will be designed to be intuitive and easy to use, ensuring a seamless experience for users.

3.2.2. Backend Integration:

The backend system will handle the interaction between the user interface and the AI model. When a user uploads a PDF file, the backend will process the file and send it to the AI model for analysis.

3.2.3. Real-Time Feedback:

The website will provide real-time feedback to users, informing them whether their PDF files contain malware. This immediate response is crucial for users to take necessary actions to protect their systems.

3.3. Embedded Systems Component:

NodeMCU ESP 8266 12E is an open-source development board and firmware based on the widely used ESP8266 -12E Wi-Fi module. It allows you to program the ESP8266 Wi-Fi module with the simple and powerful LUA programming language or Arduino IDE.



Figure 6 : MCU ESP 8266

The embedded systems component will ensure secure and efficient file transmission between microcontrollers and the web platform. The key features of this component are:

3.3.1. Microcontroller Configuration:

We will configure two microcontrollers, one as a transmitter and the other as a receiver. The transmitter will send PDF files to the web platform for analysis, while the receiver will handle safe files based on the analysis results.

3.3.2. Secure Communication Protocol:

A secure communication protocol will be implemented to ensure the integrity and confidentiality of the files being transmitted. This protocol will include encryption and authentication mechanisms to protect against unauthorized access and tampering.

3.3.3. Response Handling:

Based on the response from the web platform, the transmitter microcontroller will either forward the file to the receiver if it is safe or delete it if malware is detected. This ensures that only safe files are transmitted, maintaining a secure communication link.

3.4. System Integration:

The integration of these three components will result in a comprehensive system that provides robust malware detection and secure file transmission. The process flow of the system is as follows:

- Users upload PDF files to the web platform through the user interface.
- The web platform processes the uploaded files and sends them to the AI model for analysis.
- The AI model evaluates the files and returns a result indicating whether they are safe or contain malware.
- Based on the analysis result, the web platform provides real-time feedback to the user.
- Simultaneously, the transmitter microcontroller sends PDF files to the web platform and receives analysis results.
- The transmitter microcontroller forwards safe files to the receiver microcontroller or deletes malicious files based on the received results.



CHAPTER (4)

DEEP LEARNING

Chapter 4 : Deep Learning

4.1. Abstract:

This research introduces a pioneering approach to combating evasive PDF malware through the application of advanced deep learning techniques, specifically utilizing EfficientNet, a state-of-the-art convolutional neural network (CNN). Evasive malware, particularly that which exploits the Portable Document Format (PDF), poses a significant challenge in cybersecurity due to its ability to bypass traditional detection systems through methods such as code obfuscation, sophisticated encryption, and the use of anti-analysis tactics. These threats have necessitated the development of more robust and adaptive detection mechanisms capable of addressing the sophisticated tactics employed by modern cyber threats.

In response to these challenges, our study employs a novel methodology that integrates transfer learning with EfficientNet to address the detection of evasive PDF malware. Transfer learning is utilized to leverage the comprehensive, pre-trained data of ImageNet, thereby enhancing the learning process without the exhaustive requirement for a large, domain-specific dataset. This approach is particularly advantageous in the cybersecurity field, where obtaining extensive labeled datasets is often impractical.

The core of our methodology involves the transformation of 31,006 raw PDF files into image data, a process that allows the use of CNNs to detect subtle, malicious patterns embedded within the files. This conversion is instrumental in harnessing the visual processing strength of EfficientNet, which is optimized for high efficiency and scalability across varying scales of data through its compound scaling methodology.

The effectiveness of this innovative approach is demonstrated by the model's performance, achieving an impressive accuracy of 98.87% and an F1-score of 99.20% on the testing dataset. These results not only illustrate the model's high precision and recall but also signify a substantial enhancement over traditional malware detection methods, which rely heavily on manual feature extraction and are thus prone to errors and manipulation by sophisticated malware.

The implications of this research are far-reaching, extending beyond the realm of malware detection to broader cybersecurity applications such as intrusion detection, phishing prevention, and the mitigation of ransomware threats. Our findings set a new precedent in the application of machine learning for cybersecurity, highlighting the potential of CNNs and transfer learning in

developing next-generation cybersecurity solutions that are both effective and efficient.

The study's outcomes contribute significantly to the ongoing discourse in cybersecurity, offering a robust framework for future research and development of advanced malware detection systems. By advancing our understanding of how deep learning can be tailored to detect complex malware, this research moves us closer to developing more secure digital environments resilient to the evolving landscape of cyber threats.

4.2. Keywords :

PDF evasive malware, transfer learning, EfficientNet, deep learning, cybersecurity.

4.3. Introduction:

In today's digital age, the cybersecurity landscape is perpetually shifting, presenting new challenges that demand innovative and dynamic responses. Among the myriad of security threats, evasive PDF malware emerges as a particularly insidious threat. This form of malware exploits the complex features of Portable Document Format (PDF) files—widely used for their versatility and ubiquity—to bypass conventional antivirus and security measures. Evasive PDF malware typically employs sophisticated techniques such as code obfuscation, which involves disguising the code's true purpose to evade detection; anti-analysis tactics, which prevent security analysts from examining the malware; and encryption, which conceals the malware's malicious operations behind seemingly benign code.

Traditionally, malware detection has been anchored in signature-based methods, which depend on identifying known malware patterns or signatures. While effective against well-documented threats, these methods falter against novel or significantly modified malware strains, often missing new mutations of evasive malware embedded within PDF files. As cybercriminals continually refine their tactics, the inherent limitations of traditional approaches have become increasingly apparent, propelling the need for more sophisticated, adaptive detection technologies.

The advent of machine learning, and specifically deep learning, has introduced a paradigm shift in how cybersecurity challenges are approached. Deep Learning, particularly through the use of Convolutional Neural Networks

(CNNs), has demonstrated remarkable success across various fields such as image recognition, natural language processing, and autonomous vehicle systems, attributed to its profound ability to extract and learn complex patterns from voluminous data sets.

In the specific context of cybersecurity, CNNs are leveraged to transform the challenge of detecting evasive malware into a manageable image classification problem. This transformation involves converting malware files into image data, allowing the nuanced and complex characteristics of the malware to be captured visually. This approach permits the identification of malicious patterns that are not typically discernible by traditional, rule-based detection systems.

A significant innovation of this study is the application of EfficientNet, a state-of-the-art convolutional neural network architecture developed by Tan and Le in 2019. EfficientNet distinguishes itself through its novel compound scaling method, which systematically and uniformly scales network width, depth, and resolution, balancing the network's capacity and efficiency, and setting new benchmarks in computational performance. This makes EfficientNet particularly suited for the demanding requirements of processing and analyzing large-scale image data for malware detection.

Furthermore, this research harnesses the power of transfer learning, utilizing the pre-trained capabilities of EfficientNet on the extensive and diverse ImageNet dataset. This strategic choice leverages learned features from a broad array of visual contexts, enhancing the model's ability to generalize from image recognition to malware detection. This method not only improves the efficiency and accuracy of the model but also mitigates the challenges associated with the scarcity of labeled training data in the cybersecurity domain.

The dataset employed in this study comprises 31,006 raw PDF files, meticulously curated to train and validate the detection model. These files are processed into images to fully exploit EfficientNet's capabilities, enabling the nuanced detection of subtle malicious signatures that traditional methods might overlook.

The broader implications of this study are vast, extending the potential applications of this research to include not just malware detection but also broader security measures such as intrusion detection systems, phishing attack prevention, and effective countermeasures against ransomware. The expected outcomes of this research endeavor to propel forward the technological frameworks of malware detection and significantly enhance the resilience of digital infrastructures against advanced cyber threats.

By examining the integration of EfficientNet and deep learning within the sphere of cybersecurity, this study aims to construct a robust analytical model capable of adapting to and countering the evolving landscape of cyber threats. This comprehensive exploration will contribute to pioneering advanced cybersecurity solutions, setting the stage for subsequent sections that will discuss the related work, methodologies, and detailed findings of this research.

4.4. Background:

Modern computer technology makes it possible to move human existence from real-world surroundings into virtual ones. Cybercriminals now concentrate more on virtual life than real life. This is true because it is easier to commit a crime online than it is in real life. Online criminals frequently utilize malicious software, also known as unwanted software or malware, to launch cyberattacks. Malware strains are advancing using sophisticated packing and obfuscation techniques. Malware detection and classification are exceedingly challenging because of these obfuscation techniques.

4.5. Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) are a particular kind of neural network that has successfully been used in a variety of applications, such as the detection of skin cancer, face identification, and objects, as well as image recognition and categorization.

Vision-based malware detection approaches feed the PE malicious code binaries that are converted to an image to a deep CNN for feature extraction and classification. Thus, the general architecture of CNNs will be discussed. Any CNN architecture typically includes an input layer, convolutional layers, hidden layers with Rectified Linear Unit (ReLU) activation function, pooling layers, and fully connected layers. The input layer oversees entering the picture's raw pixel data (image height and width, together with the three-color channels RGB).

4.5.1. Convolutional Layer:

The convolutional layer is crucial to CNN operation. The usage of learnable kernels is the main emphasis of the layer's parameters. These kernels often have a low spatial dimension yet cover the whole depth of the input. Each filter is convolved across the spatial dimensions of the input by the convolutional layer as the data reaches it, creating a 2D activation map. Figure 7 illustrates how to see these activation maps.

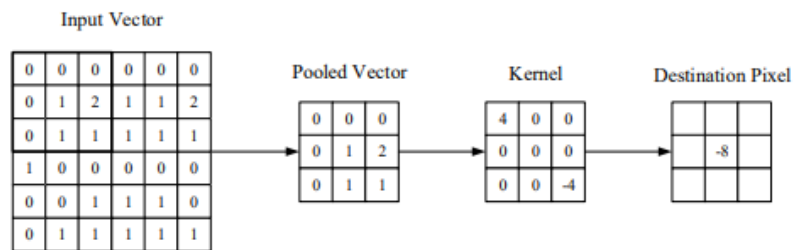


Figure 7 : A depiction of a convolutional layer in graphic form.

The input vector is placed over the kernel's central element, which is then computed and replaced with a weighted sum of any surrounding pixels and itself.

Each kernel will have an associated activation map, which will be layered along the depth dimension to create the convolutional layer's whole output volume. each of the convolutional layers is responsible for activating specific areas of the input image.

By optimizing the model's output, convolutional layers can dramatically lower the model's complexity as compared to a conventional Artificial Neural Network (ANN). Three hyperparameters: depth, stride, and zero-padding, are used to optimize the performance of convolutional neural networks.

The depth of the output volume produced by the convolutional layers may be manually modified by altering the number of neurons in each layer concerning the same region of the input. Numerous ANN models show this to be the case, with every neuron in the hidden layer being directly connected to every other neuron in the preceding layer. While reducing this hyperparameter can greatly reduce the network's total number of neurons, it can also significantly reduce the model's ability to recognize patterns.

To position the receptive field, the stride may be specified to establish the depth around the spatial dimensionality of the input. For instance, if the stride is set to 1, the receptive field would be greatly overlapped and produce very big activations. Alternately, increasing the stride will lessen the amount of overlapping and result in a result with smaller spatial dimensions.

To further manage the dimensions of the output volumes, zero-padding is a straightforward approach that involves padding the input's boundary. These methods allow for changing the spatial dimensions of the convolutional layer output. formula (1) is used to compute this:

$$\frac{(V-R)+2Z}{S+1} \quad (1)$$

Where V is the size of the input volume (height, breadth, and depth), R is the size of the receptive field, Z is the amount of zero padding set, and S stands for the stride. Since the neurons won't fit neatly across the input when the estimated answer from this equation is not a whole integer, the stride has been adjusted improperly.

4.5.2. Pooling Layer:

Pooling layers seek to gradually reduce the dimensionality of the representation, hence lowering the model's computational complexity and parameter count. The "MAX" function is used by the pooling layer to scale the dimensionality of each activation map in the input. These typically take the shape of max-pooling layers with 2 x 2-dimensional kernels applied with a 2 stride along the input's spatial dimensions. This keeps the depth volume at its regular size while scaling the activation map down to 25% of its original size.

There are only two commonly noted max-pooling strategies. The pooling layers' stride and filters are often both set to 2 x 2, which enables the layer to stretch over the input's full range of spatial dimensions. Additionally, overlapping pooling may be used with a stride of 2 and a kernel size of 3. A kernel size greater than 3 will typically cause the model's performance to suffer significantly because of the destructive nature of pooling since the pooling layer is destructive.

It's also crucial to realize that CNN designs may also include general pooling in addition to max-pooling. L1/L2-normalization, average pooling, and other common operations may all be carried out by the pooling neurons that make up general pooling layers.

4.5.3. Fully Connected Layer:

Neurons in the completely connected layer have direct connections to the neurons in the two adjacent layers; they are not linked to any neurons in those

layers. This is comparable to how neurons are placed in conventional ANN models.

Two convolutional layers are frequently stacked before each pooling layer in a CNN design, as seen in Figure 8. This is highly advised since selecting more intricate aspects of the input vector is made possible by stacking numerous convolutional layers.

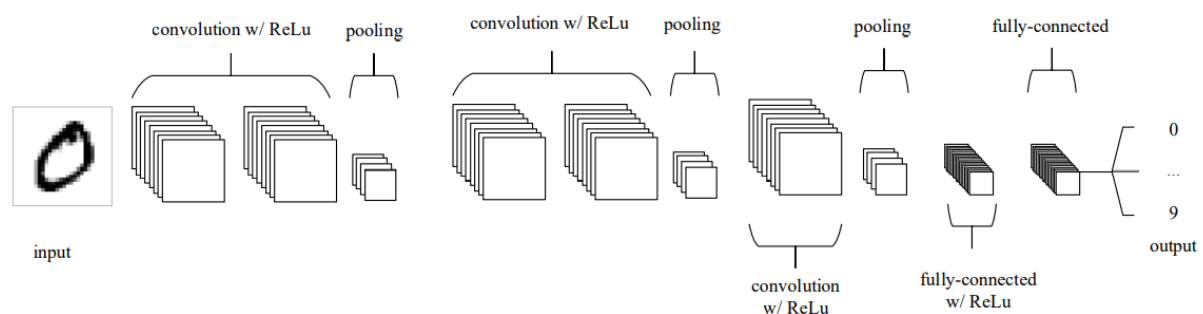


Figure 8 : A typical CNN design

A typical CNN design in which convolutional layers are continually layered between fully connected ReLUs before passing through the pooling layer and one or more fully connected ReLUs.

Although CNNs are very effective machine learning algorithms, they may be quite resource-intensive. For instance, If the input is 227×227 (as seen with ImageNet) and we're filtering with 64 kernels each with zero padding, then the result will be three activation vectors of size $227 \times 227 \times 64$ (which calculates to approximately 10 million activations) or an enormous 70 megabytes of memory per image. This problem can be illustrated by filtering a large image (anything over 128×128 could be considered large). You have two choices in this situation. First, by downsizing the raw photos to something a bit lighter, you can lower the spatial dimensionality of the input images. Alternatively, you can use larger filter sizes with a longer stride in defiance of what has been said before in this text.

4.5.4. Transfer Learning:

Transfer learning aims to raise the performance of target learners on target domains by transferring the knowledge obtained in multiple but related source

domains. By doing this, the dependency on a large quantity of target domain data for developing target learners may be reduced. Because there are so many potential applications, transfer learning has become a popular and promising area of machine learning.

Transfer learning **Error! Reference source not found.** is the process of learning a new activity more effectively by applying what has already been learned about a related one. The creation of algorithms that assist transfer learning is a subject of constant attention in the machine learning field, even though the majority of machine learning algorithms are made to handle single tasks. When modeling the second task, transfer learning is an optimization that enables quick advancement or increased performance.

Transfer learning is not just a topic for deep learning research; it also deals with issues like idea drift and multi-task learning. Transfer learning, however, is often used in deep learning due to the substantial resources needed to train deep learning models or the big and difficult datasets that deep learning models are trained on. Only general model characteristics that were learned from the initial task can be used for transfer learning in deep learning.

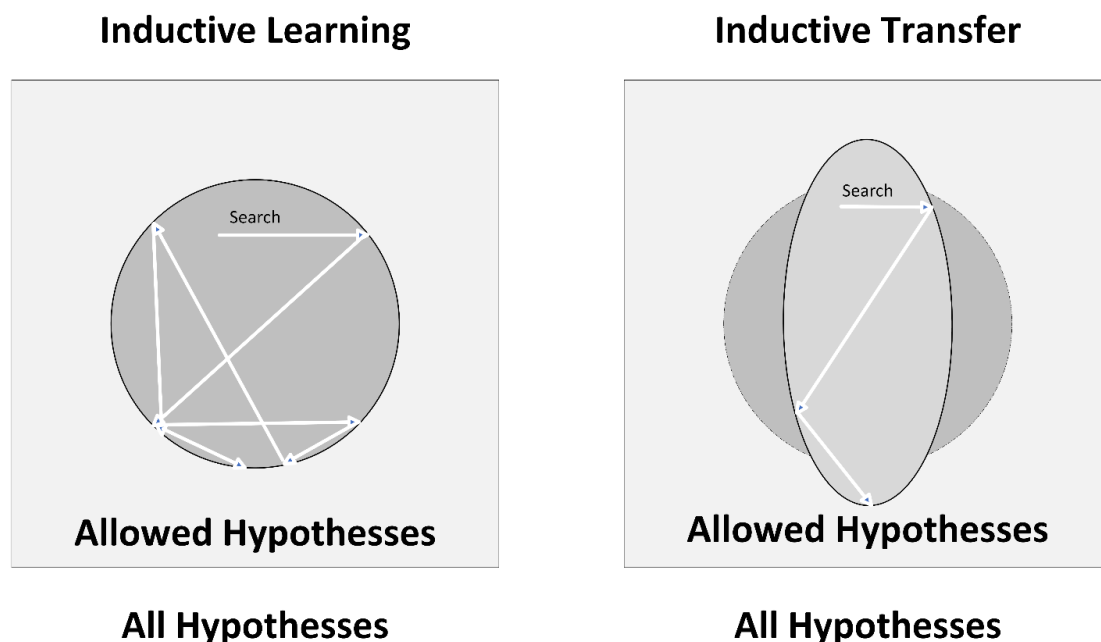


Figure 9 : Inductive transfer visualization.

In transfer learning, the learned features are first applied to a base network that is trained on a base dataset and task, and then the learned features are transferred to a second target network that is trained on a target dataset and task.

If the traits are general that is, applicable to both the base task and the target task rather than task-specific, this procedure is more likely to succeed. The inductive transfer is the type of transfer learning employed in deep learning. By utilizing a model fit on a distinct but related job, the range of potential models (model bias) is advantageously reduced in this situation. Figure 3 illustrates one way to think of inductive learning, that is as a guided search across a predetermined hypothesis space. Inductive transfer employs source-task information to modify the inductive bias, which may entail altering the search parameters or the hypothesis space.

Transfer learning is frequently used to solve challenges in predictive modeling that take input from images. This might be a prediction challenge that requires the input of images or video. A deep learning model that has been pre-trained for a difficult picture classification job like the ImageNet 1000-class photograph classification competition is frequently used for these kinds of tasks. The research organizations who create models for this competition and place well frequently make their final models available for reuse under a liberal license. In contemporary technology, these models may need days or weeks to train. These models are available for download and may be immediately integrated into new models that require picture data as input. The reason this strategy works so well is that the pictures were trained on a vast corpus of photographs and demanded the model to make predictions on a variety of classes, necessitating the model to effectively learn to extract features from photographs to succeed in the challenge.

4.5.5. EfficientNet:

Tan et al. created a new baseline network using neural architecture search and scaled it up to create the EfficientNets family of models, which outperformed earlier convolutional neural networks in terms of accuracy and efficiency. Their EfficientNet-B7 outperforms the best convolutional neural networks currently in use by being 8.4 times smaller and 6.1 times quicker at inference while achieving state-of-the-art 84.4% top-1 / 97.1% top-5 accuracy on ImageNet. With state-of-the-art accuracy on the CIFAR-100 (91.7%), Flowers (98.8%), and 3 additional transfer learning datasets, they proved that their EfficientNets also transfer well. they showed that a mobile-size EfficientNet model can be scaled up extremely well, reaching state-of-the-art accuracy with an order of magnitude fewer parameters and FLOPS. Figure 2.4 compares the performance of the EfficientNetB7 to other common convolutional neural networks. Model

size against ImageNet Accuracy. All figures are for a single crop and one model. Other convolutional neural networks are greatly outperformed by their EfficientNets. For example, EfficientNet-B7 achieves 84.4% top-1 accuracy at a new state-of-the-art level while being 8.4 times smaller and 6.1 times quicker than GPipe. ResNet-152 is 7.6 times smaller and 5.7 times slower than EfficientNet-B1.

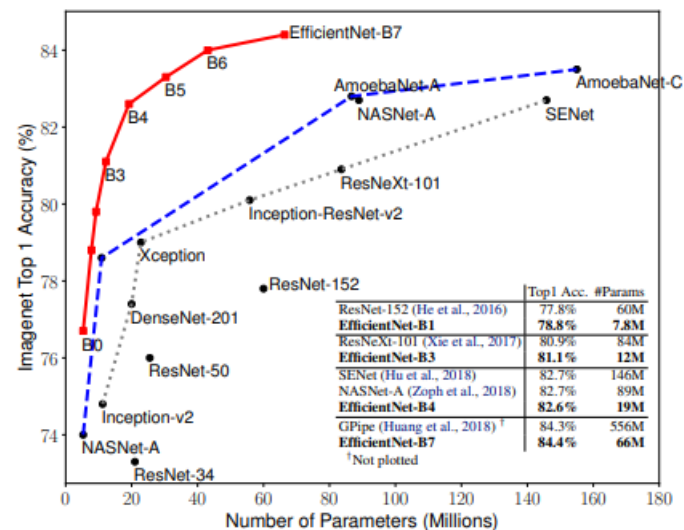


Figure 10 : EfficientNet-B7 performance

EfficientNet-B7 performance against other well-known convolutional neural networks.

4.6. Malware Detection Techniques:

Malware, or malicious software, represents a significant threat to the digital ecosystem, encompassing a broad range of software designed to harm or exploit any programmable device, service, or network. Cybersecurity professionals face the constant challenge of detecting and mitigating malware due to its evolving nature and increasing complexity. This chapter explores various malware detection techniques that have been developed over the years, detailing their methodologies, applications, strengths, and limitations. The discussion extends into modern approaches, particularly focusing on the use of machine learning and deep learning techniques, and delves into the specialized area of evasive malware detection, especially within PDF files.

4.6.1.Traditional Malware Detection Techniques

Signature-Based Detection:

The most traditional form of malware detection is signature-based detection, which relies on a database of known malware signatures which is a sort of digital fingerprint of malicious code. This method compares the signatures in the database with the files or code executing on a system to identify potential threats. While effective against known malware, this technique struggles with zero-day attacks (newly discovered malware exploits) and polymorphic or metamorphic malware, which can alter their code signatures to evade detection.

Heuristic-Based Detection:

To address the limitations of signature-based detection, heuristic techniques were developed. Heuristic-based detection uses algorithms to analyze the behavior of a program to determine if it behaves like malware. This approach involves more abstract rules and characteristics derived from known malware samples. For instance, a heuristic detector might flag any program that tries to write data to an executable file, a common behavior in viruses. The flexibility of heuristic detection makes it more effective against previously unknown viruses compared to signature-based methods, but it also leads to higher false positives.

Behavioral-Based Detection:

Expanding on heuristic techniques, behavioral-based detection monitors the behavior of programs during execution. By constructing a baseline of normal activity for the system, this method can alert on deviations that suggest malicious activity. Behavioral detection is potent against malware that can evade static analysis, including zero-day threats. However, it requires considerable system resources and can potentially impact system performance.

Sandboxing:

Sandboxing involves running programs in a virtual environment to observe their behavior without risking the main operating system. This technique allows untrusted programs or code to execute in isolation, wherein their behavior is analyzed for potential malicious actions. Sandboxing is particularly useful for detecting exploits that use legitimate programs for malicious purposes.

4.6.1.Advanced Techniques Using Machine Learning:

With the advent of more sophisticated cyber threats, traditional methods alone have proved insufficient to secure digital assets. Machine learning (ML) offers powerful tools for enhancing malware detection systems due to its ability to learn from data, identify patterns, and make decisions with minimal human intervention.

Static Analysis Using Machine Learning:

Machine learning models can be trained to analyze the static properties of executable files without running them. Static analysis involves examining the binary structure, dependencies, and embedded resources to detect malicious patterns. Features from the executable such as byte sequences, binary headers, and metadata are fed into ML models that classify the files as benign or malicious. The advantage of static analysis is its speed and the fact that it does not require execution of the malware, thereby avoiding potential harm.

Dynamic Analysis Using Machine Learning:

Dynamic analysis, on the other hand, involves observing the behavior of a program while it is running. Machine learning models are trained on runtime behaviors such as system calls, network traffic, and changes to file systems. Dynamic analysis is more effective at detecting sophisticated malware that only reveals its malicious intent when executed under specific conditions. Machine learning enhances dynamic analysis by quickly correlating complex behaviors across many dimensions, outperforming traditional behavioral detection techniques.

Deep Learning in Malware Detection:

Deep learning, a subset of machine learning with multiple layers of processing, is particularly well-suited for detecting patterns in large datasets. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Autoencoders are some of the deep learning architectures used in malware detection. These models can handle raw binary data, extracting intricate features that are not easily recognizable by humans or traditional ML techniques.

Transfer Learning in Malware Detection:

Transfer learning, where models pre-trained on large datasets are fine-tuned for specific tasks, has proven to be extremely effective in malware detection. By leveraging pre-trained models on general tasks, transfer learning reduces the need for extensive labeled datasets specific to malware, allowing for quicker and more efficient training.

Evasive Malware Detection:

Evasive malware employs sophisticated techniques to avoid detection by traditional security mechanisms. Among these, evasive malware in PDF files is particularly challenging due to the complexity and ubiquity of PDFs. PDF files are a common medium for sharing documents, making them a prime target for attackers.

Techniques for Evasive Malware Detection:

Obfuscation Techniques:

Evasive PDF malware often uses code obfuscation to disguise its true nature. This can involve complex encoding schemes, script embedding, and the use of multiple layers of embedded objects. Detecting such obfuscation requires advanced analysis techniques that can decode and deconstruct the layered structures within the PDF.

Anti-Analysis Techniques:

To evade detection, malware can incorporate anti-analysis features such as environmental checks to determine if it is being run in a virtualized or sandboxed environment. If such an environment is detected, the malware can alter its behavior or remain dormant to avoid detection. Detecting these anti-analysis techniques involves deep behavioral analysis and the use of sophisticated monitoring tools.

Machine Learning and Deep Learning for Evasive PDF Malware:

Machine learning and deep learning offer advanced methods for detecting evasive PDF malware. These techniques can be applied to both static and dynamic analysis of PDF files.

Static Analysis Using Machine Learning:

Static analysis of PDFs involves examining the file's structure, metadata, and embedded objects without executing the code. Features such as the presence of JavaScript, the number of embedded objects, and the use of suspicious fonts can be extracted and analyzed using machine learning models. By training models on known benign and malicious PDF samples, machine learning can identify patterns indicative of malicious behavior.

Dynamic Analysis Using Machine Learning:

Dynamic analysis entails running the PDF in a controlled environment to observe its behavior. Machine learning models can be trained to detect unusual activities such as unexpected network connections, file modifications, or system changes. Dynamic analysis is particularly effective against malware that only exhibits malicious behavior when certain conditions are met.

Deep Learning Approaches:

Deep learning, particularly CNNs, has shown promise in detecting evasive PDF malware. By converting PDF files into images, CNNs can be trained to recognize visual patterns that correlate with malicious behavior. This approach leverages the power of image recognition in deep learning, providing a novel method for malware detection.

Case Study: EfficientNet for Evasive PDF Malware Detection

EfficientNet, a state-of-the-art CNN architecture, has been applied to the problem of evasive PDF malware detection. By leveraging transfer learning and training EfficientNet on a large dataset of PDF files, significant improvements in detection accuracy have been achieved. The model can identify subtle patterns and anomalies in the PDF files that are indicative of malware, achieving high precision and recall.

Advantages of Deep Learning for Evasive Malware Detection:

1. **Automation:** Deep learning models can automatically extract relevant features from raw data, reducing the need for manual feature engineering.
2. **Accuracy:** Deep learning models can achieve higher accuracy compared to traditional methods, especially in detecting previously unknown threats.
3. **Scalability:** Deep learning models can handle large volumes of data, making them suitable for enterprise-level cybersecurity solutions.

Challenges and Future Directions:

While deep learning and machine learning have enhanced the capabilities of malware detection systems, they also face several challenges:

- **Adversarial Attacks:** Cybercriminals can employ adversarial machine learning techniques to craft malware that fools learning models into classifying malicious files as benign.
- **Data Quality and Availability:** High-quality, labeled datasets are crucial for training accurate models. The lack of such datasets, especially for new types of malware, can hinder the effectiveness of ML-based detection systems.
- **Interpretability:** Deep learning models, often considered as black boxes, lack interpretability, which is crucial for cybersecurity professionals to trust and understand the decision-making process of these models.

The future of malware detection will likely focus on developing more robust adversarial machine learning techniques, improving model interpretability, and integrating various data sources to enrich the training datasets. Hybrid models combining the strengths of traditional methods with advanced ML techniques may offer the best of both worlds enhanced detection capabilities with lower false positives.

4.7.Literature Review:

Vasan et al. (2020) made the supposition that multiple convolutional neural networks provide varied semantic representations of the picture following their deeper architectures. So, they trained a set of CNNs using the ImageNet dataset to extract higher-quality features (VGG16 and ResNet50). Several multiclass classifiers were trained utilizing transferred features and fused posterior probabilities using the collected features to increase the classification accuracy of families of unknown malware samples. They named their approach Image-based Malware Classification using Ensembles of CNNs (IMCEC). As they mentioned sample categorization only re-quires an average of 1.18 seconds, efficiency is the key benefit of this method. as it produced an overall exceptional performance on the MalImg dataset.

Moreover, Kumar et al. (2021) introduced a brand-new MCFT-CNN (Malware classification with fine-tuned convolution neural networks) methodology. This method replaced the ResNet50 model's initial output layer with a fully connected dense layer, followed by a SoftMax layer. It was trained on the

MallImg dataset and employed deep transfer learning on ResNet50. As a result, the network's performance greatly increased. On the MallImg dataset, they attained a high accuracy of 99.18%. When evaluated on the Microsoft malware classification dataset (BIG 2015), it similarly showed consistency, with an accuracy of 98.63%.

El-Shafai et al. (2021) further attempt to solve the issue of imbalanced datasets and reduce the false positive rate using eight precisely tuned convolutional neural networks: ResNet50, InceptionV3, DenseNet201, Places365-GoogleNet, MobileNetV2, DarkNet53, VGG16, and AlexNet. They concluded that the greatest performance could be reached by utilizing a fine-tuned VGG16 that attained an accuracy of 99.97% on the MallImg dataset by performing transfer learning with fine-tuning and comparing the performance on the previously stated models.

Also, Oloroso et al. (2020) used transfer learning on the VGG16 network to categorise the nine distinct malware families in the dataset for the Microsoft Malware Challenge (BIG 2015). Five series of tests employing various CNN architectures were evaluated using a set of performance parameters. According to the results of these tests, VGG16 fared better than all other methods, with an average classification accuracy of 98.8%.

Moreover, employing convolutional neural network-based algorithms like VGG16, ResNet, and AlexNet, Davuluru, et al. (2019) investigated the effectiveness of various transfer learning strategies. They bring forth a computationally effective classification architecture that combines KNN (K-nearest neighbours) and SVM for classification and CNNs for feature extraction. They concluded that adopting fusion-based techniques, which combine all the probabilities produced by the CNNs, significantly enhanced performance. To reach the greatest overall accuracy of 99.4%, they were able to employ a CNN for feature extraction and an SVM with a linear kernel for classification.

Additionally, Lo et al. (2019) suggested combining transfer learning with a convolutional neural network built on the Xception framework. The Microsoft malware classification challenge malware dataset (BIG 2015) and the MallImg dataset were used to train and test the Xception-based model. In this method, the predictions from the two separate file types are combined using an ensemble (.asm and .byte). The MallImg dataset and the Microsoft malware classification challenge dataset respectively obtained validation accuracy of 99.03% and 99.17%. In the original paper, other crucial measures like the f1-score were not mentioned.

A layered ensemble approach was presented by S. Abijah et al. (2020) that mimics deep learning approaches but has the benefit of being significantly less sophisticated because it does not involve backpropagation or hyperparameter adjustment. However, this method employs a 2D visual representation of the infection, unlike deep learning solutions. On the MaleVis dataset, this deep random forest method has an accuracy of 97.43%. By utilizing several sets of hyperparameters to train three models on the MaleVis, BIG 2015, and MalImg, the authors were able to achieve much better performance than other methods included in the literature review they addressed.

Transfer learning has shown promise in improving performance in various tasks. To identify malware on the MaleVis dataset, Ahmet et al. (2019) utilized transfer learning utilizing a range of contemporary CNNs, including DenseNet, AlexNet, VGG, Inception, and Resnet, which have shown efficacy in picture classification challenges. This method makes use of 3-channel colors (RGB). They compared their anticipated performance and concluded that DenseNet201, with an accuracy of 97.48%, was the model that performed the best.

Moreover, Kumar et al. (2022) introduced DTMIC (Deep Transfer Learning for Malware Image Classification), which utilized the CNN architectures that had previously undergone ImageNet training. Grayscale pictures are created from the PEs (Portable Executable Files), which are then flattened and forwarded via the network. The early halting was applied as a regularization method to deal with the overfitting issue. For testing, they employed the Microsoft malware classification challenge dataset (BIG2015) and the MalImg dataset. On the BIG 2015 dataset, DTMIC achieved a test accuracy of 93.19%. On the MalImg dataset, it was able to achieve a substantially better test accuracy of 98.92%. The authors concluded that the suggested model outperformed ResNet50, inceptionV3, VGG16, and VGG19-based networks after testing and evaluating various transfer learning networks.

In their notable 2022 paper, Yadav, Menon, Ravi, Vishvanathan, and Pham probed the escalating complexity of malware threats and presented an EfficientNet-B4 convolutional neural network (CNN)-based model for Android malware detection. Addressing the inadequacies of traditional malware detection techniques, such as heavy reliance on human intervention, extensive resource requirements, and vulnerability to polymorphic and obfuscation tactics by malware authors, they explored the potential of deep learning (DL) in this realm. This work scrutinized the performance of 26 advanced, pre-trained CNN models alongside large-scale learning with Support Vector Machine (SVM) and Random Forest (RF) classifiers, coupled with stacking with CNN models, for

Android malware detection. Their innovative approach involves the use of image-based malware representations of Android DEX files, from which the EfficientNet-B4 model extracts salient features. After going through a global average pooling layer, these features are input to a softmax classifier. The robustness of this proposed model is underscored by an impressive 95.7% accuracy rate in the binary classification of Android malware images, which eclipsed all comparative models across performance metrics. The authors also provided insightful commentary on potential directions for future research in evasive malware detection, a realm that continues to present formidable challenges, including the distinction between legitimate and malicious evasion techniques, tackling unknown evasion methods, and managing complexity in terms of time and resources.

Yadav et al. (2022) also highlighted the pressing need for an evasive behavior dataset and efficient feature extraction and representation techniques that hone in on evasion techniques tied only to malicious behavior. They pointed to the increasing production of zero-day and unknown malware variants, spurred by the ready availability of online tools that facilitate the creation of new malware or reformat existing ones using obfuscation techniques. The researchers concluded by calling for advanced machine learning techniques, both supervised and unsupervised, to be implemented for updating learning and developing models that can adaptively learn new malicious behaviors.

In the realm of PDF malware detection, Abu Al-Haija, Odeh, and Qattous (2022) have put forth a pioneering system that leverages the AdaBoost decision tree with optimal hyperparameters to distinguish benign from malicious PDF files. With the widespread adoption of PDF files, they have become attractive targets for hackers to embed malicious codes and generate security threats. This necessitates sophisticated detection methods to safeguard PDF files, which are increasingly provided by machine learning techniques.

Abu Al-Haija et al. designed a detection system trained and evaluated on the modern inclusive Evasive-PDFMal2022 dataset achieving an impressive prediction accuracy of 98.84% and a short prediction interval of 2.174 μ Sec. Thus, their system not only shows superior performance in comparison to other contemporary models in the same domain but also has a low detection overhead.

The authors demonstrated the superiority of their system across multiple performance metrics such as detection accuracy, precision, sensitivity, and F-Score. Their work has made a significant contribution to devising a comprehensive machine-learning-based model for analyzing PDF documents to

identify malicious files. The use of optimizable decision trees with the AdaBoost algorithm and optimal hyperparameters makes their model particularly potent.

The system uses 37 significant static features extracted from each PDF file in the Evasive-PDFMal2022 dataset, which consists of 10,025 records. The authors identified gaps in current state-of-the-art methods, providing valuable insights for future research directions. For instance, they suggested the exploration of other commonly used document formats and the possible employment of more complex deep learning algorithms such as CNN and LSTM for better detection of subtle changes in malware.

As for vision-based evasive PDF malware detection, Fettaya et al. (2020) presented a custom CNN to identify fraudulent PDF files based solely on their byte level. The authors draw attention to the growing use of PDF files in industry, government, and academia as a preferred means of virus delivery. They contend that it is imperative to investigate more sophisticated strategies because the conventional signature-based methods for identifying malware are proving to be less and less successful. Fettaya and Mansour's proposed CNN-based approach is shown through their trials to produce excellent accuracy in detecting fraudulent PDFs, making it a promising option for enhancing cybersecurity measures. This work shows the potential of CNN-based techniques in this area and offers useful insights into the creation of efficient malware detection systems. They have demonstrated that their network provides performance on par with some of the finest antivirus available today without the need for feature extraction or preprocessing. Their proposed approach accomplished an accuracy of 99.83% on the Contagio dataset.

In their comprehensive survey, Aboaoja et al. (2022) have emphasized the escalating intricacies of malicious software threats, particularly underlining the surge in obfuscation and evasion techniques utilized by malware creators to bypass detection systems. They highlighted the steady rise in the production of malware and malware variants every day, which has compounded the challenge of detecting and mitigating these cyber threats. The researchers pointed out that conventional malware detection solutions often struggle to identify and tackle a wide array of malware types, particularly zero-day attacks that exploit vulnerabilities before a patch is available.

The researchers stressed the lack of extensive review articles in the domain that effectively connect each analysis and detection approach with the corresponding data type. As per their assertion, such an association is of utmost importance to the re-search community as it aids in determining suitable mitigation strategies

for various kinds of malware. In a bid to fill this gap, they have meticulously presented a detailed taxonomy of malware analysis and detection methodologies, feature extraction techniques, and the most associated data types for each approach.

Moreover, they drew attention to the often-neglected aspect of feature representation methods taxonomy, arguing that it is of paramount significance for developing efficient and robust malware detection models. They proposed a novel taxonomy that classifies methods of feature representation, presenting a clear and comprehensive structure for understanding this critical aspect of malware detection research.

The survey did not merely end with proposing novel taxonomies and pointing out existing gaps in the research; it delved deeper into the root cause problems that each approach or method for analysis, detection, extraction, and representation faced, leading to their respective drawbacks. By doing so, Aboaoja et al. (2022) have not only broadened the understanding of the current challenges and issues in the field of malware detection but also presented a detailed roadmap that could guide future research endeavors in this crucial area of cybersecurity.

While a plethora of research exists on image-based malware detection using deep learning and transfer learning, the domain of evasive PDF malware detection is relatively unexplored, especially with the application of EfficientNet architectures through transfer learning. Existing research has demonstrated the efficacy of deep learning models in malware detection, but primarily focusing on image datasets or broad malware classification. Although transfer learning has shown promise, its utilization specifically for evasive PDF malware detection with EfficientNet remains a gap.

To the best of our knowledge, there is limited research on evasive PDF malware detection using transfer learning, particularly using the EfficientNet architecture. Studies suggest that EfficientNet architectures can provide superior performance compared to other architectures like ResNet and VGG (Tan et al., 2020), yet their application in evasive PDF malware detection is yet unexplored.

Next, the performance of the previously stated methods on the three primary malware classification datasets, MalImg, MaleVis, and Microsoft malware classification challenge dataset, is summarized in Table 1 from the literature review. The table lists all metrics that are accessible. Other than test accuracy ratings, some articles don't include any metrics. This may be false for datasets that are unbalanced. Additionally, it is evident that, in comparison to other

methods, the use of transfer learning in the malware detection and classification job obtained very good performance.

Table 1. Summary of related models' performance.

Reference	Dataset	Method	Accuracy	F1 score	Precision	Recall
(Vasan et al., 2020)	MalImg	IMCEC, TL(VGG16)	99.50	N/A	99.50	99.46
(Awan et al., 2021)	MalImg	SACNN, TL(VGG19)	97.38	97.21	97.28	97.56
(Kumar, 2021)	MalImg	MCFT-CNN, TL(ResNet50)	99.74	99.65	99.56	99.7
(El-Shafai et al., 2021)	MalImg	TL(VGG16)	99.97	99.02	99.04	99.01
(Olowoyo, Owolawi, 2020)	BIG 2015	TL(VGG16)	98.8	98.3	98.2	98.5
(Narayanan et al., 2020)	BIG 2015	CNN, LSTM, SVM	99.8	N/A	N/A	N/A
(Davuluru et al., 2019)	BIG 2015	CNN, SVM	99.4	N/A	N/A	N/A
(Lo et al., 2019)	MalImg, BIG 2015	TL(Xception)	99.03, 99.17	N/A	N/A	N/A
(Roseline et al., 2020)	MaleVis, MalImg, BIG 2015, Malicia	Deep RF	97.43, 98.65, 97.2, 87.68	97.42, 98.74, 97.20, 87.68	97.43, 98.86, 97.61, 87.78	97.32, 98.63, 96.79, 87.59
(Bozkir et al., 2019)	MaleVis	TL(DenseNet201)	97.48	N/A	N/A	N/A
(Barros et al., 2022)	MaleVis, MalImg	Malware-SMELL,	93.70, 97.76	93.42, 97.69	93.16, 97.84	93.06, 97.76

		TL(VGG19), Contrastive loss				
(Kumar, Janet, 2022)	MalImg, BIG 2015	DTMIC	98.92, 93.19	N/A	N/A	N/A
(Yadav et al., 2022)	Private Dataset	EfficientnetB4	95.7	96.00	96.00	96.00
(Abu Al-Haija et al., 2022)	Evasive-PDFMal2022	Optimizable Decision Trees	98.84	98.80	98.90	98.80
(Fettaya et al., 2020)	Contagio	CNN	99.83	N/A	N/A	N/A

In conclusion, there is a research gap in the application of transfer learning for evasive PDF malware detection, particularly when employing EfficientNet. This gap highlights the need for more reliable techniques that can precisely categorize different types of evasive PDF malware. Therefore, the goal of this research is to use transfer learning on the EfficientNet pre trained architecture to create a model that can accurately identify malicious PDF files using the CIC-Evasive-PDFMal2022.

4.8.Methods:

This chapter presents the methodology utilized for the detection and classification of evasive malware, with a particular focus on those distributed through PDF files. The methods applied in this study encompass the stages of data collection, data preprocessing, model design and configuration, model training and validation, and results analysis. The central goal of these methods is to build a deep learning model using the EfficientNet architecture to effectively identify and categorize evasive PDF malware.

4.8.1. Data Collection:

In this study, data was collected from the CIC-Evasive-PDFMal2022 dataset. The dataset focuses on the Portable Document Format (PDF) which, due to its widespread use, has become a target for attackers exploiting its features to

deliver malicious payloads. Specifically, this dataset includes PDF files that are evasive, making them difficult to detect through conventional learning algorithms.

According to the authors of the dataset, initially, 11,173 malicious files were collected from Contagio, 20,000 malicious files from VirusTotal, and 9,109 benign files from Contagio. The dataset underwent processing where 32 features were extracted from each file, and after de-duplication and applying clustering techniques, a more representative dataset was constructed which consists of 10,025 records with 5,557 malicious and 4,468 benign records.

In our study, we focus on the PDF files themselves rather than the CSV file. The actual dataset when downloaded and processed contains 9,107 PDF files classified as benign and 21,899 PDF files classified as malicious.

The data was programmatically retrieved using a Python script. The script downloads two compressed files (in tar.gz format) from the specified URLs. One file contains benign PDF files, while the other contains malicious PDF files.

4.8.1.1. Below is an overview of the steps performed by the script:

1. For each file URL (benign and malicious), the script sends an HTTP GET request to download the tar.gz file.
2. The script saves the tar.gz file to the local machine.
3. The tar.gz file is extracted, revealing multiple zip files.
4. The script then extracts the zip files into specified folders.

All files from subdirectories within the malicious PDF files' directory are moved into a single directory for easier access.

After extraction, the dataset consists of two main directories: Benign_PDF and Mal_PDF. The Benign_PDF directory contains benign PDF files, while the Mal_PDF directory contains malicious PDF files. The study focused on the raw PDF files themselves and did not make use of any preprocessed data in CSV format. As such, subsequent analyses and experiments were performed directly on these PDF files.

Though the dataset provider extracted 37 static features from each PDF file, including general and structural features, this study used the raw PDF files for

analysis after converting them to images. This provided the opportunity to perform independent feature extraction and analysis using various techniques.

4.8.2. Data Pre-processing:

Data pre-processing is a crucial step in this study, involving the transformation of the PDF files into a format that can be effectively utilized by the model. The malware PDF files were converted into image files which allow us to exploit deep learning techniques developed for computer vision tasks. Figure 11 summarizes the transformation process.

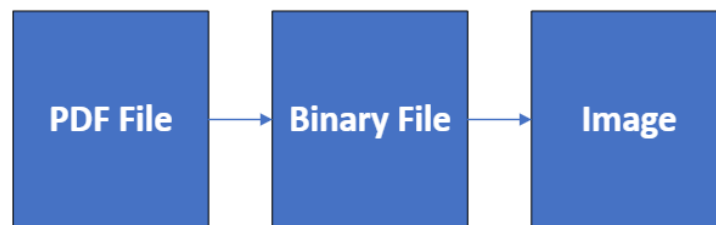


Figure 11 : The conversion process of evasive PDF malware files into images

The methodology involved two major transformations. Firstly, the PDF files, which were categorized into benign and malicious, were converted into binary format. This conversion was accomplished through a Python script that read the content of each PDF file in its raw byte form and subsequently wrote this content into a new file with a '.bin' extension. This was essential to decouple the data from the PDF file format and prepare it for further processing.

The second transformation involved converting the binary files into PNG image files. This transformation was achieved using a Python function, which created blank images with predefined dimensions of 128x128 pixels. Each byte in the binary data was interpreted as a colour channel intensity, with groups of three consecutive bytes being considered as the RGB (Red, Green, Blue) channels of a single pixel. This process filled in the blank images with pixel data derived from the binary files.

In more detail, the RGB allocation process worked by reading the binary file byte by byte. For every pixel in the image, three bytes were read from the binary data. The first byte read was assigned to the red (R) channel, the second byte was assigned to the green (G) channel, and the third byte was assigned to the blue (B) channel of the pixel. If the binary data was exhausted before all

pixels were filled (i.e., before reaching the end of the file), the remaining pixels were filled with black (RGB values of 0,0,0). The script then moved on to the next pixel, repeating the process until all pixels in the image were filled. If a file's binary data was not sufficient to fill all pixels of an image, the remaining pixels were set to black (RGB values of 0,0,0).

These PNG images were then stored and used as the input for the EfficientNet. This two-step transformation process was critical to converting the raw data in PDF files into a suitable format for image-based machine learning algorithms. An example of the transformed benign and malware images is shown in Figure 12.

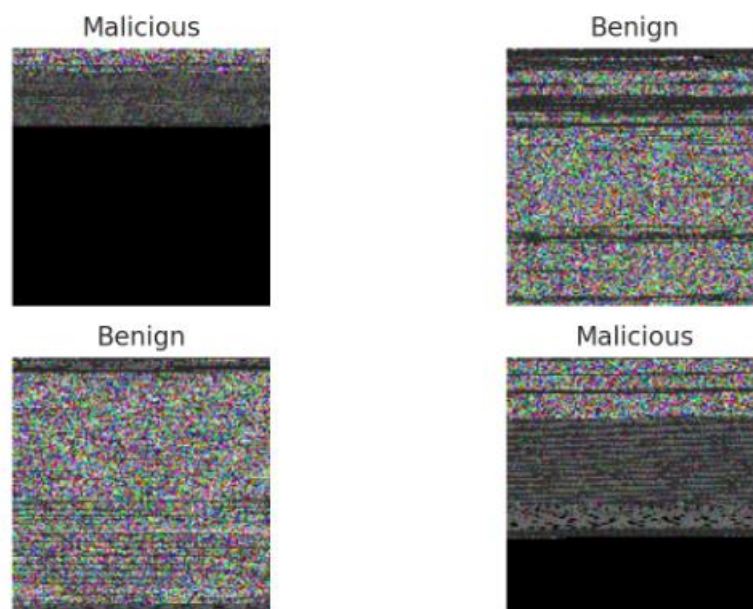


Figure 12 : Examples of benign and malicious images

Following the two-step transformation process, the next stage of data preprocessing involved organizing the image data into a structured format, and partitioning it into training, validation, and testing sets. The dataset consisted of a total of 31,006 images, of which 21,899 were malicious and 9,107 were benign. Since the dataset was imbalanced with a higher proportion of malicious images, it was imperative to ensure that each subset (training, validation, and testing) contained a representative sample of the class distribution in the overall dataset. This is essential for evaluating the model's performance accurately.

To achieve this, the dataset was first converted into a Pandas DataFrame with two columns 'file_path' and 'label'. The 'file_path' column contained the file paths of the images, and the 'label' column contained the associated labels, either 'Benign' or 'Malicious'.

Subsequently, the dataset was split into training, validation, and testing sets using stratified sampling. Stratified sampling ensures that each subset contains approximately the same percentage of samples of each target class as the complete set. In this study, the dataset was divided into 80% training data, 10% validation data, and 10% testing data. The 'stratify' parameter was set to the label column, ensuring that the class distribution was preserved across the splits. Specifically, the training set contained 7,285 benign images and 17,519 malicious images; the validation set contained 911 benign and 2,190 malicious images, and the testing set mirrored the validation set in terms of distribution.

The use of stratified sampling was crucial in addressing the class imbalance problem. By ensuring that the subsets reflected the original class distribution, it was possible to prevent the model from being biased towards the majority class, which could have led to poor generalization when classifying new data.

The ImageDataGenerator from Keras was employed to create generators for the training, validation, and testing sets. This allowed for efficient memory management by loading the images in batches, instead of loading all the images into memory at once. A batch size of 256 was chosen. The generators were also configured to resize the images to the required dimensions of 128x128 pixels and to interpret the labels in binary format (0 for benign and 1 for malicious).

In summary, the data preprocessing phase included transforming PDF files to image format, structuring the data, and partitioning it into representative training, validation, and testing sets. These steps were critical in preparing the data for training and evaluation using deep learning models.

This data preparation process was instrumental in setting up an efficient and scalable pipeline for training the EfficientNet model using the transformed image data.

4.8.3. Model Design and Configuration:

For the task of classifying PDF files as either benign or malicious based on their image representations, an EfficientNet architecture was utilized.

The EfficientNet architecture is known for its efficiency and effectiveness, achieving high accuracy with fewer parameters compared to other models. This makes it particularly suitable for dealing with image data, which can be computationally intensive to process. Specifically, the EfficientNetB0 variant was selected as the base model. This variant is the smallest and most

lightweight version of the EfficientNet family, which makes it faster and more memory-efficient which is a critical consideration for large-scale data processing and analysis. Another reason for choosing the EfficientNetB0 is its pre-trained weights on the ImageNet dataset, which can act as a good initialization for the weights and facilitate faster convergence during training.

The base model (EfficientNetB0) was configured to not include the top (output) layer, since the classification task in this study is binary (benign or malicious), which is different from the original multi-class classification task in ImageNet. The input shape was set to 128x128x3, matching the dimensions of the pre-processed image data. The base model's layers were set to be trainable.

On top of the base model, additional layers were added to tailor the network for the binary classification task. A Global Average Pooling 2D (GAP2D) layer was added to reduce the spatial dimensions of the feature maps. This is followed by a Dense layer with a single neuron, using a sigmoid activation function, which outputs the probability of the input image being malicious.

The model was compiled using the Adam optimizer, with binary cross-entropy as the loss function which is suitable for binary classification problems. The metrics used to monitor the model's performance were accuracy, precision, and recall, which are essential to evaluate the model, especially because of the imbalanced dataset.

4.8.4. Model Training and Validation:

The model was trained using the previously defined training and validation data generators. Several callbacks were utilized to monitor and optimize the training process:

- Early Stopping: This callback monitored the validation loss and stopped the training if there was no improvement in validation loss for 25 consecutive epochs. It also restored the model weights to the best iteration.
- Model Checkpoint: This saved the model weights to a file whenever there was an improvement in the validation loss.

- Reduce Learning Rate on Plateau: This reduced the learning rate when the validation loss stopped improving, to fine-tune the model parameters.

The initial learning rate was set to $1e-5$ and the model was trained for up to 300 epochs, though early stopping could terminate the process sooner if there was no improvement in validation loss. The training process involved backpropagation and optimization of the model parameters to minimize the binary cross-entropy loss.

The model's performance was continually evaluated on the validation dataset throughout the training process, and the training was configured to shuffle the training data at each epoch to prevent the model from memorizing the order of the samples.

By training the model using this methodology, the aim was to build a robust and accurate classifier capable of distinguishing between benign and malicious PDF files based on their image representations.

4.8.5. Model Evaluation:

Utilizing mainly the F1-score, we will assess the proposed malware detection technique on the testing set. The weighting-harmonic mean of the precision and recall is known as the F1 score. Various concepts are taken into consideration when defining evaluation metrics which are:

- False Positive (*FP*) is where the negative sample is incorrectly classified as the positive sample.
- False Negative (*FN*) is where the positive sample is incorrectly classified as the negative sample
- True Positive (*TP*) is where the positive sample is correctly classified as the positive sample
- True Negative (*TN*) is where the negative sample is correctly classified as the negative sample

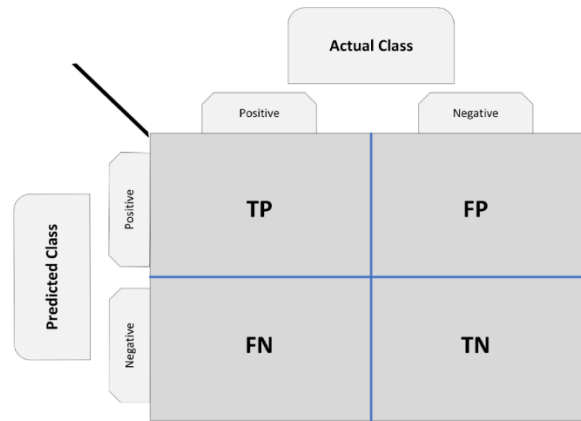


Figure 13 : Confusion Matrix for binary class classification

4.8.5.1. Accuracy:

One parameter for assessing classification models is accuracy. The percentage of predictions that our model correctly predicted is known as accuracy.

Equation (2) is the definition of accuracy:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

Accuracy is an unreliable metric itself for imbalanced datasets as in our case. Therefore, we need other metrics to measure the performance of the model.

4.8.5.2. Precision:

The percentage of labels that were correctly predicted positively is represented by the model precision score. Another name for precision is the positive predictive value. False positives and false negatives are traded off using precision together with recall. The class distribution has an impact on precision. Precision will be worse if there are more samples in the minority class. One way to think of precision is as an indicator of accuracy or caliber. A model with high precision is the one we would use if we wanted to reduce false negatives. Contrarily, we would pick a model with high recall if we wanted to reduce the number of false positives.

Precision is mostly employed when predicting the positive class is necessary since false positives are more expensive than false negatives, like in spam filtering or medical diagnosis. For instance, if a model predicts the spam status

of an email with 99% accuracy but only 50% precision, that email is not spam 50% of the time.

When the classes are severely unbalanced, the precision score is a helpful indicator of the accuracy of the forecast. It reflects the ratio of genuine positives to the total of true positives and false positives mathematically. Precision can be calculated as shown in equation (3):

$$P = \frac{TP}{TP + FP} \quad (3)$$

You can see from the calculation above that the value of false-positive will influence the precision score. Therefore, if a high precision score is crucial for the business objectives, you may choose to focus accordingly to construct models with fewer false positives.

4.8.5.3. Recall:

The model's ability to properly forecast positives out of real positives is measured by the model recall score. This differs from precision, which counts the proportion of accurate positive predictions among all positive predictions given by models. The recall score would be the percentage of positive reviews that your machine learning model properly identified as positive, for instance, if you were trying to identify positive reviews. In other words, it assesses how well our machine learning model can distinguish between all true positives and all false positives inside a dataset. The machine learning model is more adept at recognizing both positive and negative samples the higher the recall score.

Sensitivity or the true positive rate are other names for recall. A high recall score shows how well the model can locate examples of success. On the other hand, a low recall score shows that the model is poor in identifying success stories. To provide a comprehensive view of the model's performance, recall is sometimes combined with additional performance measures like precision and accuracy. It symbolizes the ratio of true positives to the total of true positives and false negatives in mathematics. Recall can be calculated as shown in equation (4):

$$R = \frac{TP}{TP + FN} \quad (4)$$

You can see from the calculation above that the recall score would depend on the false-negative value. Therefore, if a high recall score is crucial for the business objectives, you may opt to focus suitably on constructing models with smaller false negatives.

4.8.5.4. F-1 Score:

The model score as a function of recall and precision is represented by the model F1 score. A substitute for accuracy measures (it doesn't require us to know the entire number of observations), the F-score is a machine learning model performance statistic that equally weights precision and recall when assessing how accurate the model is. It is frequently used as a single value that offers summaries of the model's output quality. This is a helpful model measurement in situations where attempting to maximize either precision or recall score results in decreased model performance. It may be conceptualized mathematically as a harmonic mean of precision and recall score as shown in equation (5):

$$F1 - Score = \frac{2 * P * R}{P + R} \quad (5)$$

4.8.6 Results Analysis:

Once the training phase is complete, the final model is evaluated using a different test set that was never used before. This enables an unbiased assessment of the model's performance in real-world scenarios. The outcomes analysis is a crucial step in the evaluation process since it offers information about how well the model performed during the training phase and how well it generalized to new data.

Loss, accuracy, precision, and recall were plotted against the number of epochs for both the training and validation sets to analyze the progress made during training. This is essential for observing the model's behavior as it learns and identifying over- or underfitting. The loss plot provides information on how well the model is optimizing its weights to reduce the error between the predicted and actual labels. A decreasing trend in the training loss indicates that the model is learning. It is important to also observe the validation loss to ensure that the model is not memorizing the training data and is generalizing well to new data.

The accuracy plot depicts the fraction of correctly classified samples. Higher accuracy signifies better performance. Precision is the fraction of true positive predictions among the positive predictions made by the model. It is important when the cost of false positives is high. Recall, on the other hand, is the fraction of true positive predictions among the actual positives. It is crucial when the cost of false negatives is high.

After training, the model with the best weights (that resulted in the lowest validation loss) was identified. The performance metrics of this model on the training and validation sets were reported, including loss, accuracy, precision, and recall. This indicates how well the model performed on the data it was trained on, as well as on new data that it did not see during training.

The confusion matrix was plotted for both the validation and test sets, which provides a visualization of the true positive, true negative, false positive, and false negative counts. This is a powerful tool for understanding the types of errors made by the model. Additionally, a classification report was generated for the validation and test sets, which includes precision, recall, F1-score, and support for both classes (Benign and Malicious). This report provides a detailed view of the model's performance for each class.

Results

In this chapter, the results of the model training, validation, and testing phases are presented. The chapter is divided into subsections representing different aspects of the results, including loss, accuracy, precision, recall, and detailed reports.

4.8.7. Model Training and Validation Performance:

This section presents the loss, accuracy, precision, and recall plots side by side, which represent the performance metrics during the training and validation process. The x-axis in each plot represents the epochs, while the y-axis represents the corresponding metric. Figure 3 shows the training loss, accuracy, precision, and recall graphs for the training and validation data.

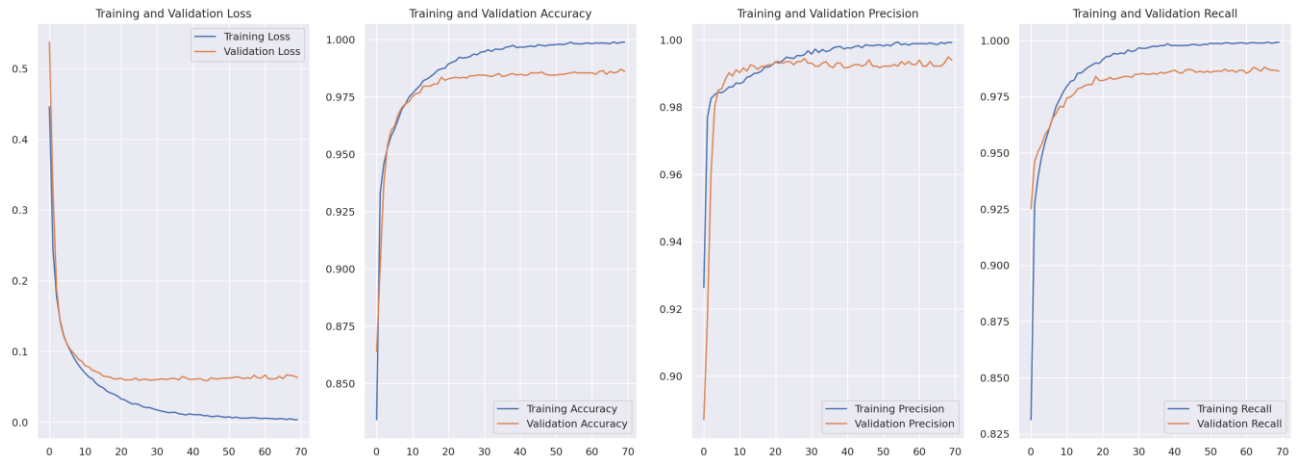


Figure 14 : The training loss, accuracy, precision, and recall graphs

The training loss, accuracy, precision, and recall graphs for the training and validation datasets.

From the plots, it can be observed that the training loss consistently decreased, and the training accuracy, precision, and recall increased over the epochs. The validation metrics also showed upward trends but with some slight fluctuations, which is expected as the model is subjected to unseen data.

4.8.7.1. Best Trained Model Performance:

The best-trained model achieved the performance metrics presented in Table 1 which indicates that the model was able to learn effectively from the training data and generalized well on the validation set. Additionally, the performance of the model with the best weights on the test set was very similar to the performance on the validation set.

Table 2. Performance metrics for the best model.

	Loss	Accuracy	Precision	Recall	F1-score
Training	0.0094	99.69%	99.77%	99.79%	99.78%
Validation	0.0586	98.55%	99.22%	98.72%	98.97%
Testing	0.0516	98.87%	99.22%	99.18%	99.20%

4.9. Detailed Classification Reports:

The confusion matrices and classification reports for the validation and testing set provide a detailed insight into how the model performed in each class. The confusion matrix shows the number of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) predictions. The confusion matrix for the model performance on the validation set is shown in Figure 15. Its corresponding classification report is shown in Table 3.

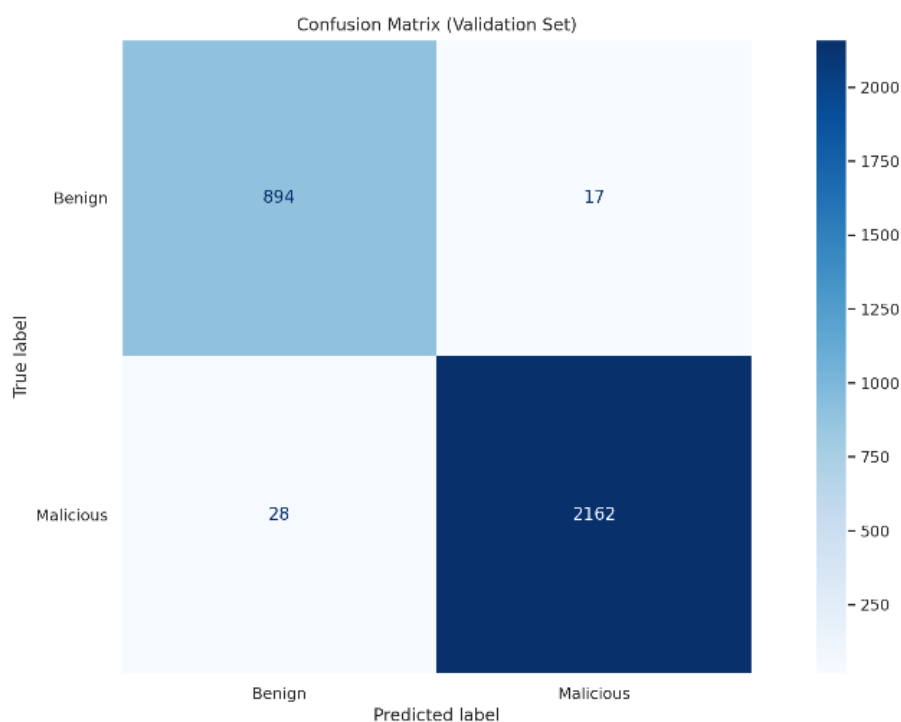


Figure 15 : Confusion matrix for the validation set.

Table 3. Classification report for the validation data.

	Precision	Recall	F1- score	Support
Benign	0.98026	0.98134	0.9808	911
Malicious	0.99223	0.99178	0.99201	2190
Accuracy			0.98871	3101
Avg	0.98625	0.98656	0.9864	3101
Weighted avg	0.98872	0.98871	0.98872	3101

Similarly, the confusion matrix and classification report for the model performance on the test set are shown in Figure 16 and Table 4.

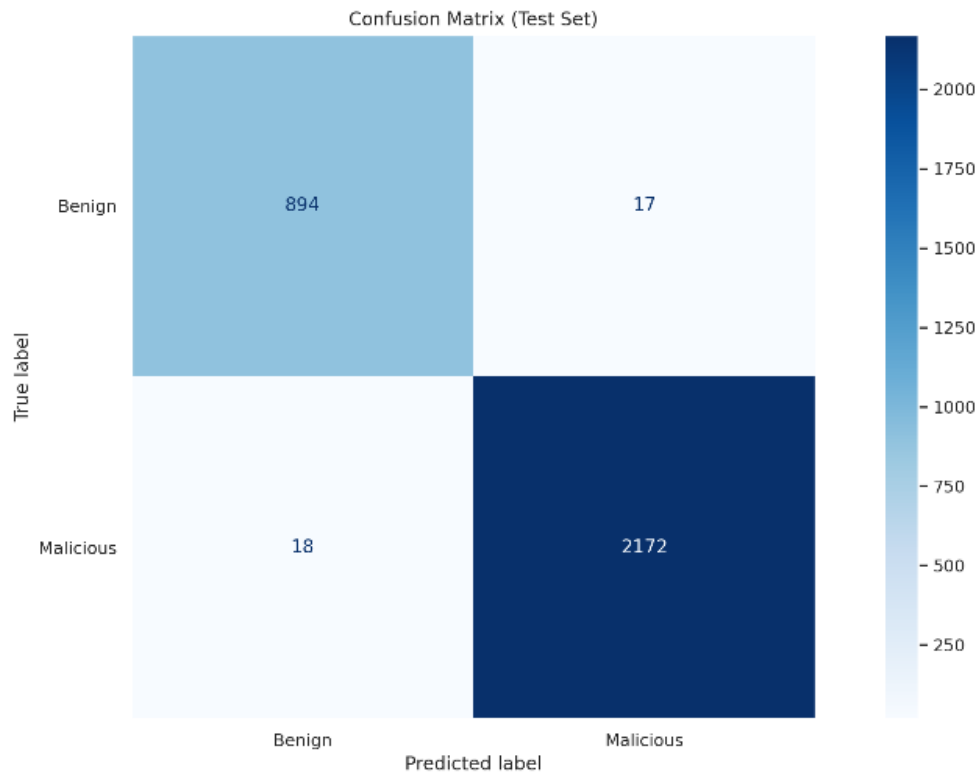


Figure 16 : Confusion matrix for the test set.

Table 4. Classification report for the test data.

	Precision	Recall	F1-score	Support
Benign	0.98026	0.98134	0.9808	911
Malicious	0.99223	0.99178	0.99201	2190
Accuracy			0.98871	3101
Macro avg	0.98625	0.98656	0.9864	3101
Weighted avg	0.98872	0.98871	0.98872	3101

4.10. Evaluation and Discussion:

During the training process, the model displayed consistent improvements in accuracy, precision, and recall metrics on the training data. However, a key observation from epochs 28 through 70 is that despite the continuing decline in loss and increasing accuracy on the training dataset, the validation loss did not display any significant improvement and rather showed signs of stagnation. The model's precision and recall on the validation set remained relatively high, which indicates its ability to correctly identify positive classes and its success in retrieving a substantial fraction of the positive instances.

The results from this study indicate that the Convolutional Neural Network (CNN) based on the EfficientNetB0 architecture is highly effective in detecting evasive malware using the CIC-Evasive-PDFMal2022 dataset. The accuracy, precision, and recall achieved are significantly high, which is in line with the findings of previous studies that have used CNNs for malware detection.

Comparatively, Yadav et al. (2022) employed an EfficientNet-B4 CNN-based model for Android malware detection and achieved an accuracy rate of 95.7%. Though the study by Yadav et al. was focused on Android malware, it resonates with the notion that EfficientNet architectures can deliver impressive results.

When considering the work of Abu Al-Haija, Odeh, and Qattous (2022), which also used the CIC-Evasive-PDFMal2022 dataset, their model achieved an accuracy of 98.84%. They employed an AdaBoost decision tree with optimal hyperparameters. Interestingly, the CNN developed in this study has achieved a comparable accuracy rate, suggesting that CNNs can be as effective as ensemble methods in this domain.

The findings of this study have implications for the development of malware detection systems. As the literature review highlighted, evasive malware is becoming more sophisticated, and traditional signature-based detection methods are increasingly ineffective. The high accuracy achieved by the CNN in this study demonstrates the viability of deep learning-based approaches to address the challenges posed by evasive malware.

Furthermore, this study also emphasizes the potential of transfer learning in improving the performance of CNNs. As discussed in the literature review, other studies such as Barros et al. (2022), who used VGG19 with transfer learning, and Ahmet et al. (2019), who experimented with various CNN architectures, found that transfer learning significantly improved model performance. This study adds to that body of evidence by successfully applying transfer learning using the EfficientNetB0 architecture.

One of the theoretical contributions of this study is that it supports the hypothesis that deep learning-based approaches can effectively address the challenges posed by evasive malware. Additionally, this study demonstrates the efficacy of transfer learning, which has been less explored in the context of evasive PDF malware detection, especially with the EfficientNet architecture.

Moreover, by achieving high performance with the EfficientNetB0 architecture, this study contributes to the ongoing discussion about the effectiveness of different CNN architectures for malware detection. This adds another dimension to the findings of Vasan et al. (2020) and Kumar et al. (2021), who used architectures like VGG16 and ResNet50.

Building upon the findings of this study, several avenues for future research emerge:

- **Testing Various Architectures:** The EfficientNet architecture performed well in this study, but testing other EfficientNet variants and comparing them to architectures like VGG or ResNet could lead to a more optimized model for evasive malware detection.
- **Feature Interpretation and Extraction:** Understanding which features are most indicative of malware could help in building more interpretable models and improving feature extraction techniques, as called for by Aboaoja et al. (2022).
- **Real-World Implementation:** Assessing the performance of the model in real-world environments, as well as integrating it with existing cybersecurity systems, could provide insights into its practicality and limitations.
- **Combating New Evasion Techniques:** With malware continuously evolving, studying new evasion techniques, and adapting models accordingly is crucial.
- **Expanding Datasets:** Experimenting with a more diverse set of PDF files, including those with new malicious techniques, could help in evaluating and enhancing the model's robustness.
- **Multi-Modal Approaches:** Exploring models that combine CNNs with other machine learning or deep learning techniques, such as recurrent neural networks, for potentially enhanced performance.

This chapter discussed the performance and implications of a CNN based on the EfficientNetB0 architecture for evasive PDF malware detection. The model's

performance was highly accurate and comparable to other state-of-the-art methods. This validates the hypothesis that deep learning, particularly CNNs with transfer learning, can effectively detect evasive malware. The results contribute to the broader context of malware detection research and suggest promising directions for future studies.

4.11.Conclusion:

In this study, we have presented a novel approach to detecting evasive PDF malware by leveraging deep learning techniques, specifically utilizing EfficientNet, a state-of-the-art convolutional neural network model. The research was conducted using an extensive and unique dataset of 31,006 raw PDF files, which were transformed into images to enhance the model's ability to identify subtle malicious patterns that traditional methods often overlook.

4.11.1. Summary of Objectives and Accomplishments:

The primary objective of this research was to explore the potential of transfer learning and deep learning, particularly EfficientNet, in enhancing the detection of evasive PDF malware. The following accomplishments highlight the significant contributions of this study:

1. **Innovative Data Transformation:** The research introduced a novel methodology of converting raw PDF files into binary files and subsequently into images. This transformation enabled the application of powerful image recognition capabilities of CNNs to the domain of malware detection, a departure from traditional feature extraction methods.
2. **Leveraging EfficientNet:** By employing the EfficientNetB0 model, known for its efficiency and high performance, the study demonstrated the model's capability in handling the computational intensity associated with large-scale data processing. EfficientNet's compound scaling methodology was particularly beneficial in balancing depth, width, and resolution, thereby enhancing detection efficiency while curtailing computational costs.
3. **Application of Transfer Learning:** The study effectively utilized transfer learning, leveraging pre-trained EfficientNet models on ImageNet to accelerate the training process and enhance the model's learning

efficiency. This approach reduced the dependency on large, domain-specific labeled datasets, which are often challenging to procure.

4. **High Detection Accuracy:** The model achieved outstanding results, showcasing an accuracy of 98.87% and an F1-score of 99.20% on the test set. These metrics indicate the model's exceptional precision and recall capabilities, consistently surpassing traditional malware detection techniques that rely on hand-crafted features and are more susceptible to manipulation by sophisticated malware.
5. **Simplification of Detection Process:** By obviating the manual extraction of features, the proposed method simplified the detection procedure and diminished potential human error. This automation is crucial in the context of rapidly evolving cyber threats, where manual feature engineering can be both time-consuming and prone to oversight.
6. **Broad Implications:** The implications of this research extend beyond malware detection to encompass broader cybersecurity concerns, such as intrusion detection, phishing detection, and ransomware identification. The study sets a fresh precedent in the realm of machine learning-based malware detection, opening new vistas for future research and development in cybersecurity.

4.11.2. Detailed Findings and Implications:

The results from this study validate the hypothesis that deep learning-based approaches, particularly those utilizing CNNs with transfer learning, can effectively detect evasive malware. The success of the EfficientNetB0 model in this domain is attributed to its depth-wise separable convolutions and compound scaling methodology, which significantly enhance efficiency and performance.

The study's findings contribute to the ongoing discussion about the effectiveness of various CNN architectures for malware detection. By achieving high performance with EfficientNetB0, this research adds another dimension to the findings of previous studies that used architectures like VGG16 and ResNet50, further underscoring the potential of EfficientNet in cybersecurity applications.

Moreover, the study highlights the potential of transfer learning in improving the performance of CNNs. As discussed in the literature review, other studies have found that transfer learning significantly improves model performance. This study corroborates those findings, demonstrating that transfer learning,

when applied using EfficientNet, can be highly effective in the context of evasive PDF malware detection.

4.11.3. Challenges and Future Directions:

While the study achieved remarkable results, several challenges and areas for future research have been identified:

1. **Adversarial Attacks:** As cybercriminals employ adversarial machine learning techniques to craft malware that evades detection models, future research should focus on developing robust adversarial defenses to enhance model resilience.
2. **Data Quality and Availability:** High-quality, labeled datasets are crucial for training accurate models. Expanding datasets to include more diverse and updated samples of PDF files with new malicious techniques can help in evaluating and enhancing the model's robustness.
3. **Model Interpretability:** Deep learning models, often perceived as black boxes, lack interpretability. Future research should aim to develop methods to improve the transparency of these models, making it easier to understand which features were most influential in the decision-making process.
4. **Combining Techniques:** Integrating convolutional neural networks with traditional machine learning algorithms that use extracted features could harness the benefits of both approaches, potentially leading to even more effective detection systems.
5. **Testing Various Architectures:** While EfficientNet performed well in this study, exploring other variants of EfficientNet and comparing them to architectures like VGG or ResNet could lead to more optimized models for evasive malware detection.
6. **Real-World Implementation:** Assessing the performance of the model in real-world environments and integrating it with existing cybersecurity systems could provide insights into its practicality and limitations.
7. **Combating New Evasion Techniques:** With malware continuously evolving, studying new evasion techniques and adapting models accordingly is crucial for maintaining robust defenses.

In conclusion, this study provides a groundbreaking approach to detecting evasive PDF malware through the innovative use of raw data transformation and deep learning techniques. The successful application of EfficientNetB0, combined with transfer learning, demonstrates the potential for deep learning algorithms to capture complex patterns and adapt to evolving malware techniques, offering a more robust and scalable solution to malware detection. As the landscape of cyber threats continues to evolve, so too must the technologies and strategies employed to detect and mitigate these threats. The approach explored in this study represents a promising avenue for future research and development in cybersecurity, contributing to the creation of more effective, adaptable, and comprehensive malware detection systems.

The significant accomplishments of this research underscore the importance of continuous innovation and adaptation in the field of cybersecurity, ensuring safer digital environments for all.



CHAPTER (5)

WEB DEVELOPMENT

Chapter 5 : Web Development

5.1. Abstract:

This Section provides a comprehensive analysis of the requirements and architecture for a website system designed to analyze PDF files for potential threats. The system's functionality encompasses user registration, login, PDF uploads, payment processing, and interaction with a machine learning model to determine if the uploaded files are benign or malicious. Key sections include functional and non-functional requirements, use case and sequence diagrams, system architecture involving MySQL, Django, and React, and a detailed description of the technologies and tools used. The report concludes with testing scenarios for user sign-up, login, file uploading, and payment processes to ensure the system's reliability, usability, and security.

5.2. Introduction:

The purpose of this report is to outline the development and deployment of a website system aimed at providing users with the capability to upload and analyze PDF files to determine their safety. This system is essential for organizations and individuals who need to verify the security of their documents in an efficient and user-friendly manner.

5.3. System Overview:

5.3.1. User Interaction:

- Users must register and log in to access the system's features.
- Upon logging in, users can upload PDF files to be analyzed.
- After three free uploads, users are prompted to purchase additional upload credits through the system.

5.3.2. Functional Components:

- **User Authentication:** Secure sign-up and login processes.
- **PDF Upload and Analysis:** Users upload files which are then analyzed by a machine learning model to determine if they are benign or malicious.
- **Payment Processing:** Supports transactions via credit card and bank transfer for purchasing additional upload credits.

5.3.3.Non-Functional Requirements:

- **Availability and Usability:** The site is accessible at all times with an intuitive interface.
- **Portability and Reliability:** Compatible with various devices and ensures reliable operation.
- **Security:** Protects user data and ensures secure transactions.

5.3.4. System Architecture:

- Utilizes MySQL for data storage, Django for server-side operations, and React for the front-end interface.
- Employs Django REST Framework (DRF) for API development and JSON Web Tokens (JWT) for secure user authentication.

5.3.5. Technologies and Tools:

- **Database Management:** MySQL with ACID compliance and transactional support.
- **Backend Framework:** Django, known for its scalability, security, and extensive built-in features.
- **Frontend Development:** React.js for a dynamic and responsive user interface.
- **Version Control and Collaboration:** Git and GitHub for efficient project management and collaboration.

5.3.6.Testing and Validation:

- Detailed test cases for user registration, login, PDF upload, and payment processes to ensure the system meets the specified requirements and provides a seamless user experience.

This report will delve into each of these areas, providing a detailed analysis of the system's requirements, architecture, and implementation, supported by diagrams and test scenarios to illustrate the system's functionality and performance.

5.4. Analysis and Requirements

5.4.1. System Requirements

5.4.1.1. Function Requirements:

Table 5: Function Requirements

Requirements	Description
Sign up	Users should make an account before uploading any PDF to check on Requirements: Enter all required credentials
Log in	To be authenticated the user should enter the Email and Password Requirements: Enter Email and Password
Data Storage	The system must store user profile information, including name, email, password, upload history purchase history.
Payment Processing	The system must support payment transactions via credit card, and bank transfer.
Upload pdf file	The user must upload the pdf file to get the response if is benign or malicious

5.4.1.2-Non-Functional Requirements:

Table 6: Non-Function Requirements

Requirements	Description
Availability	The site is available always
Usability	We made an easy user interface so anyone can use the site all they need just to register and just upload
Portability	Can open it on any device such as (phone – pc)
Reliability	All functions work well and there are no fails
Security	Users' data is secure no-one can access it

5.4.1.3. Functional Requirements Specifications

5.4.1.3.1 Stakeholders:

- User
- Admin

5.4.1.3.2. Actors:

- **User:** Uploads the PDF file and views the result.
- **System(Malero):** Processes the PDF, tracks upload counts and interacts with the ML model.
- **ML Model:** Analyzes the PDF and determines if it's benign or malicious.

5.4.2. Use Cases

5.4.2.1. Upload PDF:

- **Primary Actor:** User
- **Goal:** To upload a PDF file for analysis.
- **Main Success Scenario:**
 1. User logs into the system.
 2. The user navigates to the upload section.
 3. The user selects a PDF file to upload.
 4. The user submits the PDF file.
 5. The system confirms the receipt of the PDF file.
 6. The system checks the number of uploads by the user.
 7. If the user has reached three uploads, proceed to Prompt for Purchase.
- **Alternative Flows:**
 8. If the file format is not PDF, the system displays an error message.
 9. If the user is not authenticated, the system prompts for login.

5.4.2.2. Process PDF:

- **Primary Actor:** System (Malero)
- **Goal:** To send the uploaded PDF to the ML model for analysis.
- **Main Success Scenario:**
 1. The system receives the uploaded PDF.
 2. The system sends the PDF to the ML model.
 3. The system waits for the analysis result from the ML model.

- **Alternative Flows:**

1. If the system fails to send the PDF, it retries up to three times before notifying the user of an error.

5.4.2.3. Analyze PDF:

- **Primary Actor:** ML Model
- **Goal:** To analyze the PDF and determine if it is benign or malicious.
- **Main Success Scenario:**

1. The ML model receives the PDF from the system.
2. The ML model processes the PDF.
3. The ML model determines the status (benign or malicious).
4. The ML model sends the result back to the system.

- **Alternative Flows:**

5. If the ML model encounters an error during analysis, it logs the error and notifies the system.

5.4.2.4. Display Result:

- **Primary Actor:** System (Malero)
- **Goal:** To display the analysis result to the user.
- **Main Success Scenario:**

1. The system receives the analysis result from the ML model.
2. The system displays the result on the user's screen.
3. The user views the result (benign or malicious).

- **Alternative Flows:**

4. If the system fails to display the result, it logs the error and notifies the user.

5.4.2.5. Prompt for Purchase:

- **Primary Actor:** System (Malero)
- **Goal:** To prompt the user to purchase a package after three free uploads.
- **Main Success Scenario:**

1. System checks the number of uploads by the user.
2. If the user has uploaded three PDFs, the system displays a prompt.

3. The prompt offers the user to purchase the Gold or Diamond package.

- **Alternative Flows:**

4. If the user selects a package, proceed with the purchase process.
5. If the user declines, restrict further uploads.

5.4.3. Use case diagram

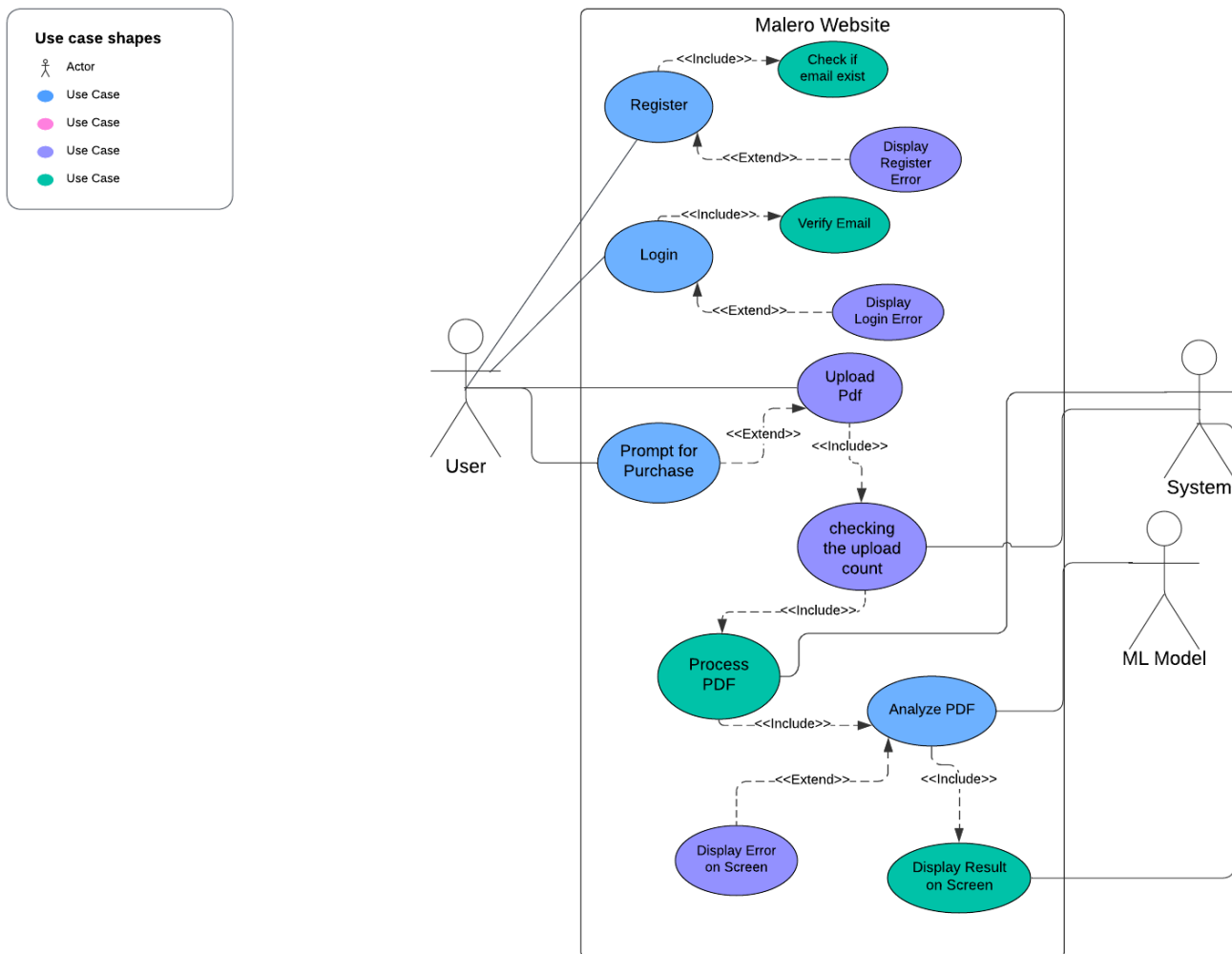


Figure 17: Use Case Diagram

5.4.4. Sequence diagram

5.4.4.1. User's Register

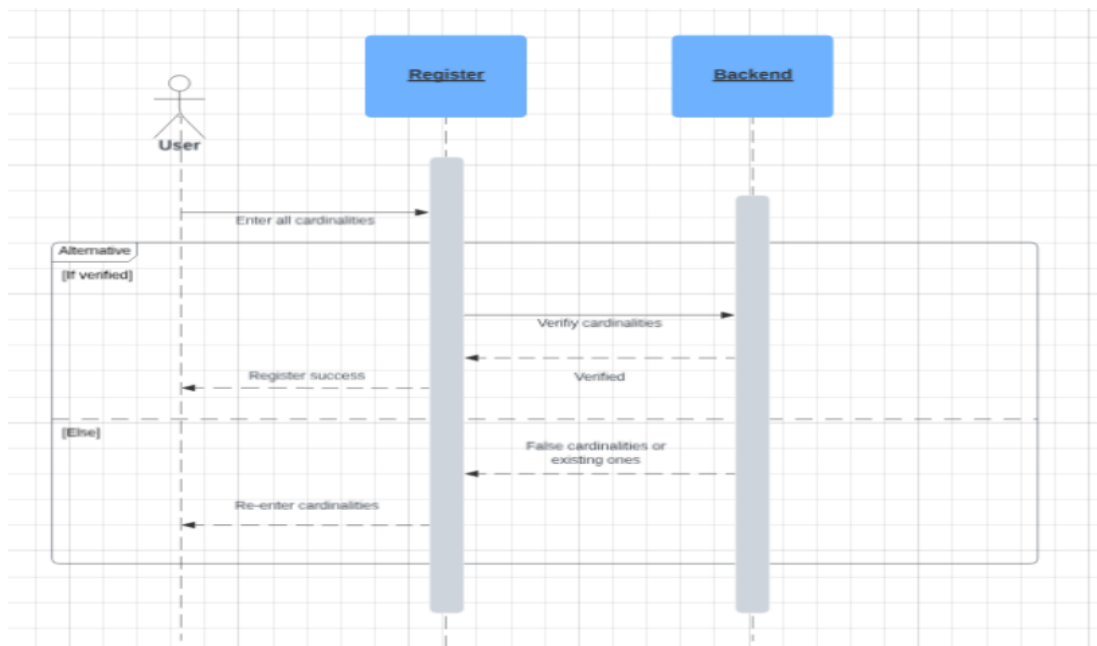


Figure 18 : User's Register Sequence diagram

5.4.4.2. USER'S LOGIN

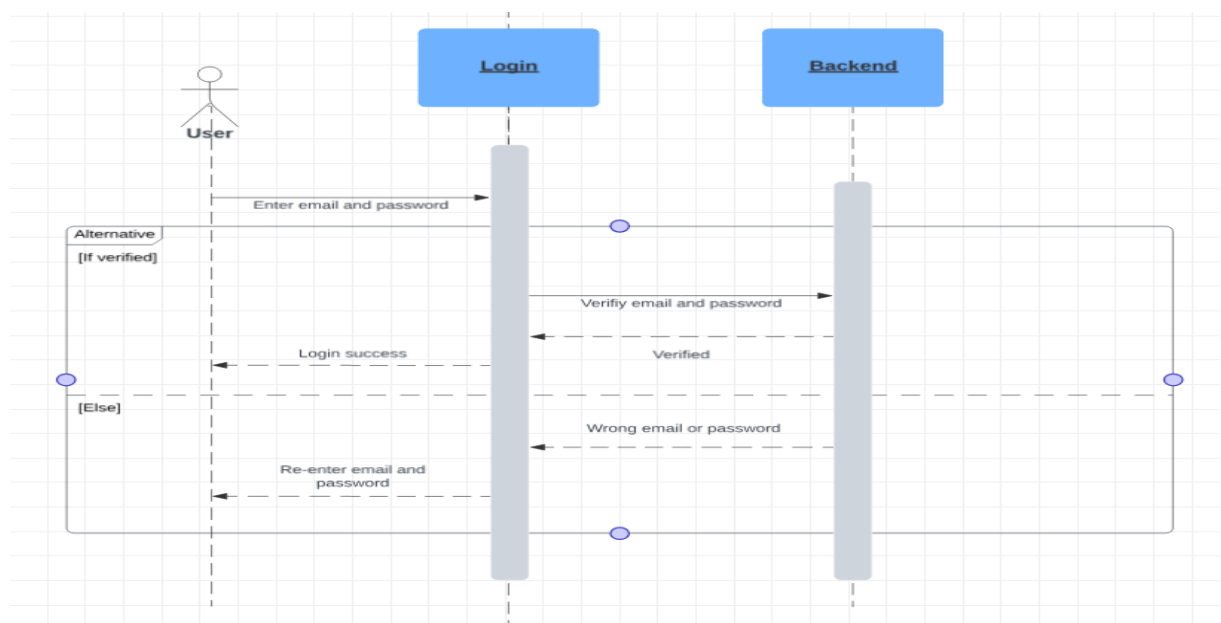


Figure 19 : User's Login Sequence diagram

5.4.4.3. User's Payment

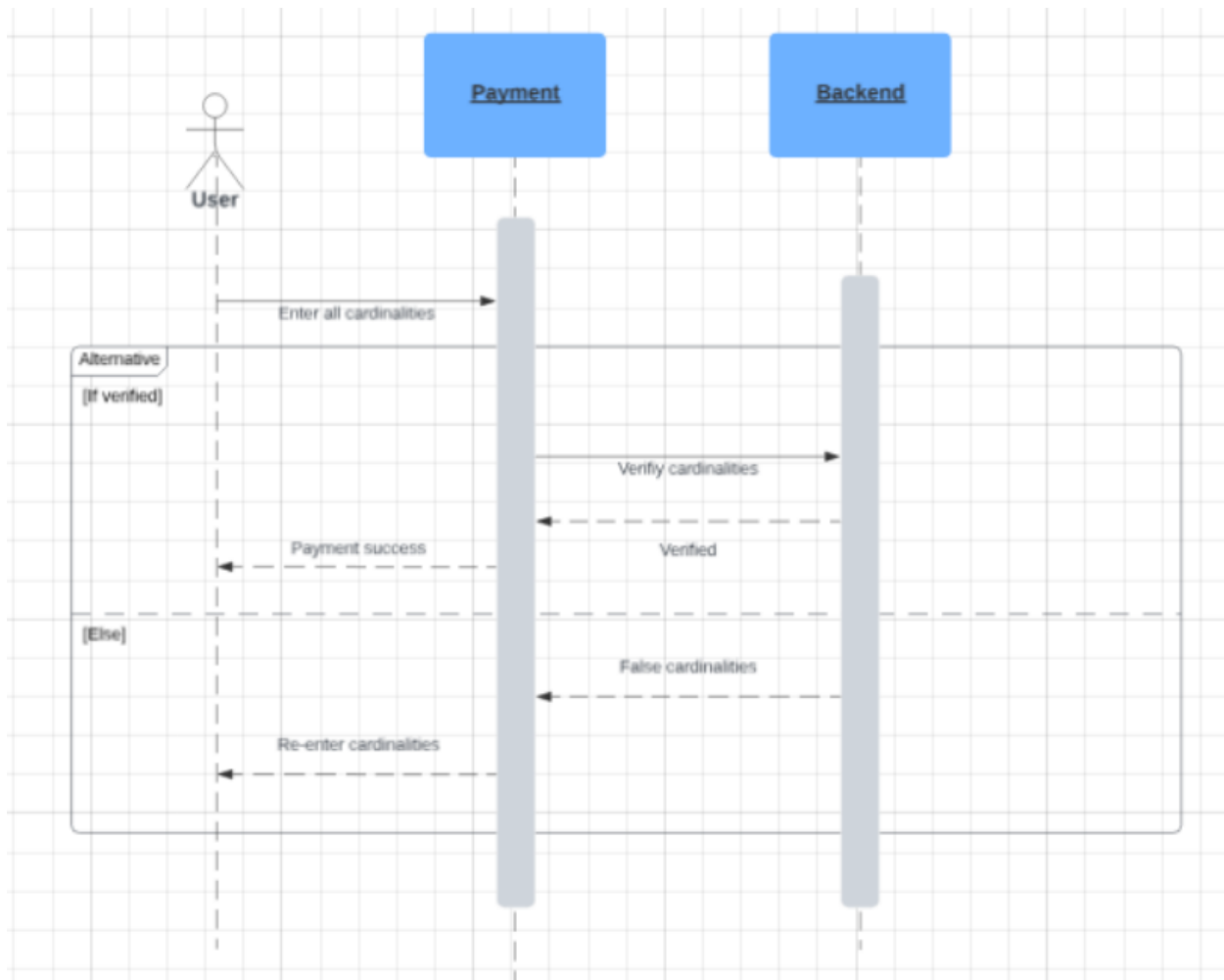


Figure 20 : User's Payment Sequence diagram

5.4.4.4. Upload PDF

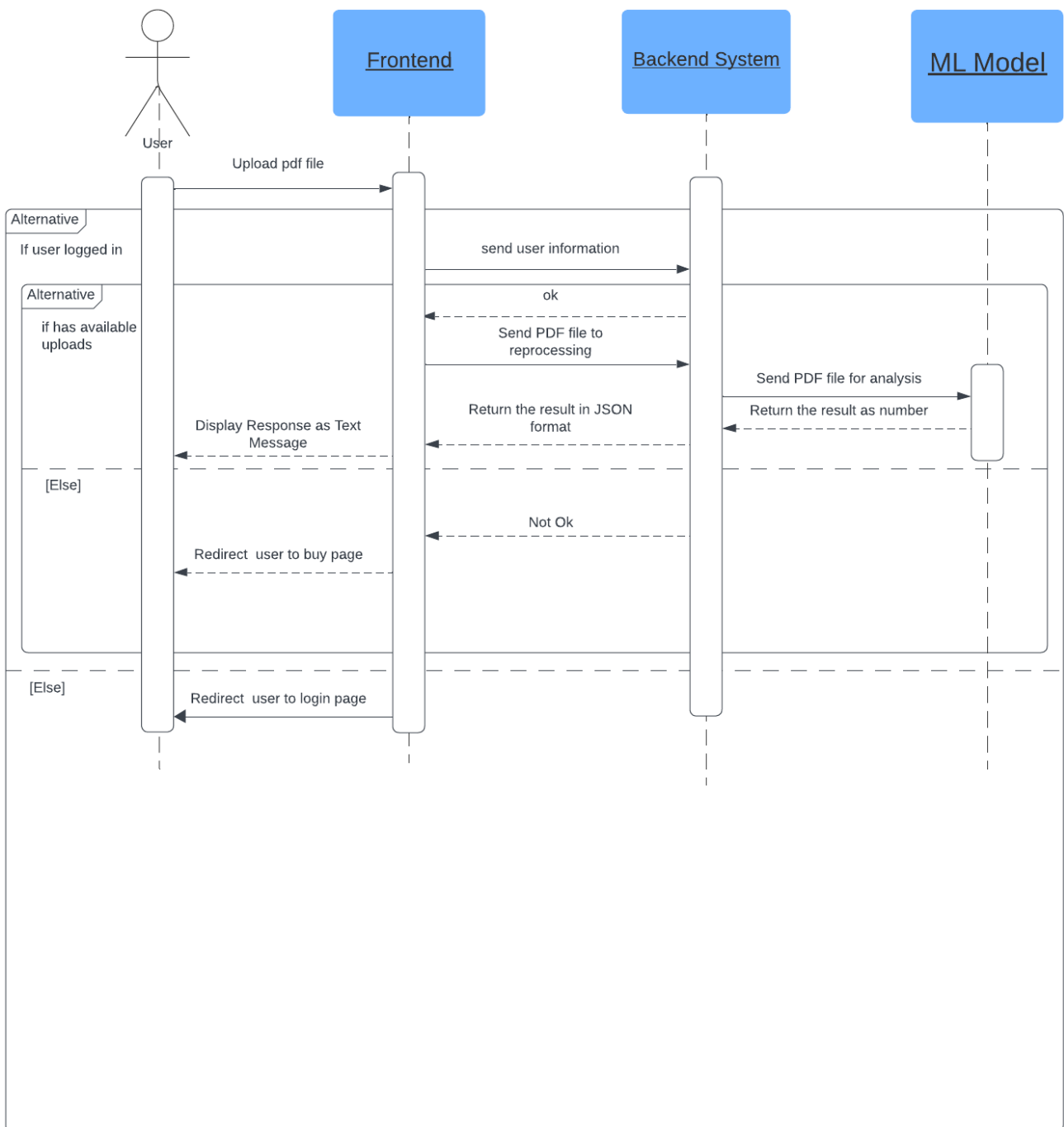


Figure 21: Upload PDF Sequence diagram

5.4.5. ERD Diagram

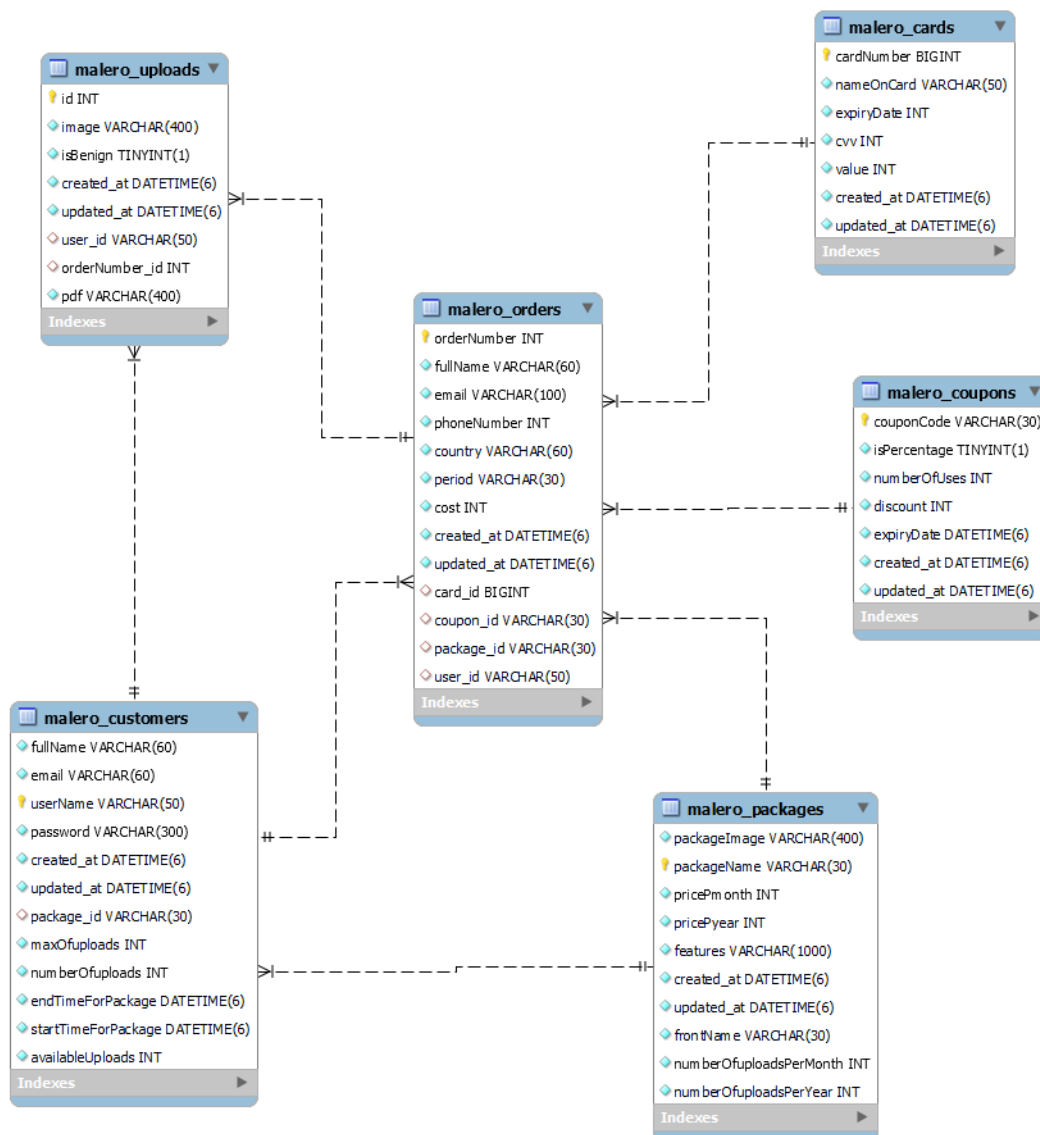


Figure 22: ERD diagram

5.4.6. Flow Chart Diagram:

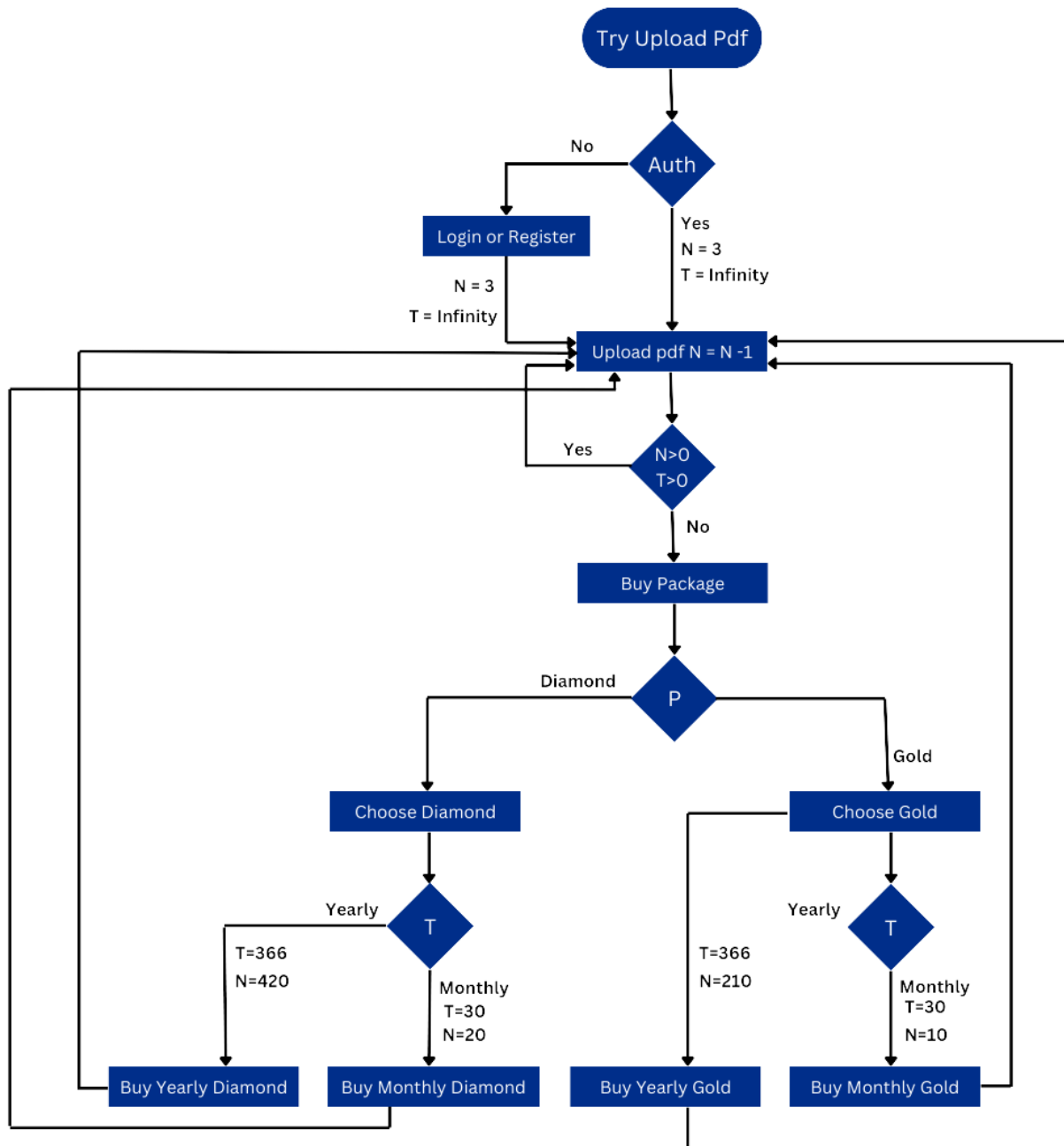


Figure 23: Flow Chart diagram

The Flow of the system starts with the user trying to upload a pdf file to check if the file is Bingen or malicious, If the user has already login to the website will take three times to upload for free but if the user did not registered the website will redirect him/his to the register page and after registration will redirect to the home page.

If the user uploads three times and tries to upload another PDF the website will show the message "You can't upload more update your package".

The user has two options to buy any of two packages Gold or Diamond and each one has two options monthly or yearly so there are four options.

5.4.6.1.The First Option:

If the user chooses **the monthly gold package** will take **10 times** uploads and can use them through one month if the number of uploads ends before the month the website will force the user to update the package or buy another package, but if the user not used all numbers of uploads the rest will add to the next month.

5.4.6.2. The Second Option:

If the user chooses **the monthly gold package** will take **210 times** uploads can be used over one year if the number of uploads ends before the year the website will force the user to update the package or buy another package, but if the user has not used all numbers of uploads the rest will add to the next year.

5.4.6.3. The Third Option:

If the user chooses **the yearly diamond package** will take 20 times uploads can be used through one month if the number of uploads ends before the month the website will force the user to update the package or buy another package, but if the user has not used all numbers of uploads the rest will add to the next month.

5.4.6.4. The Fourth Option:

If the user chooses **the yearly diamond package** will take **420 times** uploads can be used through one year if the number of uploads ends before the year the website will force the user to update the package or buy another package, but if the user has not used all numbers of uploads the rest will add to the next year.

5.4.7. System Architecture

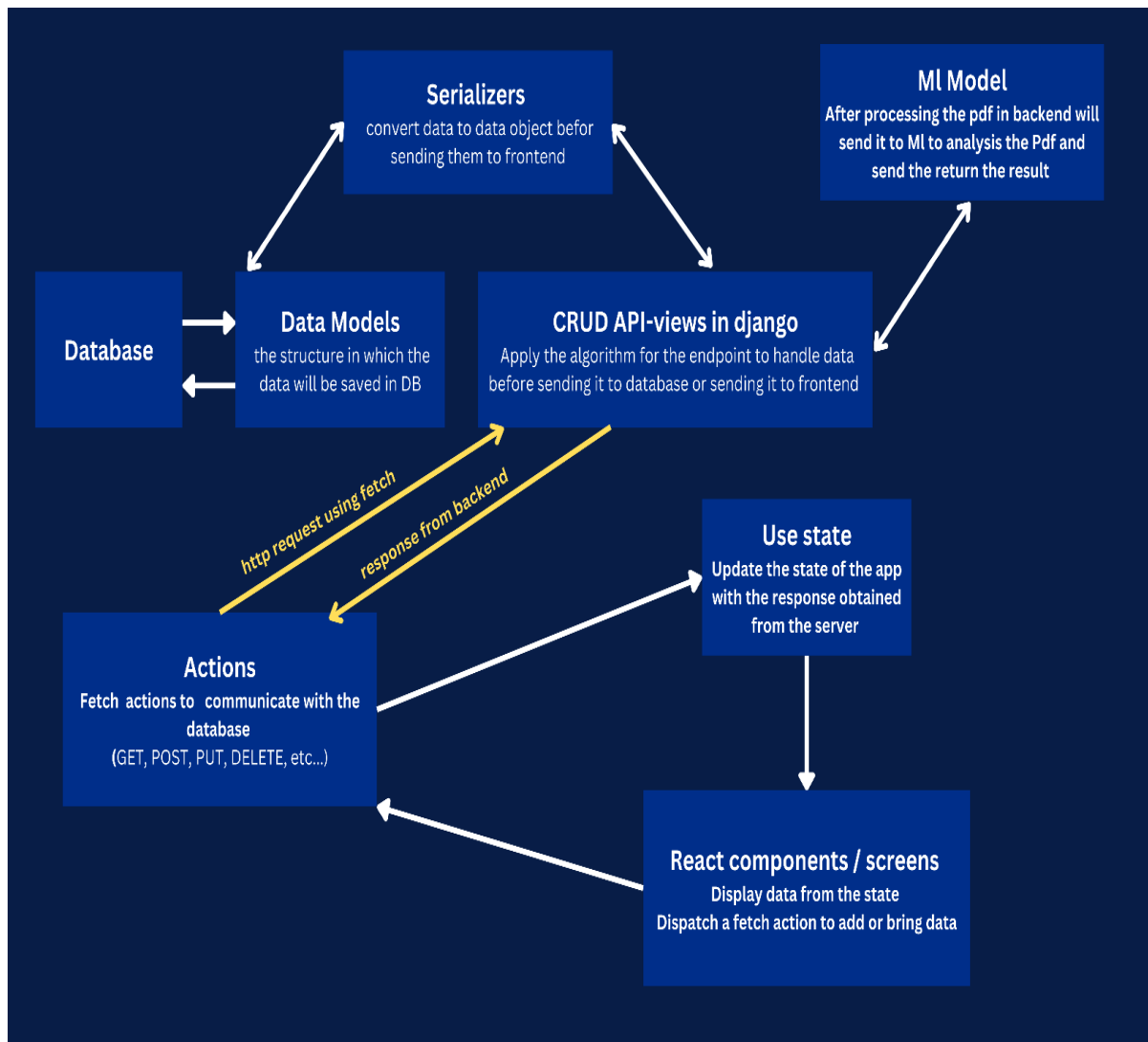


Figure 24 : System Architecture diagram

5.4.7.1. Database:

MySQL Database with MySQL Workbench

Why MySQL?

MySQL is a relational database management system (RDBMS) that offers several advantages for application development:

- **ACID Compliance:** MySQL supports Atomicity, Consistency, Isolation, and Durability (ACID) properties, ensuring that all database transactions

are processed reliably. This guarantees data integrity and consistency, even in the event of system failures.

- **Transactional Support:** MySQL's transactional capabilities make it suitable for applications requiring strong consistency and precise data manipulation. Transactions allow multiple operations to be executed as a single unit of work, which can be rolled back if any part of the transaction fails.
- **Scalability and Flexibility:** MySQL can handle large databases efficiently and can be scaled horizontally or vertically, accommodating growing data requirements.
- **Open Source:** As an open-source database, MySQL is cost-effective and has a large community of developers contributing to its improvement and providing support.

5.4.7.2. Data Models: Schema

A schema in MySQL defines the structure and organization of the data:

- **Tables and Columns:** The schema outlines the tables and columns that store the data. Each table represents a collection of related data entries, while columns define the attributes of the data.
- **Relationships:** It specifies the relationships between different tables, such as one-to-one, one-to-many, and many-to-many relationships. This relational structure ensures data integrity through foreign keys and constraints.
- **Data Types and Constraints:** The schema also defines the data types for each column (e.g., INTEGER, VARCHAR, DATE) and any constraints (e.g., PRIMARY KEY, NOT NULL) that ensure data validity and consistency.

5.4.7.3. Serializers:

Serializers play a crucial role in data transformation between the backend and frontend:

- **Data Formatting:** Before sending data to the frontend (GET request), serializers convert complex data types like objects, query sets, or models into JSON or other formats that the frontend can easily understand and use.
- **Data Validation:** When data is sent from the frontend to the backend (POST request), serializers validate and transform this data into

appropriate backend data structures. This ensures that the incoming data adheres to the expected format and constraints.

- **Framework Compatibility:** Serializers bridge the gap between backend frameworks (like Django, and Flask) and frontend frameworks (like React, and Angular), facilitating seamless data exchange.

5.4.7.4. API Views: Controllers:

API views, also known as controllers, manage the logic and processing of incoming and outgoing data:

- **CRUD Operations:** Controllers handle Create, Read, Update, and Delete operations, ensuring that data is manipulated correctly and efficiently.
- **Business Logic:** They apply specific business rules and algorithms to process data before interacting with the database or returning responses to the client.
- **Endpoint Management:** Controllers define endpoints for different API actions, mapping HTTP requests (GET, POST, PUT, DELETE) to corresponding functions that execute the necessary operations.

5.4.7.5. Actions:

Actions are essential for data communication between the frontend and backend:

- **Fetch Actions:** These functions are responsible for making HTTP requests to the backend to retrieve (GET), send (POST), update (PUT), or delete (DELETE) data.
- **Data Handling:** Fetch actions manage the data flow, ensuring that the data is correctly formatted and processed before being sent or after being received.
- **Error Handling:** They also include mechanisms for handling errors, ensuring that any issues during data transmission are appropriately managed and communicated to the user.

5.4.7.6. Use State:

State management is a core concept in modern web applications:

- **State Synchronization:** The state represents the current data and status of the application. Fetch actions update the state with data received from the server, ensuring the frontend accurately reflects the latest information.

- **Reactivity:** Changes in state trigger re-renders of components, ensuring the user interface (UI) remains dynamic and responsive to user interactions.
- **State Libraries:** Libraries like Redux or Context API are often used to manage complex state logic and provide a centralized state store accessible across the application.

5.4.7.7. React Components:

React components are the building blocks of a React application:

- **Data Display:** Components fetch data from the state and render it in the UI, providing a visual representation of the data to the user.
- **User Interaction:** They handle user interactions, such as clicks or form submissions, by dispatching fetch actions to update or retrieve data.
- **Component Lifecycle:** React components have a lifecycle (mounting, updating, unmounting) that allows developers to hook into different phases and perform necessary operations, such as fetching data when a component mounts.

5.4.7.8. ML Model:

The Machine Learning (ML) model is a sophisticated and essential component for analyzing PDF files and determining their safety.

The ML model is trained using a comprehensive dataset of both benign and malicious PDF files. This training process involves feeding the model labeled data, allowing it to learn distinguishing patterns and features that differentiate safe files from harmful ones. Advanced algorithms, such as deep learning, are employed to ensure the model can identify complex and subtle threats.

5.4.8. Used Technologies and Tools

5.4.8.1. Server: Django:

Django is a high-level Python web framework designed to encourage rapid development and clean, pragmatic design. Key features of Django include:

- **Batteries-Included Philosophy:** Django comes with a lot of built-in features that streamline the development process. This includes an ORM (Object-Relational Mapping) system that abstracts database interactions

into Python objects, built-in authentication mechanisms for user management, and an admin panel for managing application data.

- **Security:** Django provides robust security features out of the box, such as protection against SQL injection, cross-site scripting, and cross-site request forgery, ensuring that the application is secure.
- **Scalability:** Django is designed to help developers build scalable web applications. Its modularity allows for the separation of different components, making it easier to manage and scale the application as it grows.
- **Community and Documentation:** Django has a large, active community and extensive documentation, making it easier to find support, tutorials, and third-party packages to extend the functionality of the framework.

5.4.8.2. Database: MySQL:

MySQL is a popular relational database management system known for its reliability, scalability, and performance. Here's how it integrates with Django:

- **Data Storage and Management:** MySQL is used to store and manage the data for the web application. It supports ACID transactions, ensuring data integrity and reliability.
- **Django ORM:** Django's ORM allows developers to interact with the MySQL database using Python code instead of writing raw SQL queries. This abstraction layer makes database operations more intuitive and reduces the risk of SQL injection attacks.
- **Scalability and Performance:** MySQL is designed to handle large volumes of data and high traffic loads, making it suitable for scalable web applications. It supports features like indexing, replication, and partitioning to optimize performance.

5.4.8.3. API: Django REST Framework:

Django REST Framework (DRF) is a powerful and flexible toolkit for building Web APIs in Django. Its key features include:

- **Serializers:** DRF provides serializers to convert complex data types like Django models into JSON, making it easy to create APIs that can communicate with front-end applications.
- **View sets and Routers:** These components simplify the process of defining and managing API endpoints. View sets allow developers to

define common API behaviors in a single class, while routers handle URL routing automatically.

- **Authentication and Permissions:** DRF includes built-in authentication mechanisms (e.g., token-based authentication) and a flexible permissions system to secure API endpoints and control access based on user roles and permissions.
- **Browsable API:** DRF offers a browsable API interface, which is useful for testing and debugging APIs directly from the browser.

5.4.8.4. Frontend: React.js

React.js is a powerful JavaScript library for building dynamic and interactive web applications. It provides:

- **Component-Based Architecture:** React promotes the development of reusable UI components, which can be combined to create complex user interfaces. This modular approach improves code maintainability and reusability.
- **Virtual DOM:** React uses a virtual DOM to efficiently update and render components, leading to better performance and a smoother user experience.
- **Declarative Syntax:** React's declarative syntax makes it easier to understand and manage the application state and UI logic.
- **Ecosystem and Community:** React has a rich ecosystem of libraries and tools (e.g., Redux for state management, React Router for routing) and a large community, providing ample resources and support.

5.4.8.5. Styling: CSS

CSS (Cascading Style Sheets) is used to style the web application, ensuring it has a visually appealing and user-friendly interface:

- **Layout and Design:** CSS is used to control the layout, color schemes, fonts, and overall design of the application, ensuring a consistent and aesthetically pleasing look across all pages.
- **Responsive Design:** CSS techniques like media queries enable responsive design, ensuring that the application looks and functions well on various devices and screen sizes.
- **Animations and Transitions:** CSS allows for the creation of smooth animations and transitions, enhancing the user experience by making interactions more engaging.

5.4.8.6. Version Control: Git

Git is a distributed version control system that enables efficient project management and collaboration:

- **Tracking Changes:** Git tracks changes to the codebase over time, allowing developers to review, revert, and manage changes effectively.
- **Branching and Merging:** Git supports branching and merging, enabling multiple developers to work on different features or fixes simultaneously without interfering with each other's work.
- **Collaboration:** Git facilitates collaboration by allowing developers to share their code, review each other's changes, and merge contributions seamlessly.

5.4.8.7. Collaborative Platform: GitHub

GitHub is a web-based platform that utilizes Git for version control and collaborative software development:

- **Repositories:** GitHub hosts project repositories, providing a central place to store and manage the codebase.
- **Collaboration Tools:** GitHub offers tools like pull requests, code reviews, and issue tracking to facilitate collaboration and project management.
- **CI/CD Integration:** GitHub integrates with continuous integration and continuous deployment (CI/CD) pipelines, automating testing, and deployment processes.
-

5.4.8.8. Authentication: JWT (JSON Web Tokens)

JWT (JSON Web Tokens) are used to implement secure user authentication:

- **Token-Based Authentication:** JWTs are used to securely transmit information between the client and server. When a user logs in, the server generates a token that the client stores and sends with subsequent requests to authenticate the user.
- **Stateless Authentication:** JWTs allow for stateless authentication, meaning the server does not need to store session information. This improves scalability and performance.
- **Endpoint Security:** JWTs are used to protect endpoints like /api/register and /api/login, ensuring that only authenticated users can access certain resources or perform specific actions.

5.4.9. Front-end flow-chart

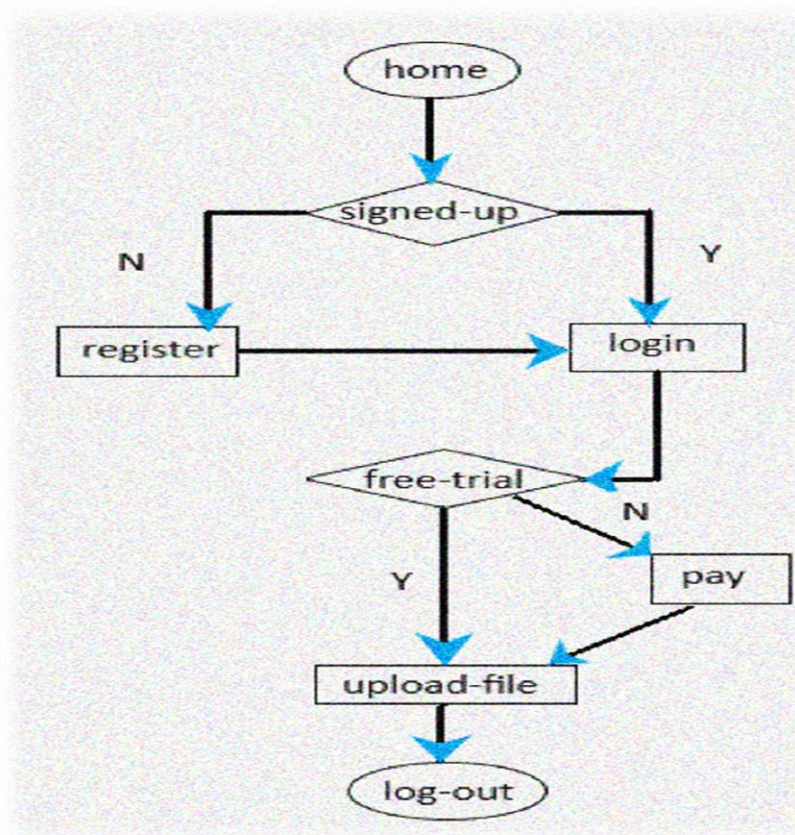


Figure 25 : Front-end flow chart

Initially, the user goes to the home page and attempts to upload files without logging in. The site informs the user that they must be authenticated to upload files. The user is then redirected to the registration page to create an account. After registering and logging in, the user can upload a maximum of 3 PDFs during their free trial. Once the free trial ends, the user must purchase one of the available packages (Gold or Diamond) to upload more PDFs. After successfully purchasing a package, the user can upload additional PDFs. Once the user finishes uploading the desired files, they can log out.

5.4.10. Snapshots for the results:

5.4.10.1. Case 1: Upload without login:

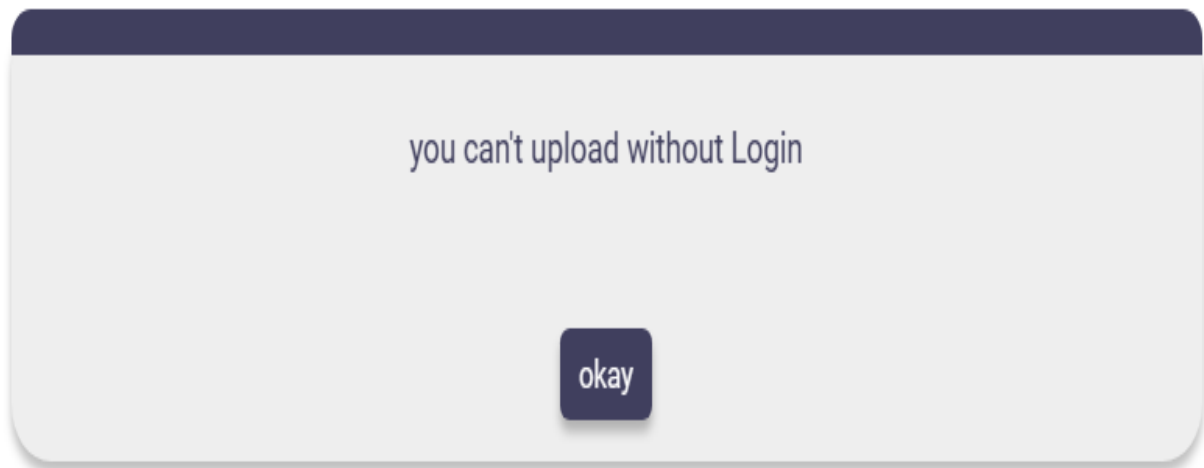


Figure 26 : Upload without login

5.4.10.2. Case 2: Upload with login and the pdf is benign:

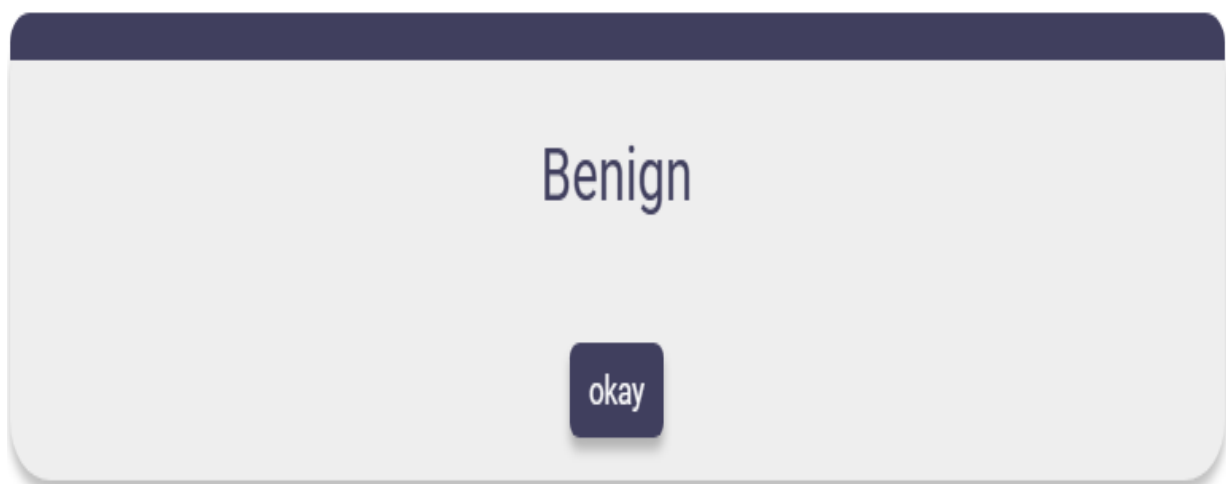


Figure 27 : Upload with login and the pdf is benign

5.4.10.3. Case 3: Upload with login and the pdf is benign:

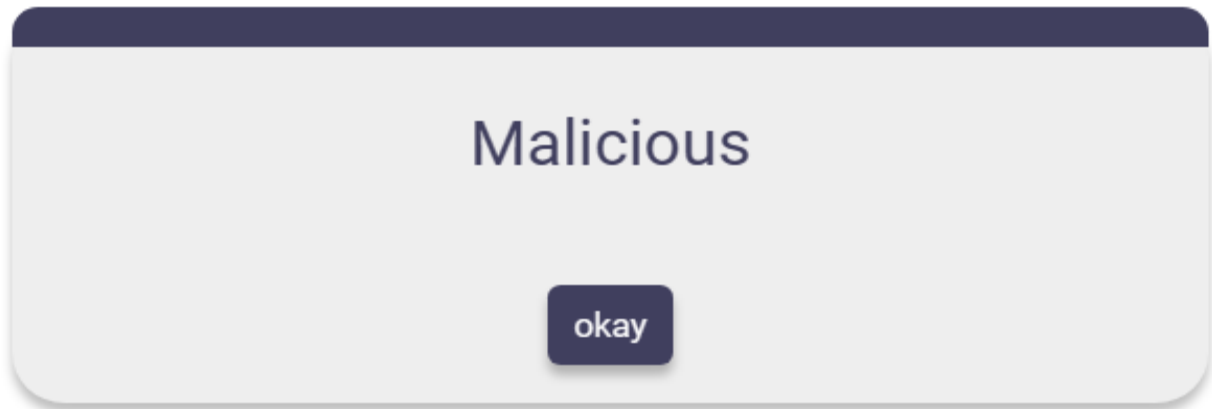


Figure 28 : Upload with login and the pdf is benign.

5.4.10.4. Case 4: Upload with login but out of trials:

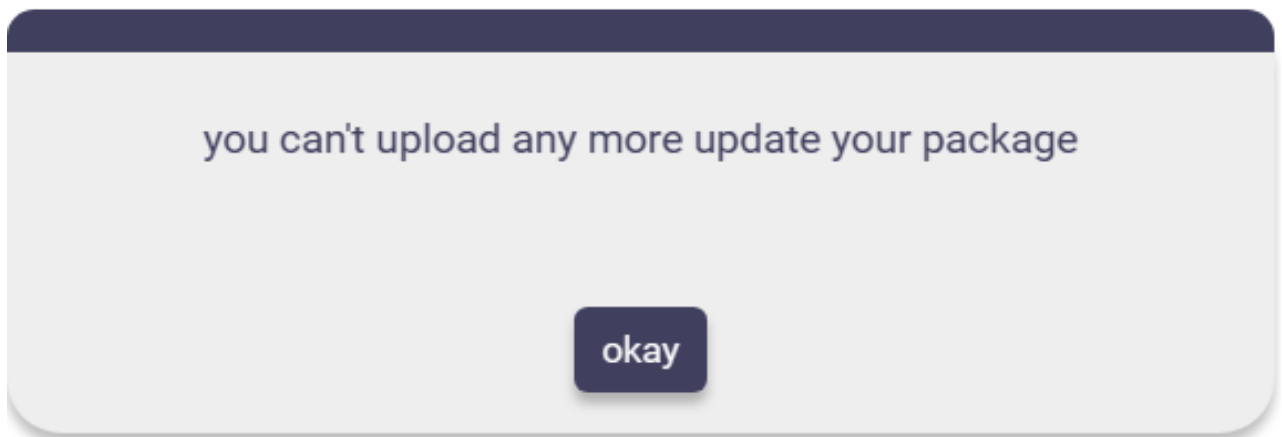


Figure 29: Upload with login but out of trials.

5.4.11. Testing

5.4.11.1. Testing User Sign-Up

Table 7: Testing User Sign-Up

Test case id	Test Objective	Pre-condition	Steps	Test data	Expected Result	Actual Result
TC_01	Successful sign up	Valid username, First-name, Last-name, Email, Password, And Re-password	1-enter username 2-enter First-name 3-enter last-name 4-enter email 5-enter password 6-re-enter password	Username=abdullah2 First-name=Abdullah Last-name=abdelmouty Email = abdo@gmail.com Password=abcd1234 Re-password=abcd1234	Sign-up Successfully can go to the home page	Sign-up Successfully can go to the home page
TC_02	A message that will appear to tell you to enter a valid email	Valid username, First-name, Last-name, Password, And Re-password Invalid Email	1-enter username 2-enter First-name 3-enter last-name 4-enter email 5-enter password 6-re-enter password	Username=abdullah2 First-name=Abdullah Last-name=abdelmouty Email = abdogmail.com Password=abcd1234 Re-password=abcd1234	The message will alert u to enter a valid email	The message appears
TC_03	A message that will appear to tell you to enter the same password in the password and Re-password fields	Valid username, First-name, Last-name, Email, Password, And Invalid Re-password	1-enter username 2-enter First-name 3-enter last-name 4-enter email 5-enter password 6-re-enter password	Username=abdullah2 First-name=Abdullah Last-name=abdelmouty Email = abdo@gmail.com Password=abcd1234 Re-password=ab1234	Message will alert u to enter the two passwords should be the same	The message appears
TC_04	Make sure that a message will appear to tell you that the field is not empty	Do not enter any field	Do not enter any field		Message will appear to tell the user that the field can't be empty	The message appears

5.4.11.2. Testing user-login

Table 8: Testing user-login

TC_05	Login successfully	Valid email and password.	1. enter the email 2. enter the password	Email =abdo@gmail.com Password=abcd1234	Login successfully and can upload pdf	Login successfully and can upload pdf
TC_06	Make sure that a message appears to tell the user that the email or password is wrong	Invalid email or password	1. enter valid email 2. enter invalid password	Email =abdogmail.com Password=abcd1234	Message will appear to tell the user that the email or password is wrong	Message will appear

5.4.11.3. Testing uploading file

Table 9: Testing uploading file

TC_07	Upload without login	Pdf to check	Upload the PDF the user wants to check	pdf	Message to alert users that they can't upload without login	Message will appear
TC_08	Upload with login	Pdf to check	Upload the PDF the user wants to check	pdf	Message to give the user the result	Message will appear
TC_09	Upload with login but the free trials ended	Pdf to check	Upload the PDF the user wants to check	pdf	Message to alert users that they can't upload without buying extra credits	Message will appear

5.4.11.4. Testing payment

Table 10 : Testing uploading file

TC_10	pay successfully	Valid username, Full name, Email, Phone, Country, Period, Package, Card-id,	1-Enter username 2-Enter full-name 3-Enter Email 4-Enter phone 5-Enter country 6-Enter period 7-Enter package 8-Enter card-id	fullName=abdullah abdelmouty, email= abdo@gmail.com, phoneNumber= 123456789, country=egypt, period=monthly, package_id=gold, card_id=12345678910, user_id=abdullah2	Payment is done successfully and can be uploaded again	Payment is done successfully and can be uploaded again
TC_11	A message that will appear to tell you to enter a valid email	Valid username, Full name, Phone, Country, Period, Package, Card-id Invalid Email	1-Enter username 2-Enter full-name 3-Enter Email 4-Enter phone 5-Enter country 6-Enter period 7-Enter package 8-Enter card-id	fullName=abdullah abdelmouty, email= abdogmail.com, phoneNumber= 123456789, country=egypt, period=monthly, package_id=gold, card_id=12345678910, user_id=abdullah2	Message to alert the user to enter a valid email	Message will appear
TC_12	A message that will appear to tell you to enter a valid email	Valid username, Full name, Phone, Country, Period, Package, Email Invalid Card-id	1-Enter username 2-Enter full-name 3-Enter Email 4-Enter phone 5-Enter country 6-Enter period 7-Enter package 8-Enter card-id	fullName=abdullah abdelmouty, email= abdo@gmail.com, phoneNumber= 123456789, country=egypt, period=monthly, package_id=gold, card_id=123, user_id=abdullah2	Message to alert user to enter valid card ID	Message will appear Message will appear
TC_13	Make sure that a message will appear to tell you that the field is not empty	Do not enter any field	Do not enter any field		Message will appear to tell the user that the field can't be empty	The message appears

5.5.Communication with hardware prototype

5.5.1 Receiving PDF from MCU

The process begins with uploading the PDF file to Google Drive. Then The MCU sends the link to the backend through an HTTP POST request. The system receives this link and uses a Python script to download the PDF file from the provided link.

5.5.2. Preprocessing the PDF File

Once the PDF file is downloaded, the backend preprocesses it by converting the PDF to an image format. This image is then sent to the Machine Learning (ML) model for analysis. The ML model processes the image and returns a response indicating whether the PDF is benign or malicious.

5.5.3. Sending the Response to the MCU

After the ML model returns its analysis, the system sends the response back to the MCU. This is done through an HTTP request where a response value of 0 indicates that the PDF is benign, and a response value of 1 indicates that the PDF is malicious.

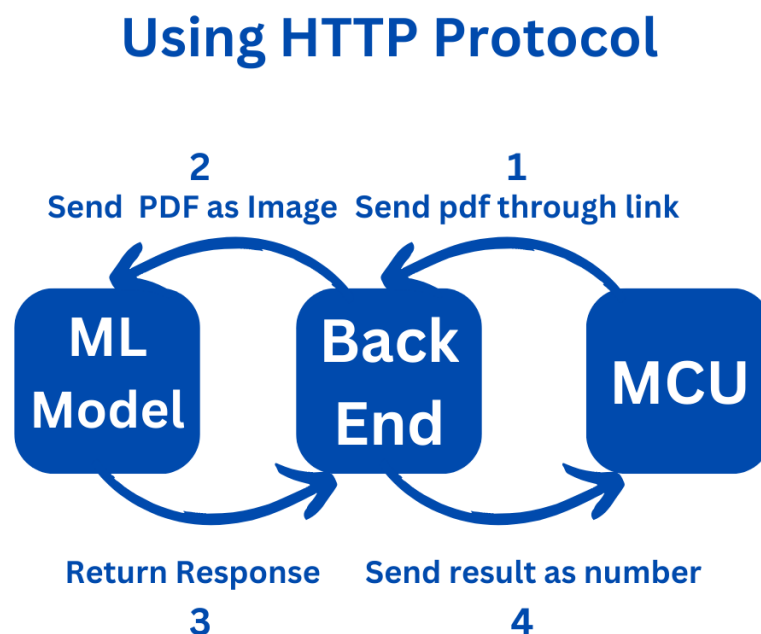


Figure 30: HTTP protocol



CHAPTER (6)

EMBEDDED SYSTEM

Chapter 6 : Embedded System

6.1. Introduction:

Embedded systems have become an integral part of modern technology, shaping the way we interact with various devices in our daily lives. From household appliances to industrial machinery, embedded systems are at the core of countless applications, enabling intelligent and efficient operations. This chapter provides an introduction to embedded systems, with a particular focus on their role in our project, which leverages the ESP8266 microcontroller.

An embedded system is a combination of hardware and software designed to perform specific tasks within a larger system. Unlike general-purpose computers, embedded systems are dedicated to particular functions and are often optimized for performance, power consumption, and cost. These systems can be found in a wide range of applications, including automotive systems, medical devices, consumer electronics, and more.

The key components of an embedded system include a microcontroller or microprocessor, memory, input/output interfaces, and software. The central processing unit (CPU), such as the ESP8266 microcontroller in our project, executes the software instructions. Memory includes both volatile (RAM) and non-volatile (ROM or flash) memory used to store the software code and data. Input/output interfaces allow the embedded system to interact with external devices and sensors, such as keyboards, displays, and network interfaces. Software, in the form of firmware or application software, controls the hardware and performs the desired functions.

Our project is divided into three interrelated parts: Artificial Intelligence (AI), web development, and embedded systems. The main objective is to detect evasive malware within PDF files using deep learning techniques. In the AI component, we develop a deep learning model capable of analyzing PDF files and detecting the presence of malware. This model is trained on a vast dataset of malicious and benign PDFs, learning to differentiate between them with high accuracy. The web development component involves creating a user-friendly website where users can upload PDF files for malware analysis. Once a file is uploaded, the website communicates with the AI model to determine whether the PDF is safe or malicious. This provides users with an easy and accessible way to check their documents.

The embedded systems component is where the ESP8266 microcontroller comes into play. This part of the project ensures secure and reliable

communication between different system elements. The transmitter microcontroller, based on the ESP8266, is responsible for sending the link of PDF file to the website for analysis. The website analyzes the PDF and sends a response back to the transmitter, indicating whether the file is safe or contains malware. If the PDF is deemed safe, the transmitter sends it to the receiver microcontroller. If the file is malicious, it is deleted, ensuring a secure communication link without malware.

Embedded systems play a crucial role in our project by facilitating the secure transmission and reception of PDF files. The ESP8266 microcontroller, with its robust Wi-Fi capabilities, ensures that the communication between the transmitter and the website is seamless and reliable. Additionally, the use of embedded systems allows for real-time processing and decision-making, enhancing the overall efficiency and security of the project.

6.2. Problem Statement:

The increasing prevalence of cyber threats has made the detection and prevention of malware a critical concern in today's digital landscape. Among the various types of malware, evasive malware poses a significant challenge due to its ability to bypass traditional security measures and remain undetected. This project addresses the issue by focusing on the detection of evasive malware within PDF files, which are commonly used for document exchange but can also serve as vectors for malicious code.

Our solution leverages a combination of artificial intelligence, web development, and embedded systems to create a comprehensive and robust defense mechanism. The core functionality of the project is to utilize deep learning techniques for the identification of malware in PDF files. Users can upload their PDF documents to a specially designed website, which then processes these files using a trained deep-learning model. This model has been developed to distinguish between safe and malicious PDFs with high accuracy, providing users with a reliable tool for malware detection.

The AI component is crucial as it enhances the system's ability to detect sophisticated and previously unknown malware variants that traditional signature-based methods might miss.

In addition to the AI-based detection system, the project integrates embedded systems to ensure secure and efficient communication of PDF files. The use of ESP8266 microcontrollers facilitates the transmission and reception of files between different components of the system. Specifically, the project employs a

transmitter microcontroller to send PDF files to the analysis website and a receiver microcontroller to handle the safe files post-analysis. The embedded systems are designed to provide real-time feedback and actions based on the analysis results. If the PDF is found to be safe, it is forwarded to the receiver for further use; if it is deemed malicious, it is deleted to prevent any potential harm.

This dual-layer approach—combining advanced AI detection with secure embedded system communication—addresses the problem of malware transmission and ensures a safe exchange of documents. The use of ESP8266 microcontrollers not only supports the wireless transmission of data but also enhances the flexibility and scalability of the system.

This design choice allows for the integration of additional features and functionalities in the future, such as remote monitoring and automated updates to the malware detection model.

Our project thus aims to create a secure communication link that eliminates the risk of malware transmission through PDF files. By implementing a deep learning model for malware detection and utilizing embedded systems for secure file handling, we provide a comprehensive solution to a pressing cybersecurity issue. The combination of these technologies ensures that users can trust the safety of their documents while maintaining the efficiency and usability of the system.

Through this project, we demonstrate the potential of integrating AI and embedded systems to address complex cybersecurity challenges, paving the way for more advanced and reliable protection mechanisms in the future.

6.3. Proposed Solution:

To address the problem of detecting and preventing the transmission of evasive malware within PDF files, our proposed solution leverages a combination of artificial intelligence, web development, and embedded systems. This multifaceted approach ensures a robust and efficient mechanism for malware detection and secure document handling. The primary components of the solution include a deep learning-based malware detection model, a user-friendly web interface, and a secure communication system facilitated by ESP8266 microcontrollers.

The first component of our solution is the development of a deep learning model trained to identify malicious content within PDF files. This model is built using a comprehensive dataset of both benign and malicious PDFs, allowing it

to learn and distinguish between safe and harmful files effectively. The model is designed to detect a wide range of malware types, including those that employ evasion techniques to bypass traditional security measures. By utilizing deep learning, our solution can adapt to new and emerging threats, offering a dynamic and continuously improving detection system.

Complementing the AI component is the development of a web-based interface where users can easily upload their PDF files for analysis. This website serves as the primary access point for users to interact with the system. Upon uploading a file, the website communicates with the deep learning model to perform a thorough analysis. The user is then provided with a clear and concise result, indicating whether the file is safe or contains malware. This interface is designed to be intuitive and user-friendly, ensuring that users of all technical backgrounds can utilize the service without difficulty.

The final component of our solution involves the use of embedded systems to manage the secure transmission and reception of PDF files. Two ESP8266 microcontrollers are employed: one as a transmitter and the other as a receiver. The transmitter microcontroller is responsible for sending the uploaded PDF files to the website for analysis. Once the website completes the analysis, it sends a response back to the transmitter, indicating the safety status of the file. If the file is safe, the transmitter forwards it to the receiver microcontroller for further use. If the file is malicious, it is deleted, preventing any potential harm. This process ensures that only safe files are transmitted and used, thereby maintaining a secure communication link.

The integration of these three components forms a comprehensive solution that effectively addresses the problem of malware detection and secure document handling. The deep learning model provides a high level of accuracy and adaptability in identifying malicious PDFs, while the web interface offers an accessible platform for users to engage with the system. The embedded systems ensure secure and reliable communication, safeguarding against the transmission of harmful files.

Additionally, the use of ESP8266 microcontrollers enhances the flexibility and scalability of the system. These microcontrollers support wireless communication, enabling remote operation and easy integration into various environments. The modular design of our solution allows for future expansions and improvements, such as incorporating additional types of files for analysis or enhancing the user interface with more features.

In conclusion, our proposed solution combines advanced AI techniques, user-centric web development, and secure embedded systems to provide a robust mechanism for detecting and preventing malware within PDF files. By addressing the problem from multiple angles, we ensure a high level of security and reliability, offering users a trustworthy tool for safeguarding their documents. This innovative approach not only solves the immediate problem but also sets the foundation for future advancements in cybersecurity and secure communication technologies.

6.4.Our System:

It's a small prototype for our application to implement a safe communication link without any malware, we have Node MCU esp8266-12e as the transmitter and another receiver, first the transmitter will communicate with our web server and send the data that is needed to check its safety and the web application will send its response to the transmitter depending on our deep learning API and then depending on the response the transmitter decided to send the data to the receiver or not.

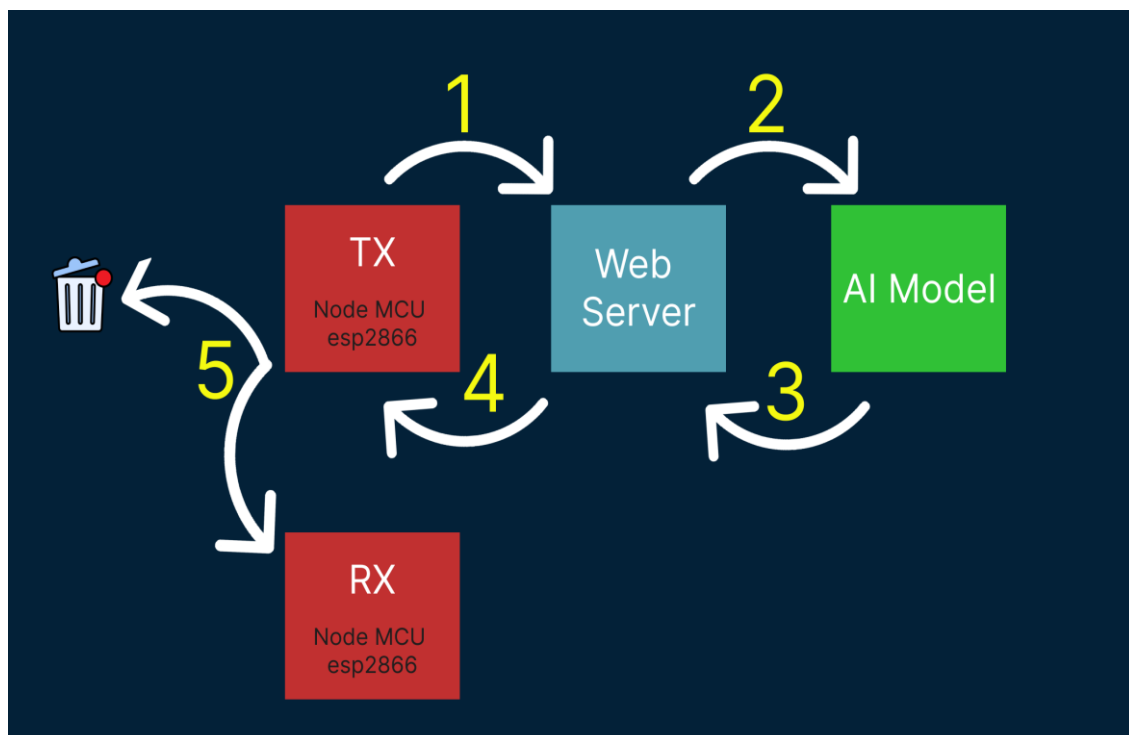


Figure 31 : Embedded system block diagram

6.4.1 Hardware Used:

-Node MCU ESP 8266 12E

NodeMCU is an open-source development board and firmware based on the widely used ESP8266 -12E Wi-Fi module. It allows you to program the ESP8266 Wi-Fi module with the simple and powerful LUA programming language or Arduino IDE.

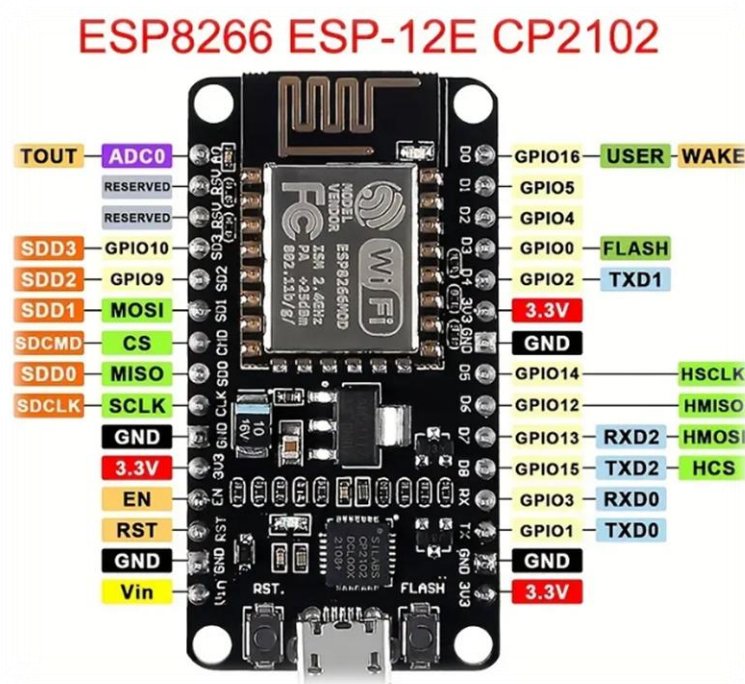


Figure 32 : Node MCU esp8266 pin mapping

Features:

- Finally, a programmable Wi-Fi module.
- Arduino-like (software-defined) hardware IO.
- Can be programmed with the simple and powerful Lua programming language or Arduino IDE.
- USB-TTL included, plug & play.
- 10 GPIOs D0-D10, PWM functionality, IIC, and SPI communication, 1-Wire and ADC A0, etc. all in one board.
- Wi-Fi networking (can be used as an access point and station, host a Web server), connect to the internet to fetch or upload data.
- Event-driven API for network applications.
- PCB antenna.

6.4.2. Implementation:

6.4.2.1. Communicate with web server:

The communication between the ESP8266 microcontroller and the Python backend web application is established through HTTP requests, enabling secure and efficient data transmission. The ESP8266 is configured to connect to a Wi-Fi network and send HTTP POST requests to a web server. This server is designed to handle file uploads, specifically PDF files, for analysis. The microcontroller sends the PDF link to the server. The application, running on the server, processes these requests by receiving the files and performing necessary operations, such as malware analysis. This setup ensures a seamless interaction where the ESP8266 acts as a transmitter, sending data to the server, which in turn analyzes the content and responds accordingly. This method leverages the robustness of the ESP8266's Wi-Fi capabilities to create a reliable and scalable communication system for secure file handling.



Figure 33 : HTTP Request

HTTP: (Hypertext Transfer Protocol) is the foundation of data communication on the World Wide Web. It defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands. HTTP requests are made by a client (usually a web browser) to a server, which then processes the request and returns a response. Two of the most common types of HTTP requests are GET and POST.

HTTP Request

An HTTP request is a message sent by the client to the server to initiate an action. It consists of the following main components:

- **Request Line:** Includes the method (e.g., GET or POST), the resource URL, and the HTTP version.
- **Headers:** Provide additional information about the request, such as content type, user agent, and cookies.
- **Body:** Contains data being sent to the server (used mainly in POST requests).

GET Request

A GET request is used to retrieve data from a server. When a client sends a GET request, it asks the server to return the requested resource. GET requests are typically used to fetch web pages, images, and other data that do not involve changing server state.

Characteristics of GET Requests:

- **Data is sent in the URL:** Parameters are appended to the URL in a query string, e.g., `http://example.com/page?param1=value1¶m2=value2`.
- **Idempotent:** Multiple identical GET requests will have the same effect as a single request.
- **Safe:** Should not alter server state.
- **Cached:** Responses can be cached by the browser or intermediary proxies to reduce server load

POST Request:

A POST request is used to send data to the server to create or update a resource. The data is included in the body of the request, not in the URL. POST requests are commonly used for form submissions, file uploads, and when performing actions that change server state.

Characteristics of POST Requests:

- **Data is sent in the body:** Allows for large amounts of data to be sent.
- **Not idempotent:** Multiple identical POST requests may have different effects.
- **Can alter server state:** Often used for actions that change data on the server.

- **Not cached:** Responses are typically not cached due to their potential to change state.

6.4.2.2. Decision making:

In our system, the decision-making process for handling PDF files is crucial for ensuring secure communication. Once a PDF file is transmitted by the ESP8266 microcontroller (acting as the transmitter) to the web server for analysis, the server processes the file using advanced deep-learning techniques to detect any presence of malware.

Upon completion of the analysis, the server sends back a response to the transmitter indicating the result of the malware check. This response is either a 1 or a 0.

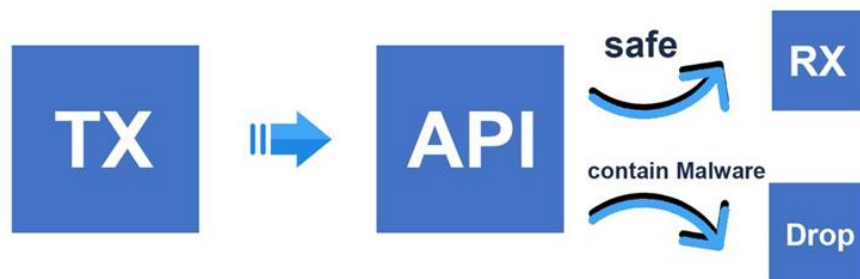


Figure34 : Decision making

A response of 1 signifies that the PDF file is safe and free from malware. Upon receiving this positive response, the transmitter proceeds to send the link to the receiver ESP8266 microcontroller, allowing the file to be safely transmitted and used. Conversely, a response of 0 indicates that the PDF file is malicious. In this case, the transmitter does not forward the link to the receiver; instead, it deletes the file to prevent any potential harm.

This decision-making mechanism ensures that only safe files are communicated and transmitted within the system, thereby maintaining a secure and reliable communication link free from malware. This approach combines robust malware detection with real-time decision-making, leveraging the capabilities of both the web server and the ESP8266 microcontrollers to achieve a secure data exchange environment.

6.4.2.3. Sending from TX to RX:

6.4.2.3.1 Communication Protocol:

-SPI (Serial Peripheral Interface)

SPI (Serial Peripheral Interface) is a synchronous serial communication protocol widely used in embedded systems, microcontrollers, and other digital integrated circuits. It facilitates high-speed, full-duplex data transmission between a master device (such as a microcontroller or CPU) and one or more slave devices (typically sensors, displays, or memory chips). The protocol operates using four main signals:

- **SCLK (Serial Clock):** Controls the timing of data transmission.
- **MOSI (Master Out Slave In):** Carries data from the master to the slaves.
- **MISO (Master In Slave Out):** Transfers data from slaves to the master.
- **SS/CS (Slave Select/Chip Select):** Determines which slave device is actively communicating with the master.

SPI offers advantages such as simplicity, high data rates (up to several MHz or even tens of MHz), and efficient communication with multiple peripherals through its use of separate slave select lines. Its straightforward architecture makes it ideal for applications requiring rapid data exchange and real-time operation, such as in sensor networks, data acquisition systems, and peripheral interfaces in microcontroller-based designs.

Wiring :

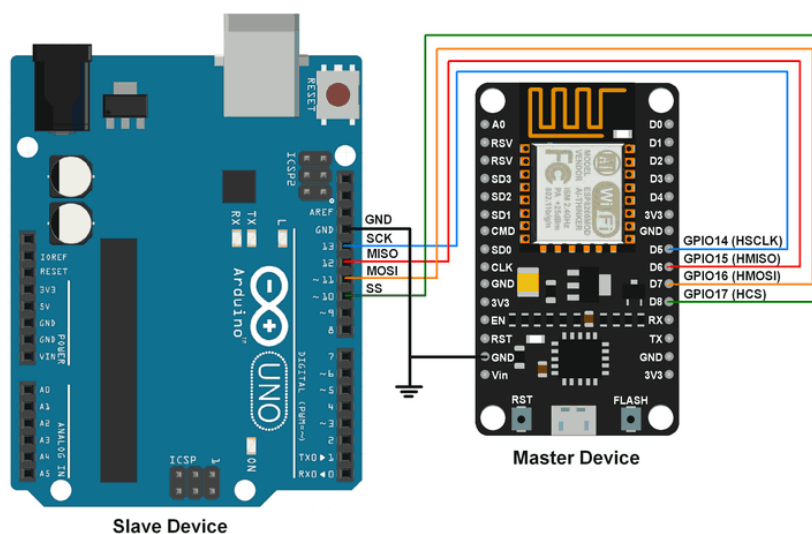


Figure 35 : SPI wiring

Upon receiving a positive response of 1 from the web server, indicating that the PDF file has been verified as safe, the ESP8266 microcontroller functioning as the transmitter initiates the next phase of secure data transmission. Utilizing the Serial Peripheral Interface (SPI), the transmitter communicates the link or identifier of the verified PDF file to the ESP8266 microcontroller acting as the receiver. SPI is chosen for its high-speed, full-duplex communication capability, ideal for real-time data exchange between microcontrollers. This communication link ensures that the receiver receives the necessary information promptly and accurately. Once the link is successfully transmitted, the receiver is ready to securely retrieve and utilize the verified PDF file. This streamlined process not only guarantees the integrity of transmitted data but also underscores the system's efficiency in handling secure document exchange. By integrating SPI communication with robust malware detection capabilities, our solution achieves a reliable and responsive framework for safe data transmission, safeguarding against potential cybersecurity threats effectively.



CHAPTER (7)

USER INTERFACE

Chapter 7 : User Interface

The user interface (UI) of our proposed system plays a crucial role in ensuring an intuitive and seamless experience for users as they interact with the malware detection platform. The UI is designed to be user-friendly, providing clear navigation, real-time feedback, and a secure environment for uploading PDF files. Below are the key features and design elements of our user interface:

7.1. Key Features:

7.1.1.File Upload Mechanism:

- **Drag-and-Drop Interface:** Users can easily drag and drop PDF files into the designated area on the webpage for uploading.
- **File Selection Button:** An alternative option allows users to browse their device and select the PDF files they wish to upload.

7.1.2.Real-Time Feedback:

- **Progress Indicators:** A progress bar will display the upload status of the PDF file, giving users a visual indication of the upload process.
- **Analysis Status:** Once the file is uploaded, the UI will show real-time analysis status, informing users that the file is being scanned for malware.

7.2.Result Display:

- **Clear Notifications:** After the analysis is complete, the UI will display a clear notification indicating whether the PDF is safe or contains malware.

7.3.User Authentication:

- **Login/Signup:** Users will have the option to create an account or log in to access additional features and track their uploaded files.
- **Secure Session Management:** To ensure security, sessions will be managed securely with appropriate timeout and logout mechanisms.

7.4.Responsive Design:

- **Cross-Device Compatibility:** The UI will be designed to be fully responsive, ensuring compatibility with various devices, including desktops, tablets, and smartphones.

7.5.Design Elements:

7.5.1.Clean and Modern Layout:

- The UI will feature a clean and modern design, utilizing a minimalist approach to avoid clutter and enhance usability.
- Consistent use of colors, fonts, and icons will ensure a cohesive and visually appealing interface.

7.5.2.Intuitive Navigation:

- Clear navigation menus and breadcrumbs will guide users through different sections of the platform, making it easy to access various features and functionalities.
- Tooltips and help icons will be available to assist users with any unfamiliar features.

7.5.3.Accessibility:

- The UI will be designed with accessibility in mind, ensuring that users with disabilities can easily interact with the platform. This includes support for screen readers, keyboard navigation, and appropriate contrast ratios for text and background colors.

7.5.4.Security Indicators:

- Security is paramount in our system. The UI will include visual indicators, such as padlock icons, to assure users that their interactions and data uploads are secure.

7.6.User Workflow:

- **Access the Platform:** Users navigate to the website and, if necessary, log in or create an account.
- **Upload a PDF File:** Users drag and drop or select a PDF file for analysis.
- **Monitor Upload and Analysis:** Users can see the upload progress and receive real-time status updates during the malware scanning process.
- **Receive Results:** The UI displays whether the file is safe or contains malware. Users can view a detailed report if desired.

7.7.Snapshots:

7.7.1. Logo:

Our logo achieves:

Brand Identity: It serves as a visual representation of your brand. It helps customers recognize and remember your brand among competitors.

Professionalism: A well-designed logo conveys professionalism and builds trust with your audience, showing that you take your business seriously

Memorability: A good logo is memorable and helps customers recall your brand when making purchasing decisions.

Differentiation: It sets you apart from competitors in your industry, helping customers distinguish your products or services.

Marketing Tool: It's used across various marketing materials and platforms, reinforcing your brand message consistently.

Emotional Connection: Logos can evoke emotions and create a connection with your audience, influencing their perception of your brand.

Versatility: A versatile logo can be scaled to different sizes and used across different mediums, from business cards to digital platforms.

In essence, a logo is a cornerstone of your brand identity, playing a crucial role in shaping how customers perceive and interact with your business.



Figure 36 : Logo

7.7.2. Header :



Figure 37 : website header

The header of a website is crucial for several reasons:

- **First Impressions:** The header is often the first thing visitors see when they land on your website. A well-designed header can create a positive first impression, encouraging visitors to stay and explore further.
- **Navigation:** It typically contains the main navigation menu, helping users find their way around the site. A clear and intuitive navigation structure in the header makes it easier for users to locate the information they need.
- **Branding:** The header often includes the company logo, tagline, and sometimes a brand slogan. This helps reinforce brand identity and ensures that visitors know they are on the correct website.
- **Key Information:** Important information, such as contact details, call-to-action buttons, and links to key pages (e.g., Home, About Us, Services, Contact), is often placed in the header for easy access.
- **Consistency:** A consistent header across all pages of the website provides a cohesive look and feel, which is important for maintaining brand integrity and user experience.
- **Trust and Credibility:** Including trust signals such as certifications, awards, or security badges in the header can enhance the credibility of your website.
- **Accessibility:** A well-designed header improves the accessibility of your website by making it easier for users with disabilities to navigate and understand the site structure.

the header of a website plays a vital role in usability, branding, and user experience, making it a key component in effective web design.

7.7.3. Section one of the home page



Figure 38 : Section one of the home page

7.7.4. Section two of the home page

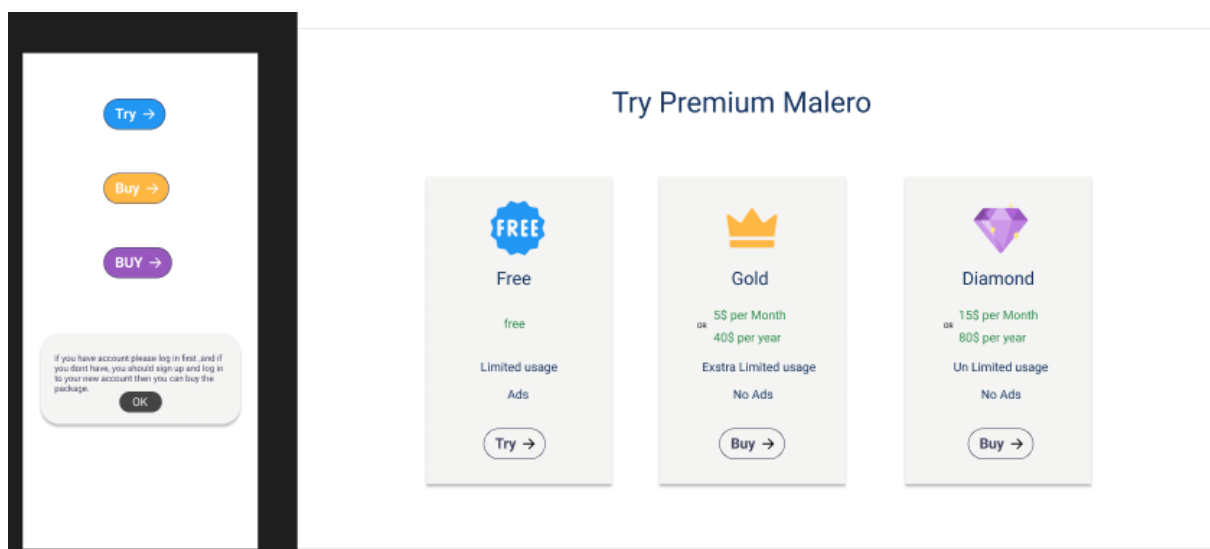


Figure 39 : Premium Malero

7.7.5. Section three of the home page



Figure 40: Why Malero

7.7.6. Section four of the home page

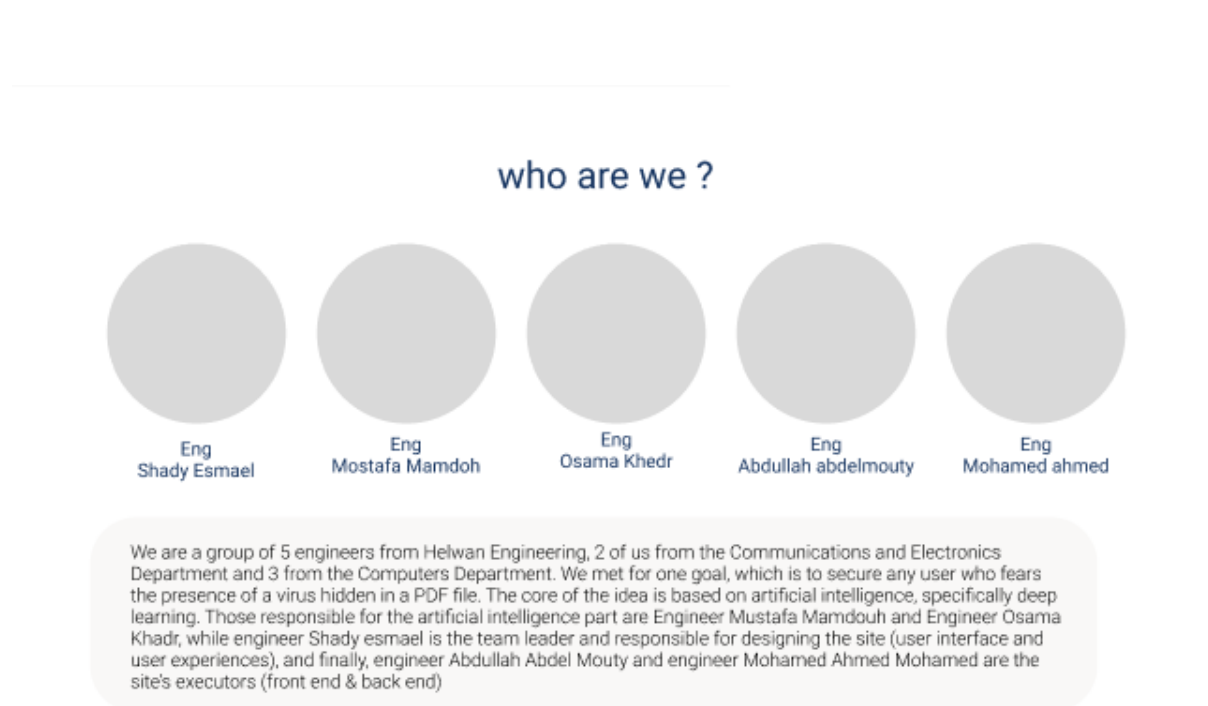


Figure 41: who are we

7.7.7. Section five of the home page

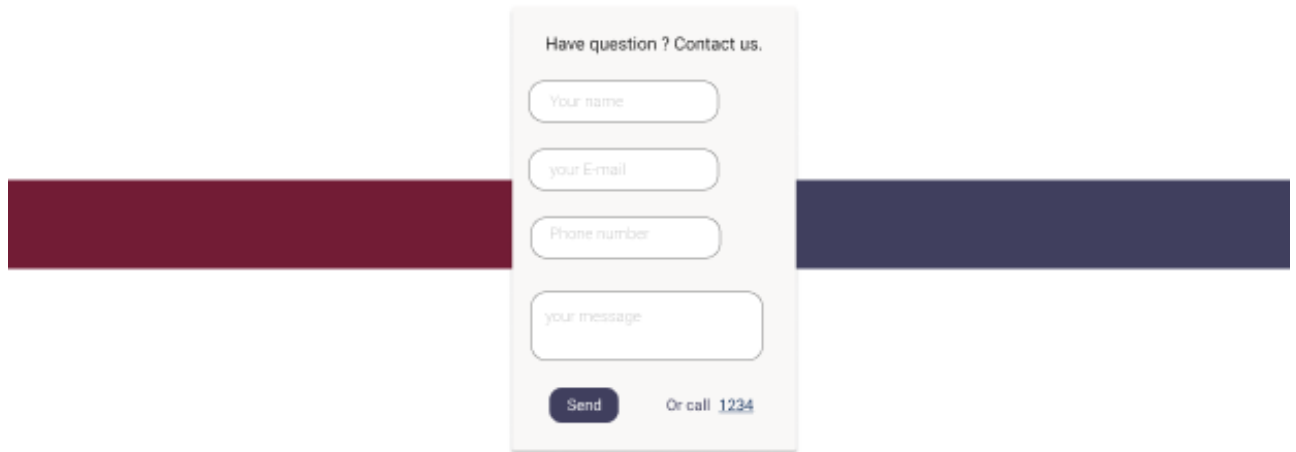
A contact form overlay is centered on the page. It has a light gray background and rounded corners. At the top, it says "Have question ? Contact us." Below this are four input fields: "Your name", "your Email", "Phone number", and "your message". At the bottom, there is a "Send" button and a link that says "Or call 1234". The form is positioned over a dark red horizontal bar on the left and a dark blue horizontal bar on the right.

Figure 42: Contact us

7.7.8. Footer



Figure 43: Footer

7.7.9. Home dark mode

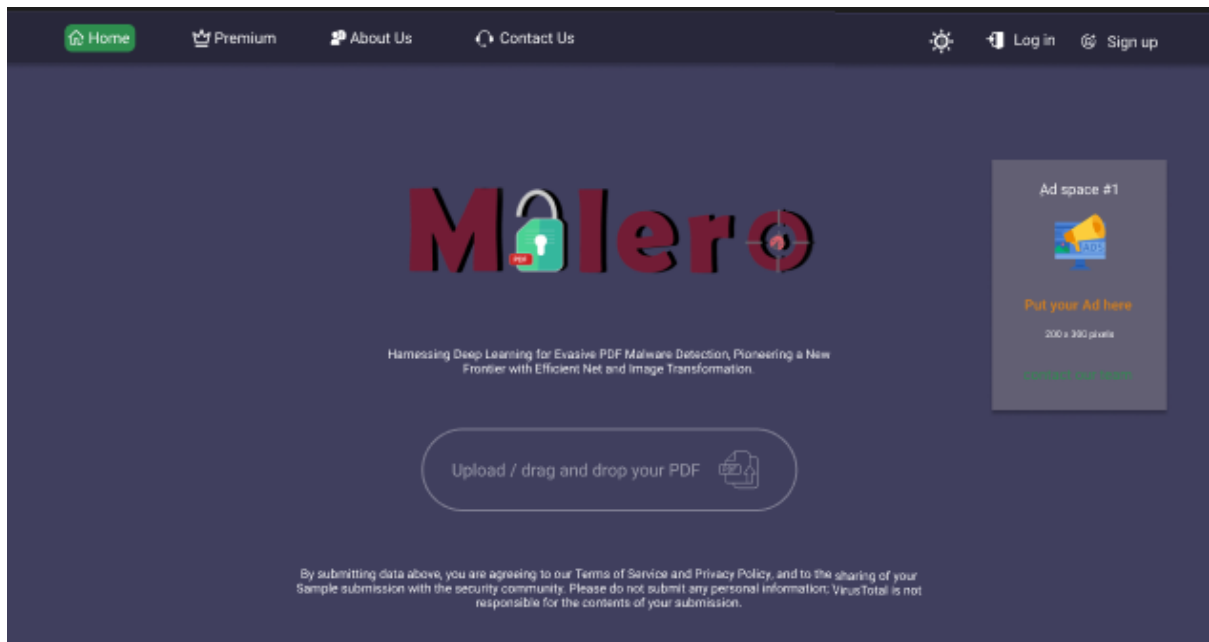


Figure 44 : Home dark mode

7.7.10. Registration page

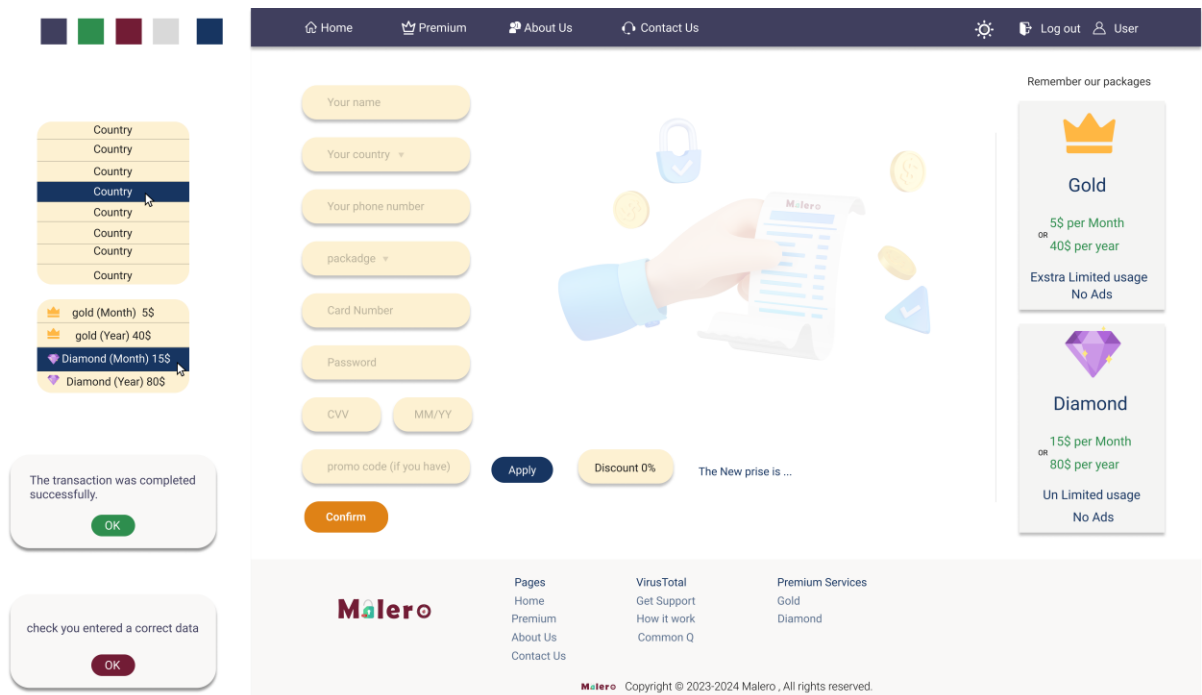


Figure 45: Registration page

7.7.11. Registration page dark mode

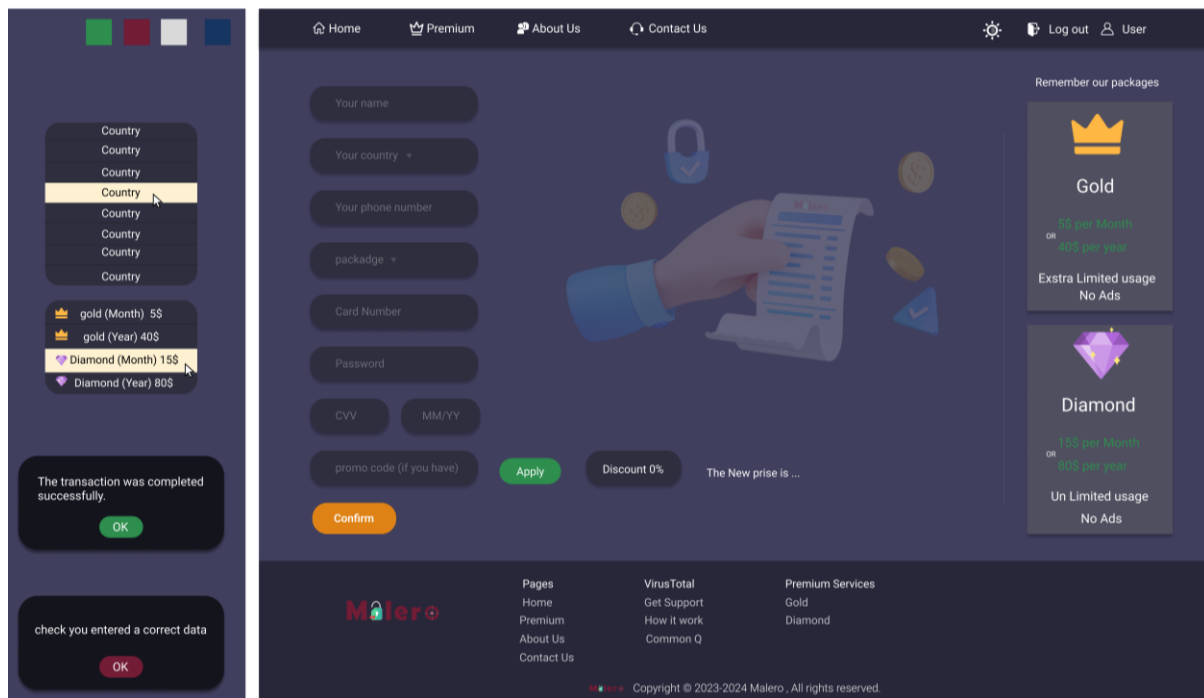


Figure 46: Registration page dark mode

7.7.12. Purchasing page

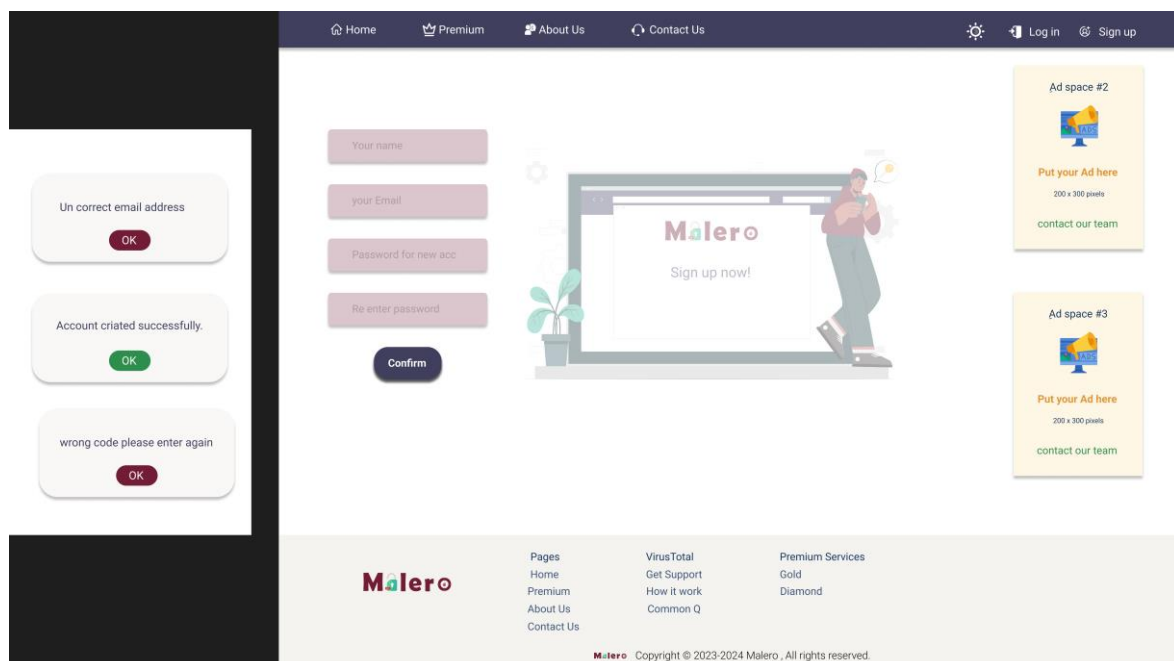


Figure 47: Purchasing page

7.7.13. Purchasing page dark mode

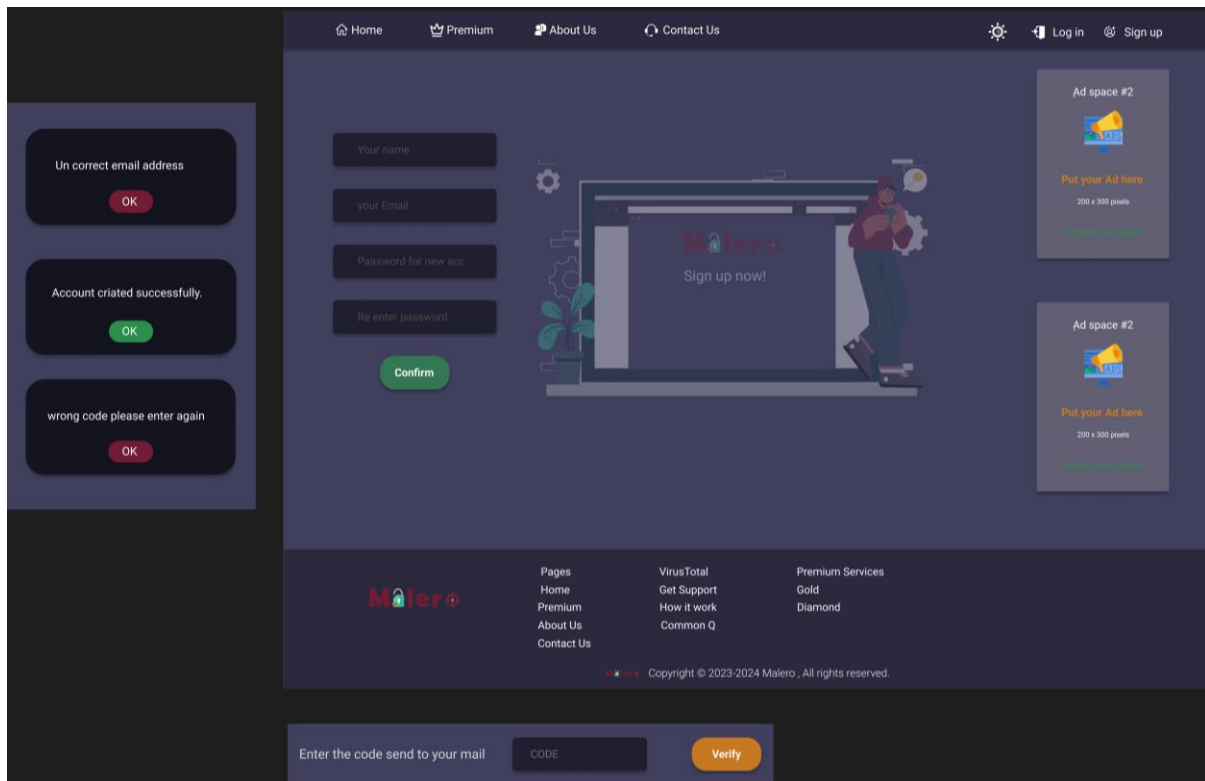


Figure 48: Purchasing page dark mode

7.8.Tooles and software used:

7.8.1. Software:

Figma



Figure 49 : Figma

Photoshop



Figure 50: Photoshop

Illustrator



Figure 51: Illustrator

7.8.2. Free images and icons source :

Flaticon



Figure 52 : Flaticon

Freepic



Figure 53 : Freepic



CHAPTER (8)

CONCLUSION AND FUTURE WORK

Chapter 8: Conclusion and Future Work

8.1. Conclusion:

Our project aims to address the significant challenge of detecting evasive malware in PDF files through an innovative and integrated approach. By leveraging the strengths of artificial intelligence, web development, and embedded systems, we are creating a robust solution that not only enhances malware detection capabilities but also ensures secure and reliable communication of data.

Our deep learning model, trained on a diverse dataset of benign and malicious PDFs, serves as the core of our system. It employs advanced techniques to identify complex patterns associated with malware, thus providing a high level of accuracy in detection. This model is supported by a user-friendly web interface that allows users to easily upload their PDF files for analysis, offering real-time feedback on the safety of their documents. The web platform's design focuses on accessibility, responsiveness, and security, ensuring a seamless and trustworthy user experience.

The integration of embedded systems further strengthens our solution by enabling secure file transmission between microcontrollers and the web platform. This component ensures that only safe files are forwarded, while malicious ones are deleted, thereby maintaining a secure communication link. The microcontroller setup, with a transmitter and receiver, facilitates this process efficiently, demonstrating the practical application of our system in real-world scenarios.

Our project also builds upon existing solutions, addressing their limitations and enhancing their capabilities. Traditional antivirus software, heuristic analysis, sandboxing, and other methods provide foundational protection but often fall short against sophisticated evasive malware. Our approach, utilizing deep learning, offers a more adaptive and comprehensive defense mechanism. Additionally, by combining these techniques with secure web development practices and embedded systems, we provide a holistic solution that addresses multiple aspects of cybersecurity.

The benefits of our system are multifaceted. It not only improves the detection and prevention of malware but also ensures that users can trust the integrity and safety of their communications and data transfers. This is particularly crucial in an era where cyber threats are becoming increasingly sophisticated and pervasive. Our project's emphasis on real-time analysis, user-friendly interaction, and secure file handling sets it apart as a comprehensive and effective tool in the fight against malware.

Moreover, our system is designed with scalability and future-proofing in mind. As new malware emerges and evolves, our deep learning model can be continuously updated and retrained to adapt to these threats, ensuring sustained efficacy. The modularity of our system components also allows for easy integration of additional features and improvements, catering to the dynamic nature of cybersecurity needs.

In essence, our project represents a significant advancement in malware detection and cybersecurity. By integrating deep learning, web development, and embedded systems, we provide a robust, reliable, and user-friendly solution to the growing problem of evasive malware. Our system not only enhances current malware detection methodologies but also offers a secure and efficient means of communication, protecting users from potential threats. This comprehensive approach ensures that our solution is not only effective today but will continue to be relevant and resilient in the face of future cybersecurity challenges.

8.2. Future Work:

To enhance and expand the capabilities of our malware detection system, several future work initiatives are planned. These initiatives aim to improve system performance, accessibility, and robustness. Below is an extended overview of the future work we intend to undertake:

8.2.1. Deploy the Website on a Cloud Service:

A crucial step in making our malware detection system widely accessible is to deploy the website on a cloud service such as Amazon Web Services (AWS). Cloud deployment offers several advantages, including scalability, reliability, and global accessibility. By hosting our application on AWS, we can ensure that it is available to users around the world, providing consistent performance regardless of user location. Additionally, AWS offers robust security features

that will help safeguard user data and maintain the integrity of our system. This step will also facilitate easier maintenance and updates, ensuring that our service remains up-to-date with the latest security measures and technological advancements.

8.2.2. Testing Various Architectures:

To further optimize our deep learning model, we plan to explore and compare different neural network architectures, such as EfficientNet, VGG, and ResNet. Each architecture has its strengths and trade-offs in terms of computational efficiency, accuracy, and complexity. By systematically testing these architectures, we aim to identify the most effective model for detecting evasive malware in PDF files. This process will involve extensive experimentation and performance evaluation to determine which architecture provides the best balance between detection accuracy and computational efficiency.

8.2.3. Feature Interpretation and Extraction:

Understanding which features are most indicative of malware is crucial for building more interpretable models and refining our detection techniques. As suggested by Aboaoja et al. (2022), investigating feature importance can lead to significant improvements in model transparency and performance. We plan to delve deeper into feature interpretation, identifying the key attributes that distinguish malicious PDFs from benign ones. This knowledge will inform the development of more targeted and effective feature extraction methods, enhancing the overall accuracy and reliability of our detection system. Additionally, more interpretable models can provide valuable insights to users and cybersecurity professionals, aiding in the understanding of malware characteristics and attack vectors.

8.2.4. Multi-Modal Approaches:

Exploring multi-modal approaches involves combining convolutional neural networks (CNNs) with other machine learning or deep learning techniques, such as recurrent neural networks (RNNs). This hybrid approach can potentially enhance our system's performance by leveraging the strengths of different

models. For instance, CNNs are excellent at capturing spatial features, while RNNs are adept at handling sequential data. By integrating these models, we can create a more comprehensive and nuanced analysis of PDF files, improving the detection of evasive malware. This approach will also allow us to experiment with different data modalities and representations, potentially uncovering new ways to identify and mitigate threats.

8.2.5. Continuous Learning and Adaptation:

As malware continues to evolve, our detection system must adapt to new and emerging threats. Implementing continuous learning mechanisms will allow our model to update and retrain new data, ensuring it remains effective against the latest malware variants. This involves setting up a pipeline for regularly collecting new samples of benign and malicious PDFs, incorporating them into our training dataset, and periodically retraining the model. By doing so, we can maintain high detection accuracy and stay ahead of adversaries who develop new evasion techniques.

8.2.6. Integration with Existing Cybersecurity Systems:

To maximize the impact of our malware detection system, we plan to explore integration with existing cybersecurity infrastructures. This could involve developing APIs or plugins that allow our system to work seamlessly with popular antivirus software, email filters, and network security tools. Such integration will enhance the overall security posture of organizations by providing an additional layer of defense against PDF-based malware attacks. It will also facilitate easier adoption of our technology, as users can incorporate it into their existing workflows without significant disruption.



REFERENCES

References

For Deep Learning:

1. Aboaoja, F.A., Zainal, A., Ghaleb, F.A., Al-rimy, B.A.S., Eisa, T.A.E. and Elnour, A.A.H. (2022) Malware detection issues, challenges, and future directions: A survey. *Applied Sciences*, 12(17), p.8482.
2. Abu Al-Haija, Q., Odeh, A. and Qattous, H. (2022) PDF malware detection based on optimizable decision trees. *Electronics*, 11(19), p.3142.
3. Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M.J., Levi, M., Moore, T. and Savage, S. (2013) Measuring the cost of cybercrime. In *The economics of information security and privacy*, pp. 265-300.
4. Barros, P.H., Chagas, E.T., Oliveira, L.B., Queiroz, F. and Ramos, H.S. (2022) Malware-SMELL: A zero-shot learning strategy for detecting zero-day vulnerabilities. *Computers & Security*, p.102785.
5. Bozkir, A.S., Cankaya, A.O. and Aydos, M. (2019) Utilization and comparison of convolutional neural networks in malware recognition. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1-4.
6. Cyrus, C. (2021) IoT cyberattacks escalate in 2021, according to Kaspersky. *IoT World Today* [online]. Available from: <https://www.iotworldtoday.com/2021/09/17/iot-cyberattacks-escalate-in-2021-according-to-kaspersky> [Accessed 22 December 2022].
7. Damaševičius, R., Venčkauskas, A., Toldinas, J. and Grigaliūnas, Š. (2021) Ensemble-based classification using neural networks and machine learning models for Windows PE malware detection. *Electronics*, 10(4), p.485.
8. Davuluru, V.S.P., Narayanan, B.N. and Balster, E.J. (2019) Convolutional neural networks as classification tools and feature extractors for distinguishing malware programs. In *IEEE National Aerospace and Electronics Conference (NAECON)*, pp. 273-278.
9. El-Shafai, W., Almomani, I. and AlKhayer, A. (2021) Visualized malware multi-classification framework using fine-tuned CNN-based transfer learning models. *Applied Sciences*, 11(14), p.6446.
10. Fettaya, R. and Mansour, Y. (2020) Detecting malicious PDF using CNN. *arXiv preprint arXiv:2007.12729*.
11. Kumar, S. (2021) MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in the Internet of Things. *Future Generation Computer Systems*, 125, pp.334-351.
12. Kumar, S. and Janet, B. (2022) DTMIC: Deep transfer learning for malware image classification. *Journal of Information Security and Applications*, 64, p.103063.
13. Lo, W.W., Yang, X. and Wang, Y. (2019) An Xception convolutional neural network for malware classification with transfer learning. In *2019*

- 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1-5.
14. Maryam, I., Princy, V., Ali, T. and Arash, H. (2022) PDF malware detection based on stacking learning. In The International Conference on Information Systems Security and Privacy.
 15. Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B.S. (2011) Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on visualization for cyber security, pp. 1-7.
 16. Ngo, Q.D., Nguyen, H.T., Le, V.H. and Nguyen, D.H. (2020) A survey of IoT malware and detection methods based on static features. *ICT Express*.
 17. Narayanan, B.N. and Davuluru, V.S.P. (2020) Ensemble malware classification system using deep neural networks. *Electronics*, 9(5), p.721.
 18. Olowoyo, O. and Owolawi, P. (2020) Malware classification using deep learning technique. In 2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC), pp. 1-6.
 19. Roseline, S.A., Geetha, S., Kadry, S. and Nam, Y. (2020) Intelligent vision-based malware detection and classification using deep random forest paradigm. *IEEE Access*, 8, pp.206303-206324.
 20. Sriram, S., Vinayakumar, R., Sowmya, V., Alazab, M. and Soman, K.P. (2020) Multi-scale learning-based malware variant detection using spatial pyramid pooling network. In IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 740-745.
 21. Tan, M. and Le, Q. (2019) EfficientNet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning, pp. 6105-6114.
 22. Vasan, D., Alazab, M., Wassan, S., Safaei, B. and Zheng, Q. (2020) Image-based malware classification using ensemble of CNN architectures (IMCEC). *Computers & Security*, 92, p.101748.
 23. Wegner, P. (2022) Global IoT market size grew 22% in 2021 - these 16 factors affect the growth trajectory to 2027. *IoT Analytics* [online]. Available from: <https://iot-analytics.com/iot-market-size> [Accessed 22 December 2022].
 24. Yadav, P., Menon, N., Ravi, V., Vishvanathan, S. and Pham, T.D. (2022) EfficientNet convolutional neural networks-based Android malware detection. *Computers & Security*, 115, p.102622.
 25. Yan, P. and Yan, Z. (2018) A survey on dynamic mobile malware detection. *Software Quality Journal*, 26(3), pp.891-919.

For Web development:

Official Documentation

1. React Official Documentation:

- The official React documentation is an excellent starting point for learning React. It provides comprehensive guides and tutorials on core concepts, advanced techniques, and best practices. It also includes detailed API reference and examples to help developers understand how to use React effectively.
- React Official Docs

FreeCodeCamp

2. React Course:

- FreeCodeCamp offers an interactive React course that covers both basic and advanced topics. This course is project-based, which means you'll build real-world projects while learning React, helping you apply your knowledge practically.
- FreeCodeCamp React Course

3. CSS Course:

- FreeCodeCamp provides a comprehensive CSS course that is beginner-friendly. It covers fundamental concepts like selectors, properties, and values, as well as advanced topics such as Flexbox and Grid layout. The course includes hands-on exercises to reinforce learning.
- FreeCodeCamp CSS Course

MDN Web Docs

4. React Documentation on MDN:

- MDN Web Docs provide a detailed guide on getting started with React, including an overview of core concepts, tutorials, and examples. MDN also explains how React integrates with other web technologies and best practices for using React in web development.
- [MDN React Documentation](#)

5. CSS Documentation:

- MDN is a great resource for learning CSS, offering detailed documentation and examples on everything from basic syntax and selectors to advanced layout techniques and animations. It also includes guides on responsive design and accessibility.
- [MDN CSS Documentation](#)

6. JavaScript Guide:

- The MDN documentation is a comprehensive resource for learning JavaScript. It covers basic to advanced concepts, including syntax, control structures, functions, objects, and the Document Object Model (DOM). It also provides best practices and in-depth explanations of JavaScript features.
- [MDN JavaScript Guide](#)

Scrimba

7. Learn React for Free:

- Scrimba offers interactive tutorials for learning React. The platform provides video lessons with interactive coding environments, allowing you to write and execute code directly within the lessons. This hands-on approach helps reinforce learning and improve retention.
- Scrimba React Course

8. Learn JavaScript for Free:

- Scrimba also provides interactive tutorials for learning JavaScript. The course covers fundamental concepts and advanced topics, including ES6+ features, and offers practical coding exercises to help you master JavaScript.
- Scrimba JavaScript Course

CSS-Tricks

9. CSS-Tricks:

- CSS-Tricks offers guides, tips, and tricks for all levels of CSS users. The site features articles, tutorials, and examples on a wide range of CSS topics, from basic styling to complex layouts and

animations. It also includes a comprehensive Almanac that details CSS properties and selectors.

- [CSS-Tricks](#)

W3Schools

10.CSS Tutorial:

- W3Schools provides an easy-to-follow tutorial on CSS basics and advanced topics. It covers a wide range of CSS features, including selectors, properties, responsive design, and animations. The site includes interactive examples and quizzes to test your knowledge.
- W3Schools CSS Tutorial

11.JavaScript Tutorial:

- W3Schools offers an easy-to-follow tutorial on JavaScript basics and advanced topics. It covers fundamental concepts, DOM manipulation, event handling, and modern JavaScript features. The tutorial includes interactive examples and exercises.
- W3Schools JavaScript Tutorial

12.MySQL Tutorial:

- W3Schools provides tutorials on MySQL, covering everything from basic queries to advanced database management. It explains how to use MySQL for data storage, retrieval, and manipulation, and includes practical examples and exercises.
- W3Schools MySQL Tutorial

Django

13.Django Documentation:

- The official Django documentation offers extensive guides and resources for learning Django. It covers everything from installation and setup to advanced features like custom middleware and security practices. The documentation also includes a tutorial that guides you through building a Django application from scratch.

- [Django Documentation](#)

14.Django REST Framework:

- The Django REST framework documentation provides detailed information and tutorials for building RESTful APIs with Django. It covers key concepts such as serializers, viewsets, routers, and authentication, and includes examples to help you implement these features in your applications.
- [Django REST Framework Documentation](#)

For Embedded system:

Official Documentation

1. **ESP8266EX Resources:** Official documentation from Espressif, the manufacturer of the ESP8266 chip. It includes datasheets, technical reference manuals, and other valuable resources.

Books

1. **"Programming with NodeMCU" by Dogan Ibrahim:** This book provides comprehensive guidance on using NodeMCU, covering Lua scripting and practical projects.
2. **"Internet of Things with ESP8266" by Marco Schwartz:** A book focused on using ESP8266 in IoT projects, including tutorials and project ideas.

Online Tutorials and Guides

1. [NodeMCU Documentation](#): The official NodeMCU documentation provides a detailed guide on getting started, API reference, and Lua programming.
2. [Random Nerd Tutorials](#): A popular website offering numerous tutorials on ESP8266, NodeMCU, and related IoT projects.
3. [Instructables](#)

GitHub Repositories

1. [**NodeMCU Firmware**](#): The official NodeMCU firmware repository.
2. [**ESP8266 Arduino Core**](#): This repository allows you to program the ESP8266 using the Arduino IDE. It includes comprehensive documentation and examples.

YouTube Channels

1. [**The Hook Up**](#)
2. [**Tech Note Ph**](#)

Online Communities and Forums

1. [**ESP8266 Community Forum**](#)
2. [**Stack Overflow**](#)