Princess Sumaya University for Technology

King Hussein School for Computing Sciences

Malware Analysis in PDFs

Dr. Ammar Odeh

Software Engineering Course (13477)

Fall - 2022/2023

Prepared By:

Rand Abdel Fattah 20190231

Anton Bahou 20190144

Hassan Abualhaj 20200500

Mohammed Madi 20200386

Contents

Table of Figures

Table of Tables

Chapter 1: Problem Definition

1.1 Introduction

Malware, short for malicious software, has a self-explanatory name. Whether that malicious intent was to cause damage, leak information, or gain unauthorized access, it is without a doubt unwelcome. Just as technology has grown exponentially in its innovation and importance, the world of malware hasn't been slacking behind.

With cybersecurity risks being on the rise, the purposes and intents of malicious attack perpetrators became more varied. Cyberattacks over the last few years uncovered many culprits targeting governments and corporations. The motive for many of these attacks resides in growing the black market of activities including selling leaked private information, identity theft, bank fraud, etc.

Analyzing malware generally falls into two categories. The first is analyzing them statically, which means that the malware is examined without executing the malware itself. The second approach is analyzing them dynamically, which means that the malware is studied by looking at its behavior as it is being executed. The static approach is much simpler than the dynamic approach. Which approach to take is also dependent on the nature of the malware that is being analyzed. A malware being a virus, ransomware, or trojan etc. can have an effect or limitation on the way it will be treated in the analyzation process.

The rampant growth of the internet, cloud services, digital communication services and many more technologies has given malware a new medium to emerge through. The vast amounts of Portable Document Format (PDF) files being transferred between individuals makes them appealing to malicious attackers and threatening to unaware individuals to the same extent.

This paper will discuss the importance of analyzing malware and current systems that analyze malware in PDFs, and propose a potential system based on the shortcomings of the current ones.

1.2 System Description

Our system's main aim is to detect any Portable Document Format (PDF) file that is embedded with malware using machine learning.

Our system uses Logistic Regression algorithms and contains a model that detects embedded malware in PDF files.

In our approach supervised machine learning will be used, which means our dataset is labeled and will accurately classify previously unseen data into two labels; malicious or safe.

Our models will be trained using 70% of the dataset, and their performance will be tested using the remaining 30%.

1.3 System Purpose

Rapid development of technology resulted in an increased use of PDFs. The side-effects of such an increase in use of this format brought forward new malware attacks on people. The attackers conceal their preferred method of malware into the PDF using steganography and wait for the user to fall into the trap.

The purpose of our research is to provide a method that prevents future cyberattacks through the use of Artificial Intelligence practices, these practices will bring forward a mechanism that understands the behavior of such malware as well as its purpose. These understandings will be established through datasets of such attacks, then incorporated into machine learning models and statistical methods for a well-rounded mechanism.

Our intention is to deliver the best f1-score possible, as we believe security is one of the most important aspects of software engineering. We also want to deliver a method that can be utilized by many devices, no matter their capabilities, as people these days use an enormous range of devices each with a different processing power forcing our method to be executed in an acceptable complexity.

1.4 Problem Statement

PDF files are some of the most common files for an attacker to use as a “trojan”. Simply because it's very easy to convince a victim to use it through social engineering. Maybe it looks like a research paper, or some class notes from a friend. Luckily A Lot of these attacks can be stopped if the attack is using a previously seen “Signature”. A signature is a pattern in the malware that allows security software to detect malicious software. It could be a sequence of bytes over a network or a hidden sequence of code within a file, in this case within a PDF. The issue arises when an attack uses a never-before-seen signature, probably an undiscovered vulnerability; a so-called Zero-day attack.

This is a big problem because of the huge space of possible different signatures which have never been seen before. 80% of all successful data breaches in 2019 were the result of a zero-day attack. It's estimated that 42% of all attacks in 2021 were zero-day attacks.[3]

The focus of this paper is to offer a machine learning based alternative which performs better against zero-day attacks while also performing well against previously seen signatures.

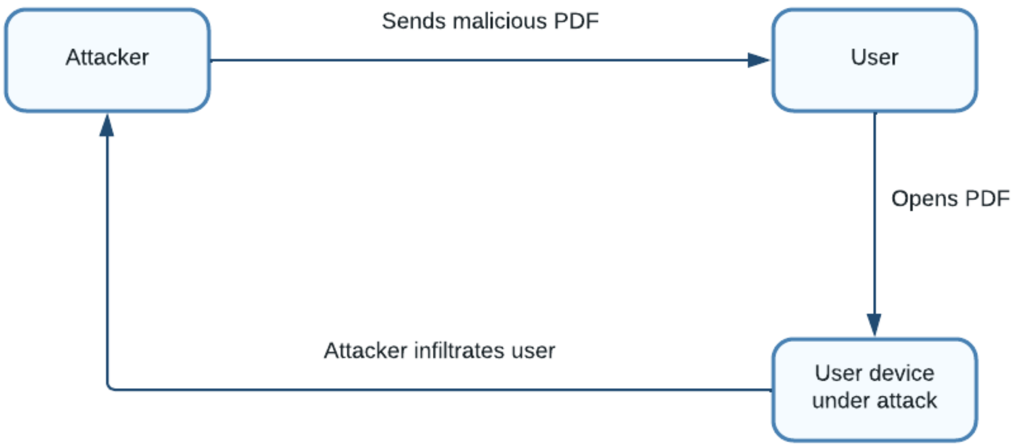


Figure 1: Malicious PDF/Image/Doc Attack Scenario

1.5 System Context View

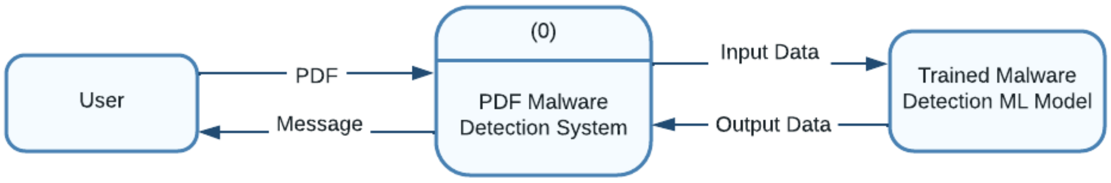

User
PDF
Message
(0)
PDF Malware Detection System
Input Data
Output Data
Trained Malware Detection ML Model

Figure 2: The System Context View

1.6 Literature Review

Dr. Robert Chun, Dr. Mike Wu, and Mr. Navneet Goel studied the structure of PDFs and why it's a sweet spot for attackers to embed several viruses and malware. They also discovered some techniques that detect malicious files such as forensic analysis. They talked about the importance of using machine learning to analyze and stop attacks. They also discussed the need for continuous enhancements of machine learning techniques, as attackers will exploit the vulnerabilities in these techniques sooner or later.[1]

Priyansh Singh, Shashikala Tapaswi, and Sanchit Gupta mentioned in their paper that in order to attack a document you can either target the document itself, target a software that opens this document, or target both of them. To target a certain document, you need to exploit some of the features of this document. When creating a document with interactive features or visual elements like audio or video, you can easily embed a visual element that loads and executes malware without the user noticing any foreign actions as these elements are normally provided by the environment. This method is called the exploitation of programming and interactive application features. While the structural attack uses the Crafted-Content method to target the software that opens certain documents, it changes the structure of the document which causes the software to react unexpectedly. Piggybacking on this reaction helps embedding malicious code in the memory for future execution. This might lead to a buffer overflow which is considered usually as a starting point in an attack.[2]

In A. Corum, D. Jenkins, and J. Zheng’s paper, PDFs were transformed into a grayscale version using byte values and Markov plots to allow the extraction of important features for the development of the classification model. These features were based on keypoint descriptors and texture features. This model was created by a learning algorithm that used Random Forest, Decision Trees and K-Nearest Neighbor incorporated with mathematical functions in the field of vectors to detect malware in PDFs. Their performance was similar to the level of popular antivirus scanners. The dataset used was from Contagio.[4]

Another interesting research was done by F. Mercaldo and A. Santone. The researchers also converted the images into a PNG grayscale version; they used a script on a GitHub repository to achieve that.[5] The grayscale image was then converted into a histogram of pixel intensity values that was entered into a 3-layer deep neural network. The neural network was responsible for detecting if the image is a malware and finding out which family of malware it belonged to. The dataset used belonged to AMD. 17 different Algorithms were used and compared, 12 of them were deep learning algorithms.[6]

This paper, which was written by H. D. Samuel, M. S. Kumar, R. Aishwarya and G. Mathivanan, focused on the problems caused by multiple algorithms for steganography. A special software for template matching was used to solve this issue. Grayscale conversion was utilized again in combination with a machine learning algorithm to identify the spyware hidden in the RGB layer. The system requires a lot of resources however a high accuracy is achieved. The image goes through a process called regression to separate it and classify it correctly according to the type of malware as part of the detection process and handled with accordingly.[7]

Since images are closely related to PDFs, papers [6] and [7] were used to find more viable ways of analysis for PDFs to be used in our approach.

In his paper, Sultan S. Alshamrani talked about how modern antivirus software have become outdated and don't offer enough protection against malicious PDFs. He proposes a Random Forest ML model that can identify JavaScript and malicious API calls in PDFs by analyzing the documents statically and dynamically using a non-signature method in order to perform well against zero-day malware attacks, as well as alternative Random Forest, Stochastic Gradient Boosting and Support Vector classifiers. The proposed model achieved an F1 score of 0.986, making it more efficient that all the tools referenced in the paper.[8]

This paper by K. O. Babaagba and S. O. Adesanya talks about the effects of feature selection and the ML approach used on the yielded results. The paper tests various supervised and unsupervised classification and clustering ML algorithms with and without feature selection. The paper concludes the machine learning approach in detecting malware is effective, and that using feature selection based on Information Gain leads to better values in performance metrics. The detection conducted in this paper is focused on viruses, and states that similar work should be done on other types of malwares to observe the process more accurately.[9]

Ke He and Dong Kim explain that RNN (Recurrent Neural Network), the most widely used neural network for detecting malware, is vulnerable to redundant API injections. They propose an alternative technique which utilizes a CNN (Convolutional Neural Network), it works by turning the Malware information into an image, then processing it through the network. Doing this, as they explained, solves the redundant API injection problem.[10]

Seong Il Bae, Gyu Bin Lee and Eul Gyu Im explain that ransomware is different from normal malware, most normal signature-based detection methods have an issue detecting zero-day ransomware attacks. They offer an alternative machine learning based approach. First, they process the executable file and turn it into N-gram vectors using CF-NCF, which is similar to TF-ITF. Then, they use several distinguishable machine learning algorithms, the most notable was Random Forest which yielded an f1-score of 99.52.[11]

1.7 Challenges

- Finding, preprocessing, and merging the appropriate datasets. Getting data from multiple sources and ensuring that the data is preprocessed correctly to ensure optimal performance from the model.
- Choosing the correct ML models. Different models use various ML techniques. Finding the suitable model
- Fine tuning the models and their hyperparameters and checking their performance metrics.

1.8 Projection

Table 1: Phase 1 Tasks

Activity	Predecessor	Time (Days)
Planning and Splitting Tasks	-	1
Introduction	Planning and Splitting Tasks	2
System Description	Literature Review	1
System Purpose	Literature Review	1
Problem Statement	Planning and Splitting Tasks	2
System Context View	System Description	1
Literature Review	Introduction AND Problem Statement	3
Challenges	System Description	1
Projection	-	1

Table 2: Phase 2 Tasks

Activity	Predecessor	Time (Days)
Brainstorming	Finish Phase 1	3
Functional Requirements	Brainstorming	2
Non-Functional Requirements	Brainstorming	2
Class Diagram	FR and NFR	1
Use Cases	FR and NFR	2
Activity Diagram	FR and NFR	1

Table 3: Phase 3 Tasks

Activity	Predecessor	Time (Days)
Sequential Diagram	Finish Phase 2	1
Brainstorming	Sequential Diagram	5
Coding	Brainstorming	5
Prepare Simulation Results	Coding	2
Conclusion	Simulation Results	1

1.8.1 Gantt Charts

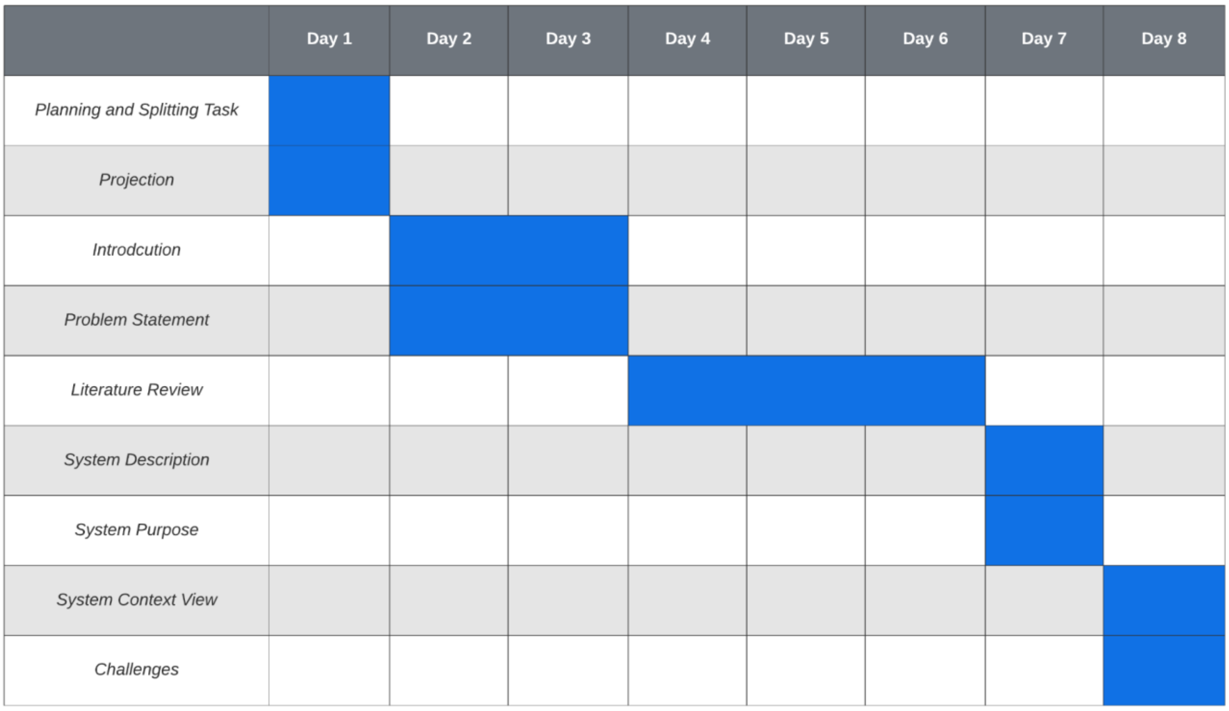
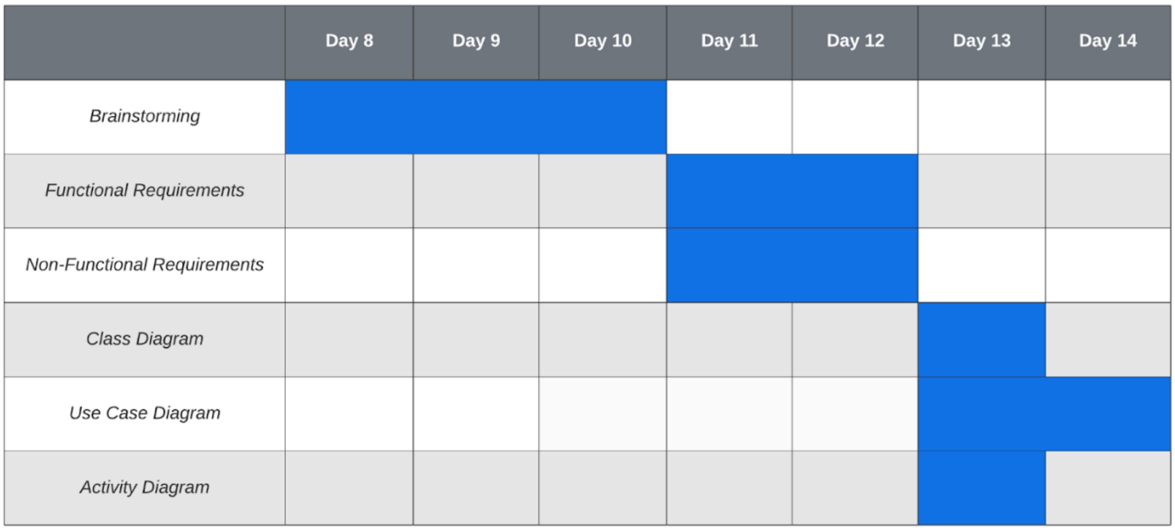
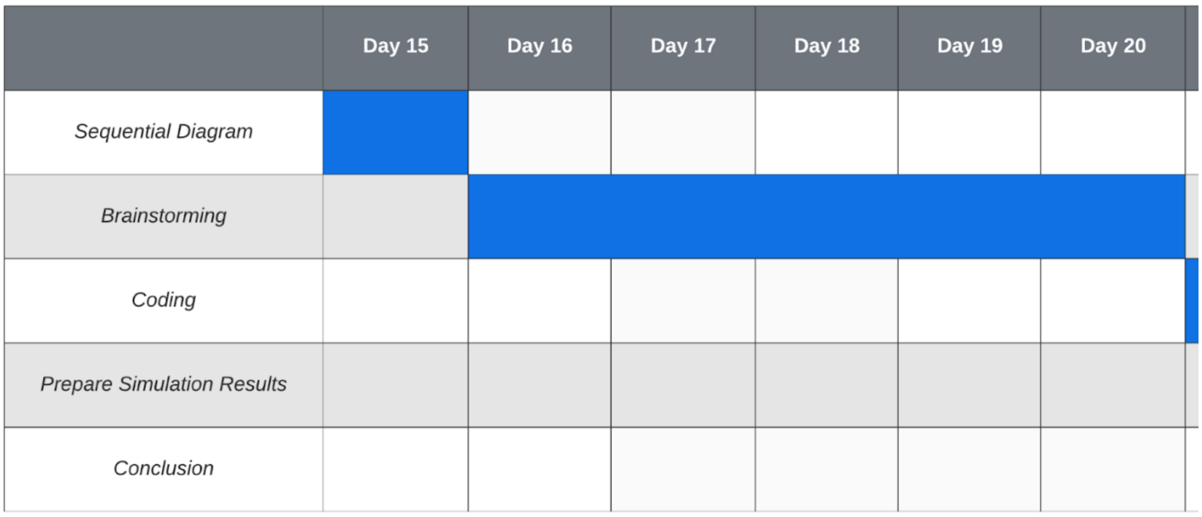


Figure 3: Phase 1 Gantt Chart



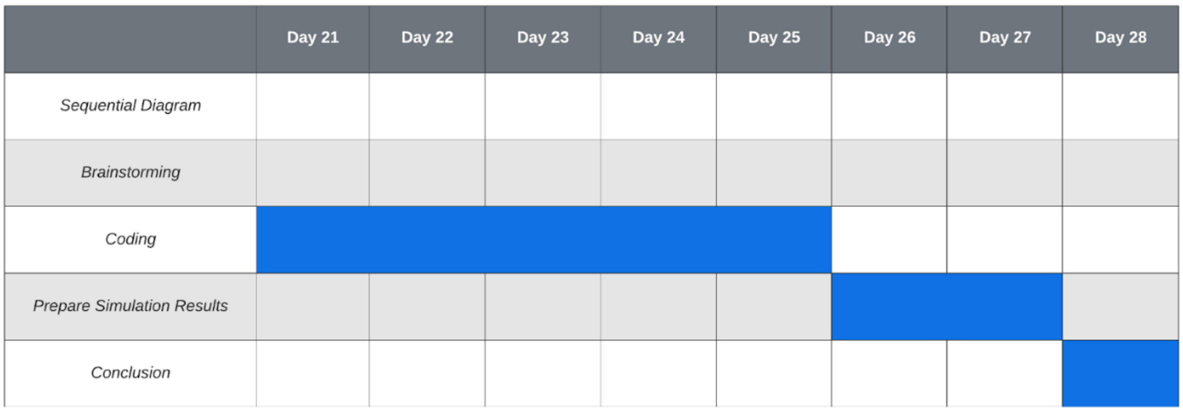
	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
Brainstorming	🟦	🟦	🟦				
Functional Requirements				🟦	🟦		
Non-Functional Requirements				🟦	🟦		
Class Diagram						🟦	
Use Case Diagram						🟦	🟦
Activity Diagram						🟦	

Figure 4: Phase 2 Gantt Chart



	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20
Sequential Diagram	🟦					
Brainstorming		🟦	🟦	🟦	🟦	🟦
Coding						
Prepare Simulation Results						
Conclusion						

Figure 5: Phase 3.1 Gantt Chart



	Day 21	Day 22	Day 23	Day 24	Day 25	Day 26	Day 27	Day 28
Sequential Diagram								
Brainstorming								
Coding	🟦	🟦	🟦	🟦	🟦			
Prepare Simulation Results						🟦	🟦	
Conclusion								🟦

Figure 6: Phase 3.2 Gantt Chart

1.8.2 Network Diagrams

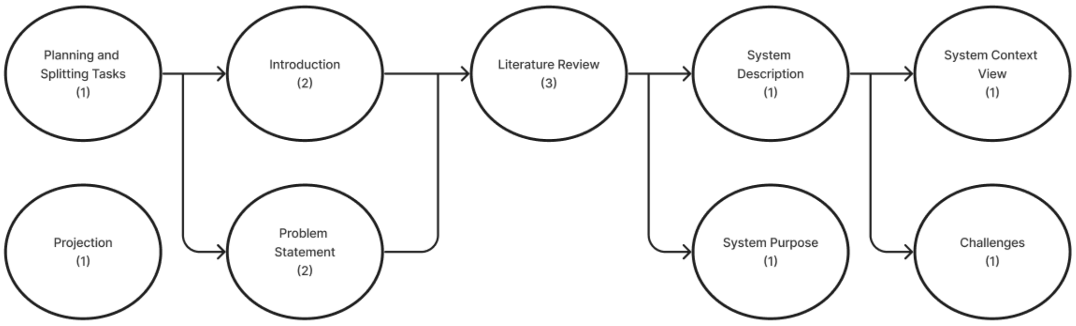


Figure 7: Phase 1 Network Chart

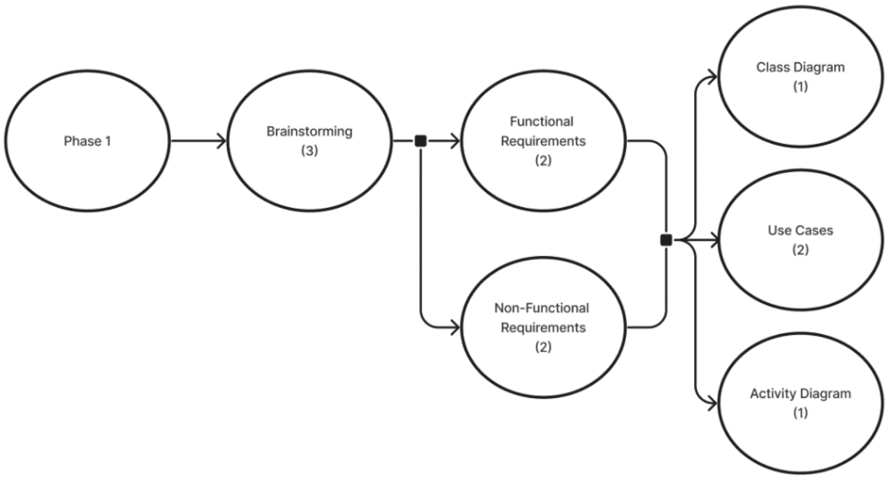


Figure 8: Phase 2 Network Chart

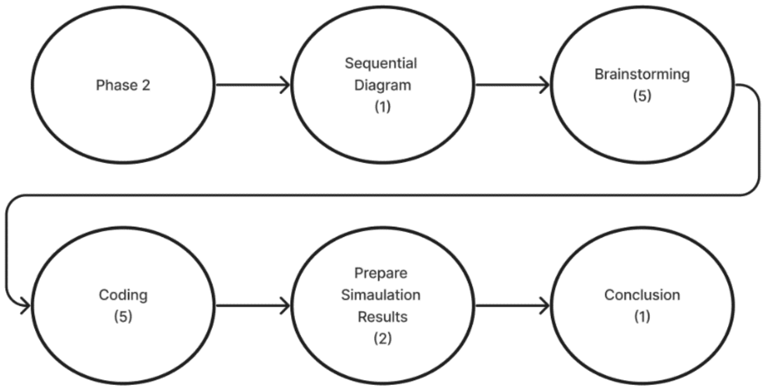


Figure 9: Phase 3 Network Chart

Chapter 2: Analysis and Design

2.1 Functional Requirements

Table 4: Functional Requirements

ID	Name	Description
FR1	Scan PDFs	The system must be able to scan PDF files for known malware signatures or other indicators of malicious issues.
FR2	Detect Malicious PDFs	If the system detects a PDF file that is potentially malicious, it must alert the user and provide information about the detected threat.
FR3	Handle Malicious PDFs	The system must be able to delete malicious PDF files to prevent them from being opened or executed.
FR4	Run Scans	The system must be able to run scans on demand.
FR5	Update ML Model	The system must be able to update its model regularly using new malware signatures to ensure that it can detect the latest threats.

2.2 Non-Functional Requirements

Table 5: Non-Functional Requirements

ID	Name	Description
NFR1	Accuracy	The system must be accurate enough in detecting malicious PDF files, with a low rate of false positives and false negatives.
NFR2	Speed	The system must be able to scan PDF files quickly, without causing delays or performance issues.
NFR3	Portability	The system must be work on a group of different hardware and software platforms.
NFR4	Ease of Use	The system must be easy to install and configure, with clear instructions making it easy to use. A user-friendly interface with minimal controls. It must be well-documented as well with a documentation that explains its capabilities.
NFR5	Availability	The system must be available as much as possible whenever the user needs it with minimal crashes or interruptions.
NFR6	Hyperparameter Optimization	Optimizing model hyperparameters to increase the accuracy and performance of grid search.

2.3 Class Diagram

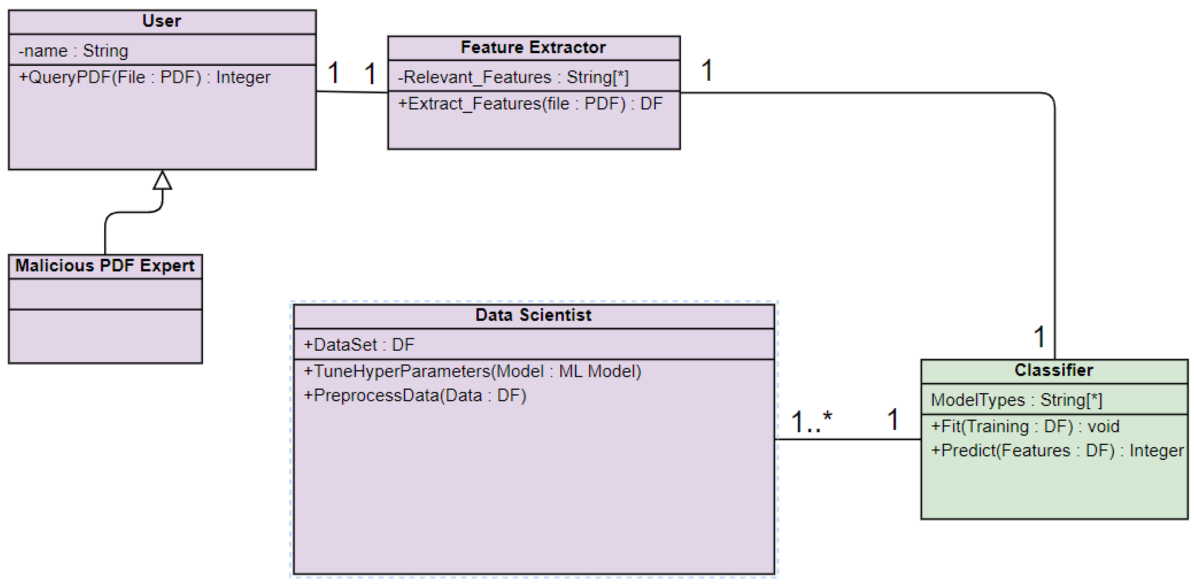

User
-name : String
+QueryPDF(File : PDF) : Integer
1
1
Feature Extractor
-Relevant_Features : String[*]
+Extract_Features(file : PDF) : DF
1
Malicious PDF Expert
Data Scientist
+DataSet : DF
+TuneHyperParameters(Model : ML Model)
+PreprocessData(Data : DF)
1..*
1
1
Classifier
ModelTypes : String[*]
+Fit(Training : DF) : void
+Predict(Features : DF) : Integer

Figure 10: Class Diagram

The Class diagram Above shows the relationship between the User, The Classifier and the Data scientist that builds the Classifier.

If the user wants to scan a PDF it will first go through a feature extraction module which will gather the relevant features within that pdf, then it will be passed to the classifier which predicts whether the pdf is malicious or not.

2.4 Use-Case Diagram

2.4.1 Malicious PDF Detection System

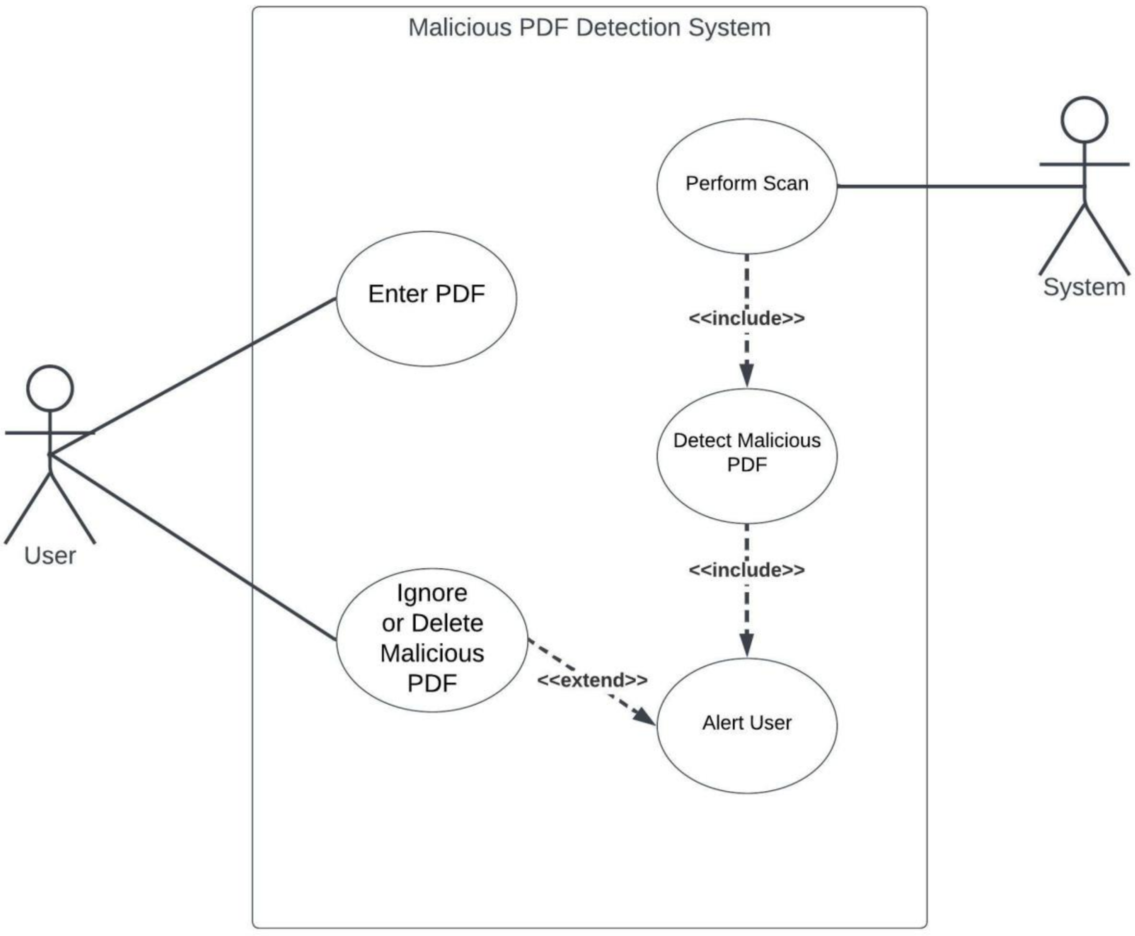


Figure 11: Malicious PDF Detection System Use-Case

Expanded Use-Case:

- Use-Case: Enter PDF
- Actor: User (initiator).
- Purpose: Alert the user if the PDF they entered is malicious.
- Overview (Success Scenario):
  The user will supply the model with the PDF they want to scan. The system will proceed to scan the PDF and detect if it contains malicious content based on the model. Accordingly, the user will get alerted about the malicious PDF with a pop up that asks them to decide what should happen with the PDF. The user proceeds to choose between quarantining or deleting or leaving the PDF.
- Type: Primary
- Cross Reference: FR1, FR2, FR3, FR4

Table 6: Malicious PDF Detection System Typical Course of Events

Actor Action	System Response
1.This Use Case begins when a user decides they want to scan a certain PDF.	
2. The user inputs the PDF they want to scan into the model using the provided interface.	3. The system proceeds to scan the PDF for malicious content.
	4. The system detects if the PDF contains malicious content.
	5.The system alerts the user of the malicious PDF using a pop-up with options of what to do with it.
6.The user decided if they want to delete or ignore the malicious PDF.	7. The system proceeds to delete or ignore the malicious PDF according to the user's choice.

Alternatives:

Line 2. The user inputs an incorrect file format. Indicate error.

Line 6. The user chooses to exit the pop-up without choosing. The system asks the user if they are sure they want to keep the PDF as it is (Same as ignoring).

2.4.2 Model Enhancement Process Service

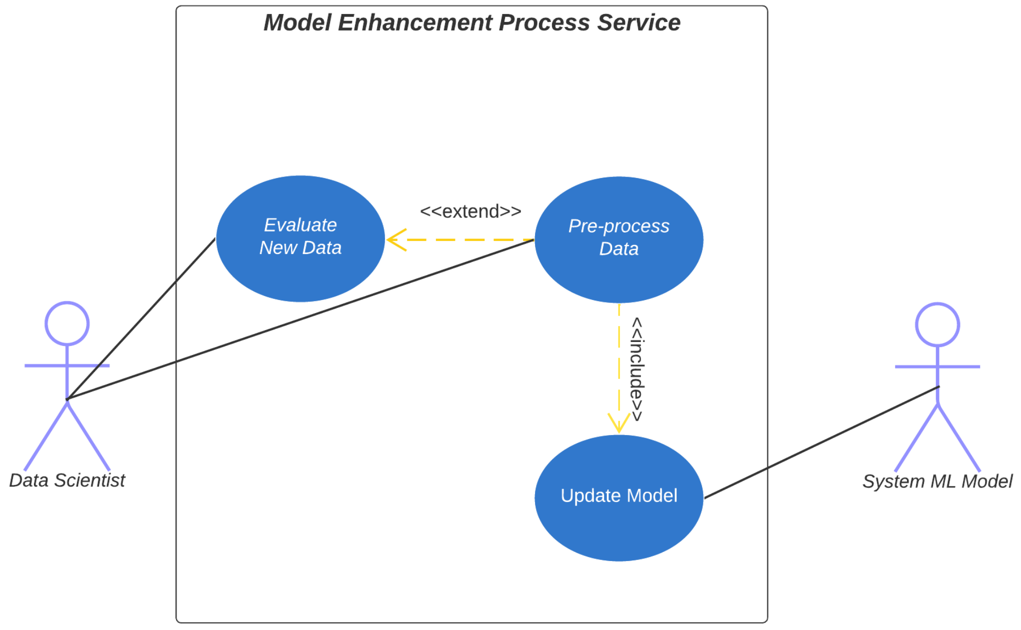


Figure 12: Model Enhancement Process Service Use-Case

Expanded Use-Case:

- Use-Case: Evaluate New Data
- Actor: Data Scientist (initiator)
- Purpose: Update Machine Learning Model
- Overview (Success Scenario):
  The Data Scientist finds new data that they might want to include in their model. The Data Scientist proceeds to evaluate the new data to decide whether it has any important substance that needs to be included in the model. If the data is important the Data Scientist pre-processes it. The System will then update its machine learning model accordingly.
- Type: Optional
- Cross Reference: FR5

Table 7: Model Enhancement Process Service Typical Course of Events

Actor Action	System Response
1.This Use Case begins when the Data Scientist finds about a new malicious threat not covered by the machine learning model.	
2.The Data Scientist evaluates the new data for important content.	
3.The Data Scientist pre-processes the data in a form suitable for the machine learning model.	4.The system updates its machine learning model according to the pre-processed data provided.

Alternatives:

Line 2. There is no important content in the new data. Stop, no more steps left.

Line 3. The Data Scientist provides an unsuitable form. Indicate error.

2.4.3 Testing Malicious PDF Detection System

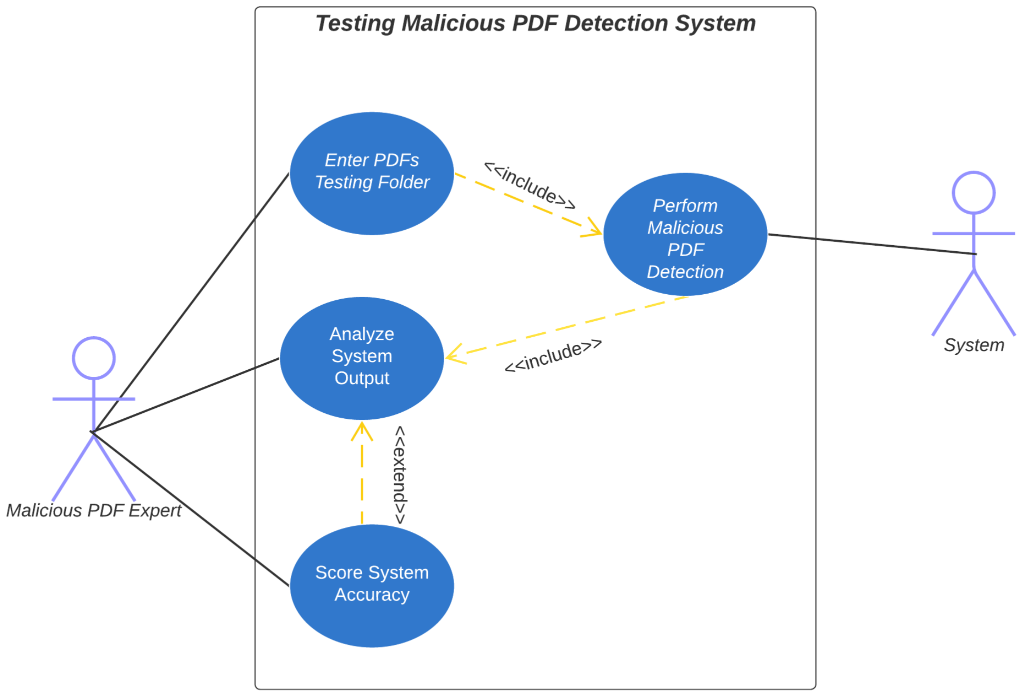


Figure 13: Testing Malicious PDF Detection System Use-Case

Expanded Use-Case:

- Use-Case: Analyze System Output
- Actor: Data Scientist (initiator)
- Purpose: Score System Accuracy
- Overview (Success Scenario):
  The Malicious PDF Expert inputs some malicious and safe PDFs into the system. The system performs detection and outputs its findings. The Malicious PDF Expert analyzes the output of the system with the expected results. The Expert proceeds to score the accuracy of the model.
- Type: Optional
- Cross Reference: FR1, FR4

Table 8: Testing Malicious PDF Detection System Typical Course of Events

Actor Action	System Response
1.This Use Case begins when the Malicious PDF Expert wants to check how accurate the current model is.	
2.The Malicious PDF Expert inputs the PDF they want to scan.	3. The system proceeds to scan the PDF for malicious content.
	4. The system detects if the PDF contains malicious content.
	5. The system outputs its results for the expert to analyze.
6.The Malicious PDF Expert analyzes the output of the system in reference to their own expectations.	
7. The Malicious PDF Expert proceeds to score the accuracy of the model used by the system.	

Alternatives:

Line 2. The Malicious PDF Expert inputs an incorrect file format. Indicate error.

2.5 Activity Diagrams

2.5.1 System Interaction with Document

Below is the activity diagram for the system interaction when receiving a document from the user.

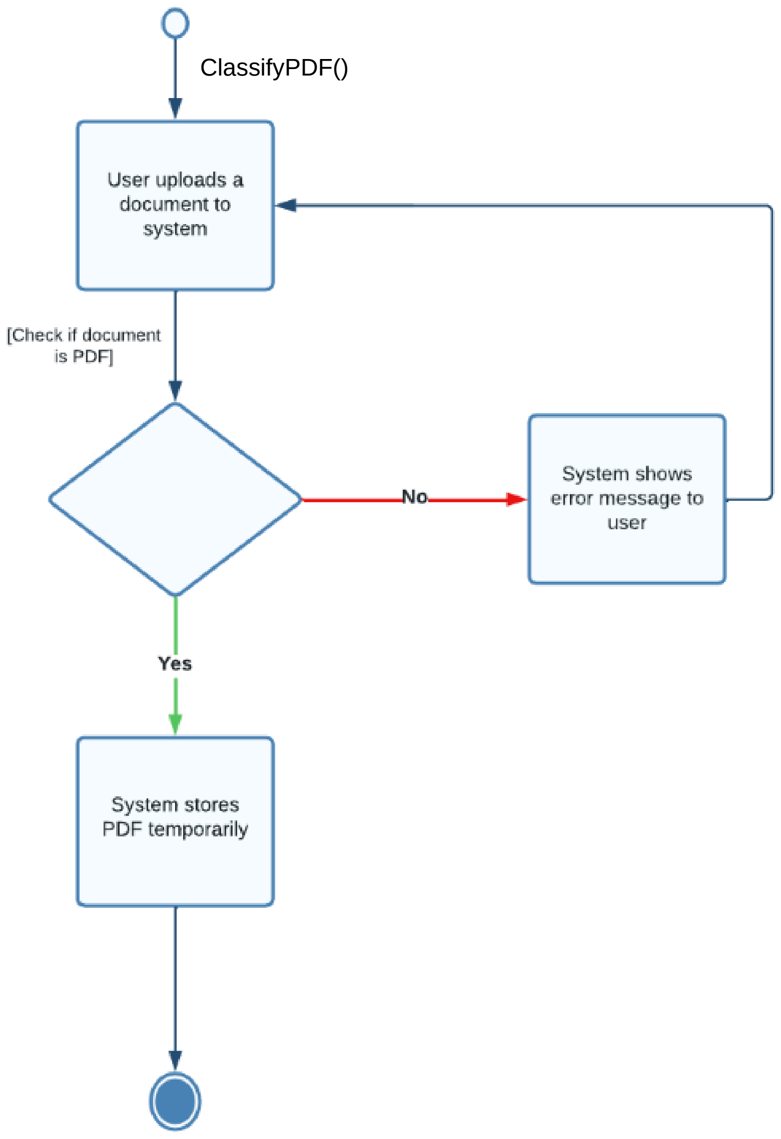

ClassifyPDF()
User uploads a document to system
[Check if document is PDF]
No
System shows error message to user
Yes
System stores PDF temporarily

Figure 14: System Interaction with Document Activity Diagram

The user will upload a PDF document, system will check if it’s a PDF document and stores it temporarily in order to be classified as malicious/ safe.

2.5.2 Classifier

The below activity diagram shows details regarding the classifier.

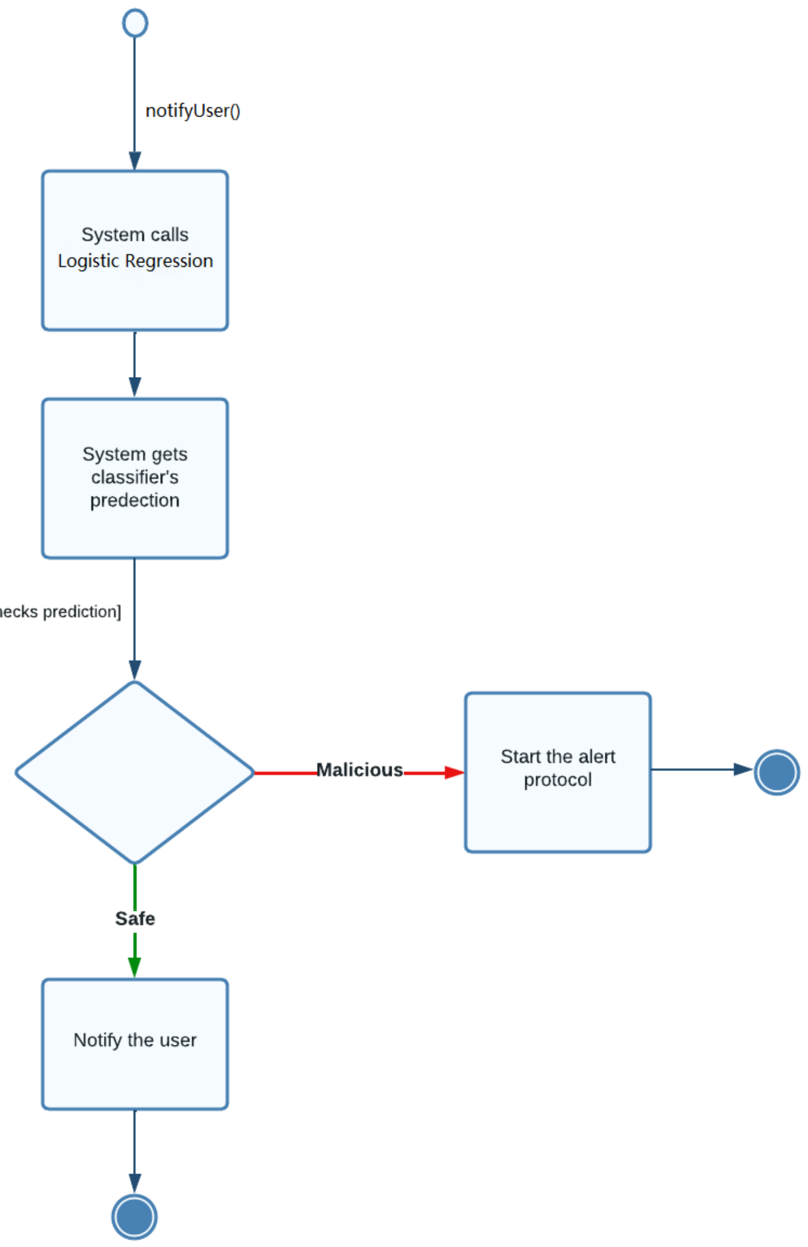


Figure 15: Classifier Activity Diagram

The Logistic Regression Classifier will predict if the document is safe or malicious based on the dataset. If it’s safe the system will notify the user, otherwise the alert protocol will start.

2.5.3 System Alert Protocol

The below activity diagram shows the system’s alert protocol.

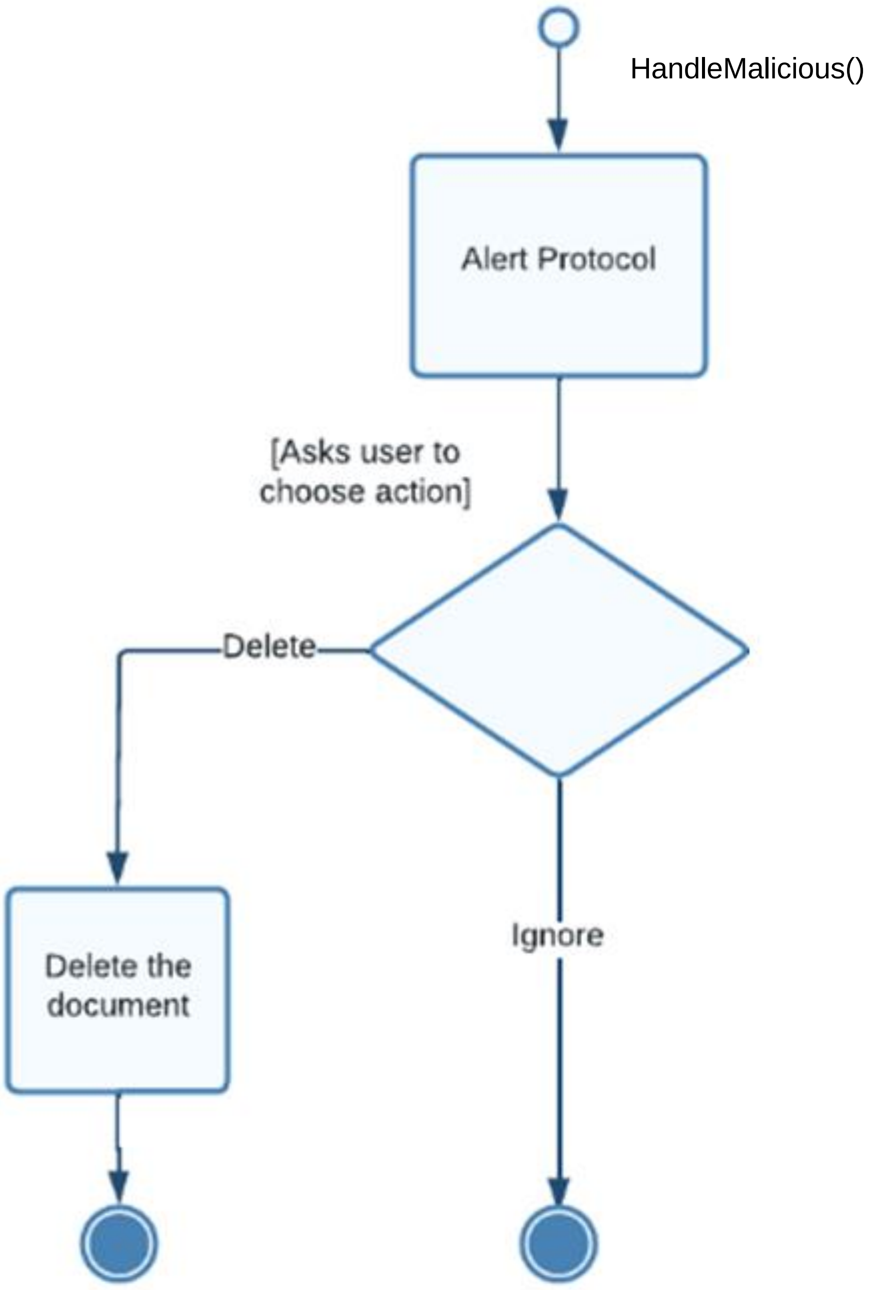


Figure 16: System Alert Protocol Activity Diagram

If the PDF is malicious, the system will give the user 2 options; delete the PDF or ignore the PDF and be done.

Chapter 3: Implementation

3.1 Sequence Diagrams

3.1.1 User Input Sequence Diagram

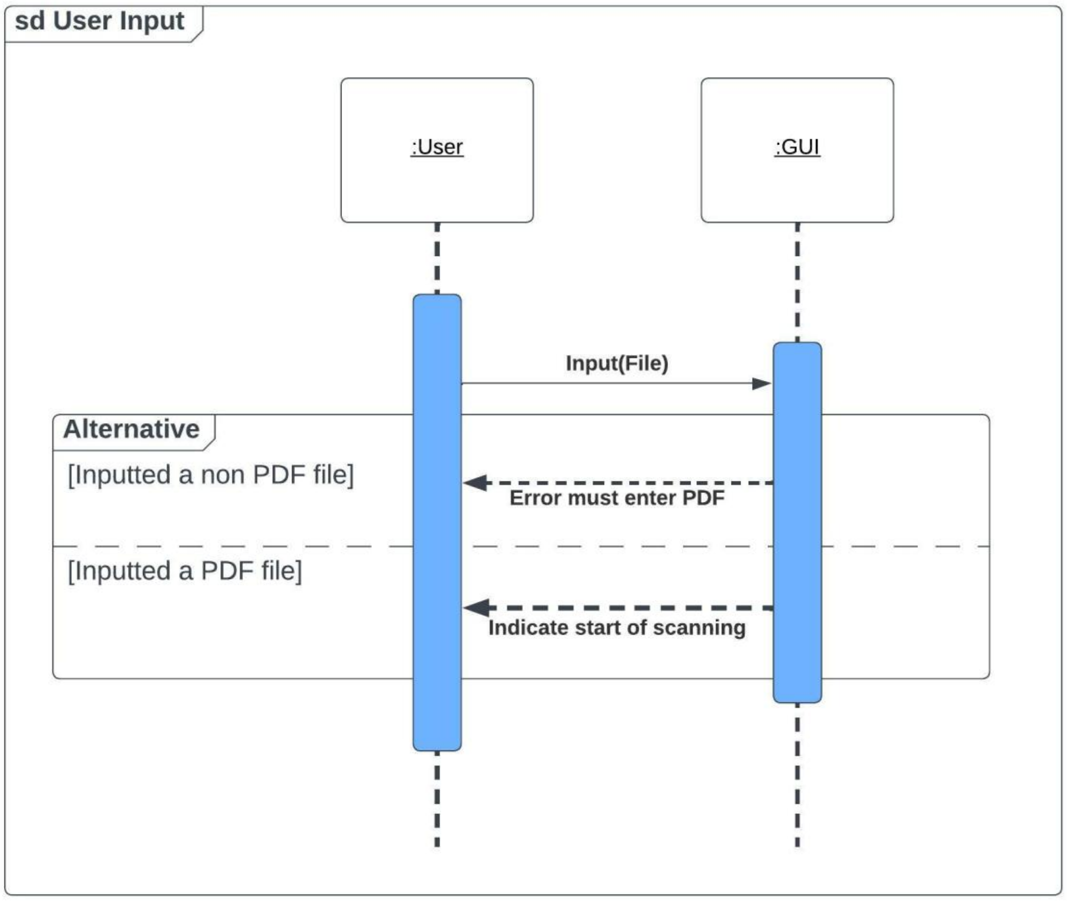


Figure 17: User Input Sequence Diagram

This sequence diagram shows how the User and GUI interact with each other during the input PDF process. The GUI checks if the file is a PDF or not and informs the User.

3.1.2 Process Outline Sequence Diagram

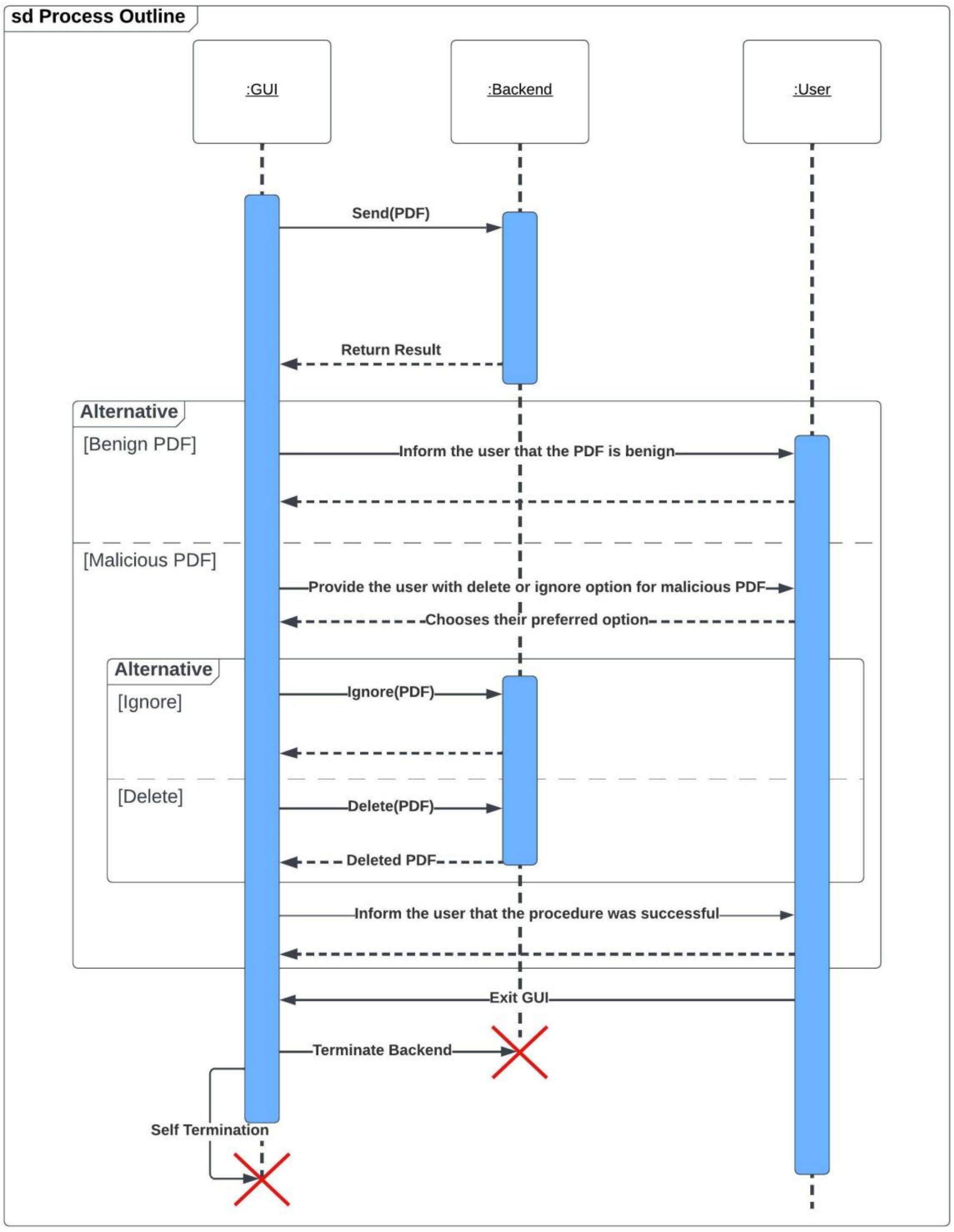


Figure 18: Process Outline Sequence Diagram

This sequence diagram shows how the GUI handles the PDF and what the Backend returns to the GUI to show the User. The GUI sends the PDF to the backend which then returns if it is benign or Malicious. If benign the User is informed and nothing happens. If malicious the User is informed then given 2 options, Ignore or Delete. Ignoring does nothing while deleting deletes the PDF. If the User chooses to exit the GUI termination of the Backend and the GUI happens.

3.1.3 Backend Architecture Sequence Diagram

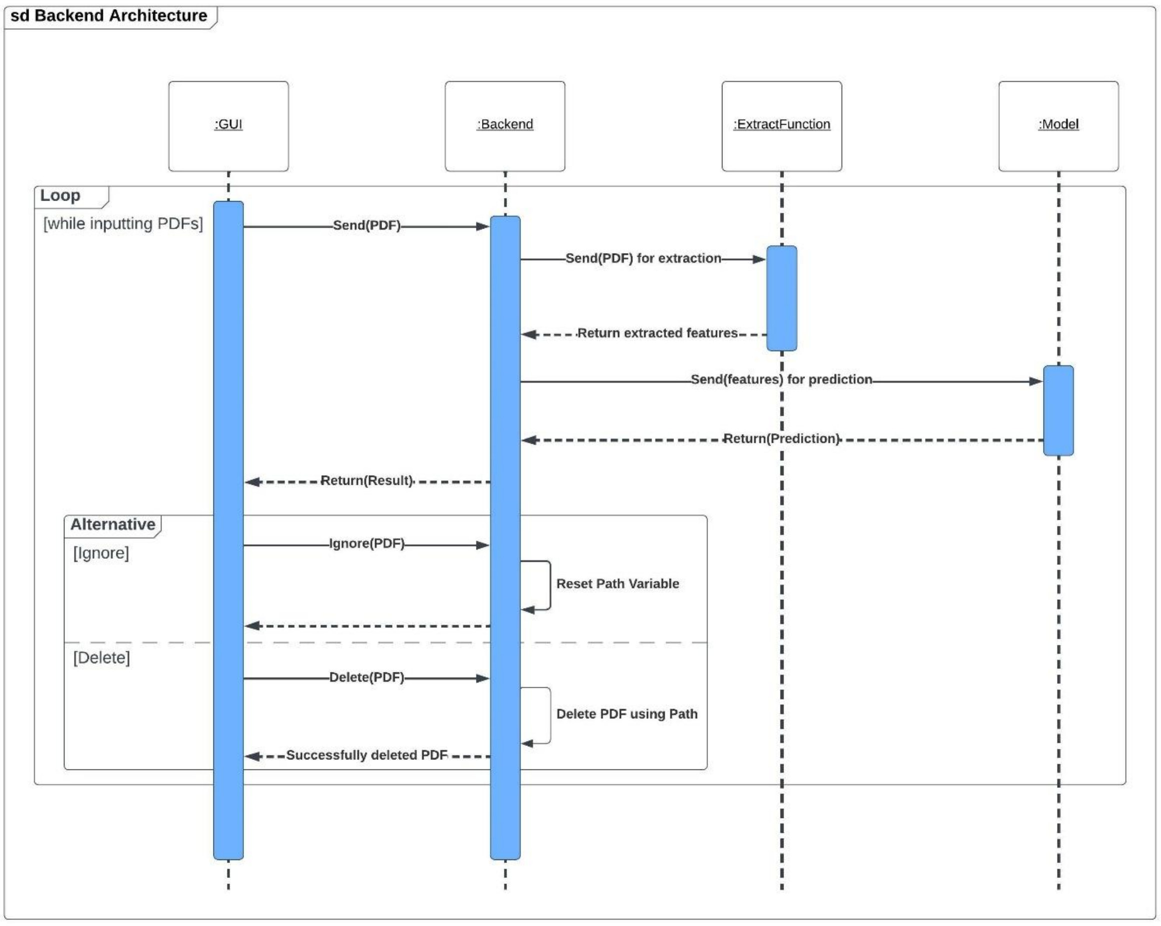


Figure 19: Backend Architecture Sequence Diagram

This sequence diagram shows how the Backend functions exactly depending on what the GUI sends. This process is looped as long as the user inputs PDFs. The GUI sends the Backend the PDF. The Backend sends it to the Extract Function in order to get all of the features of the PDF. The Backend sends the features it received to the Model which predicts if the PDF is malicious or not. The Backend returns the result to the GUI. If the GUI received ignore from the user as shown in previous sequence diagrams, the GUI sends ignore to the Backend which proceeds to reset the path variable containing the path of the PDF. If delete chosen, the GUI sends the Backend delete, the Backend deletes the PDF using its path and informs the GUI that it is deleted.

3.2 Proposal Algorithm

The premise of this paper is to create a system that can take any PDF as input from the user, and inform the user of whether that PDF is malicious or benign based on the prediction of an embedded machine learning model in that system. The model we chose in our system is a Logistic Regression model, due to the fact that the features extracted from the PDF files are all continuous numeric features. Logistic Regression also uses the relationship between all those features to predict the probability of the file belonging to a certain label.

Our system architecture consisted of three main stages; data preparation, learning process, and evaluation:

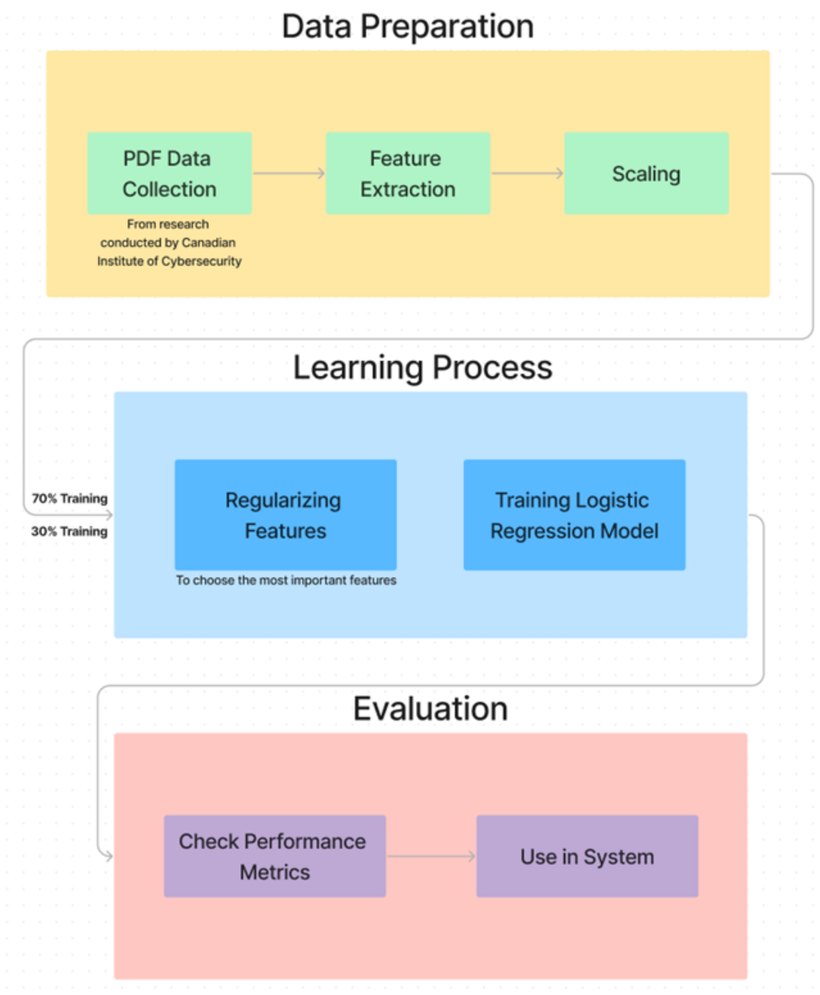

Data Preparation
PDF Data Collection
From research conducted by Canadian Institute of Cybersecurity
Feature Extraction
Scaling
Learning Process
70% Training
30% Training
Regularizing Features
To choose the most important features
Training Logistic Regression Model
Evaluation
Check Performance Metrics
Use in System

Figure 20: Proposed Framework Architecture

The dataset used in this research included 4000 malicious and 4000 benign PDF files. The files were acquired from research conducted by the Canadian Institute of Cybersecurity. Since the dataset just consisted of PDF files, extracting of features from those PDFs was necessary to train the model. Each PDF yielded 21 extracted features, including embedded JavaScript contents, XREFs, byte streams, binary compressed images, and more.

70% of the dataset was then taken to train the model, while 30% was reserved for later use to test the performance metrics. During the training process, features were regularized using Lasso regularization to shrink, or eliminate, the coefficients of the less important features, in order to reduce the model complexity and prevent over-fitting. Lasso regression also deals the issue multicollinearity.

The measures used to check the performance of the outcome model are reflected by the confusion matrix parameters obtain from the classification. The confusion matrix reflects the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) outcomes of the model. These rates are further used to calculate certain performance metrics, like accuracy, precision, recall and F1-score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

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

Figure 21: Performance Metrics Equations

3.3 Simulation Result

Once the logistic regression model proposed by this paper was finalized, it's performance metrics indicated it did incredibly well. The model achieved the following confusion matrix. The matrix shows that the model is well rounded, and doesn't indicate any class imbalance issues.

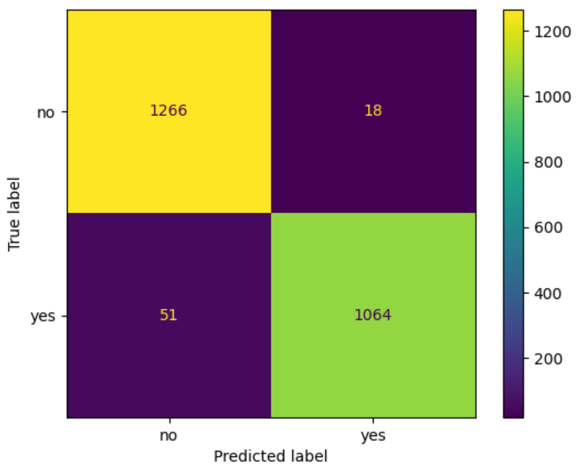


Figure 22: Confusion Matrix

The model achieved an overall accuracy of 0.98, as well as an F1-score of 0.98 as well. As for the precision, recall, and F1-score for each label, the model achieved the following:

Table 9: Performance Metrics

	Malicious	**Benign**
Precision	0.99	0.99
Recall	0.96	0.96
F1-score	0.98	0.98

3.4 Comparison

Table 10: Comparison Table

Reference	Classification Model	Accuracy (%)	F1 Score (%)
[12]	KNN	95.02	-
[12]	CNN	98.76	-
[12]	NAÏVE BYES	89.71	-
[12]	RANDOM FOREST	92.01	-
[13]	ResNet-50 CN N	89.56	-
[14]	SVM	99.6	-
[15]	MLP	96	96
[15]	RF	92	92
[16]	AdaBoost	-	96
[17]	DT	-	65.8
Proposed Model	Logistic Regression	98	98

The previous table compares our proposed model compared to other models present in the papers we referenced. As you can see our accuracy is one of the best accuracies available; however, it is not the best which leads us to believe that our model can be developed further and made better. The F1 scores presents a different idea where our model performed the best out of the bunch which proves that our research we needed.

3.5 Conclusion and Future Work

To summarize, our research devised a model that was trained using 4000 clean and 4000 malicious PDFs. The proposed model being Logistic Regression used a feature extraction module that goes over the whole PDF and extracts certain features that are then placed into a CSV file, in total we selected 21 features. Scaling was then utilized due to our abundant number of different features in order to speed up our system. The task the model had to do was simply distinguishing between malicious and benign PDFs. 9 different research papers were analyzed before coming up with our target which was to achieve the same accuracy in less time as this helps in increasing the security of your device.

Our logistic regression model achieved an accuracy and F1-score of 0.98. Our model achieved a promising result, competing with the other available models, especially considering the size of the dataset used and the concise time this research was conducted in.

Our future vision for this research is increasing the performance metrics. Experiencing a slight drop in accuracy compared to the rest of the models was expected as we valued speed of execution in this case; with further training and devising of the model higher accuracy while maintaining the new speed is achievable and would help the detection systems tremendously. If in other future researches the speed of execution was of less importance, more complex models and features could be used to obtain better results.

REFERENCES

[1] S. S. Pachpute, “Malware Analysis on PDF.” *San Jose State University Library*. doi: 10.31979/etd.pf8d-htjh.

[2] P. Singh, S. Tapaswi, and S. Gupta, “Malware Detection in PDF and Office Documents: A survey,” *Information Security Journal: A Global Perspective*, vol. 29, no. 3. Informa UK Limited, pp. 134–153, Feb. 13, 2020. doi: 10.1080/19393555.2020.1723747.

[3] “2022 cyber security statistics trends &amp; data,” PurpleSec, 17-Oct-2022. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/#ZeroDay. [Accessed: 13-Nov-2022].

[4] A. Corum, D. Jenkins and J. Zheng, "Robust PDF Malware Detection with Image Visualization and Processing Techniques," *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*, 2019, pp. 108-114, doi: 10.1109/ICDIS.2019.00024.

[5] Leeroybrun, “Leeroybrun/bin2png: Convert a binary file to a PNG image and then back to binary.,” GitHub. [Online]. Available: https://github.com/leeroybrun/Bin2PNG. [Accessed: 15-Nov-2022].

[6] F. Mercaldo and A. Santone, “Deep learning for image-based mobile malware detection,” *Journal of Computer Virology and Hacking Techniques, vol. 16, no. 2. Springer Science and Business Media LLC*, pp. 157–171, Jan. 13, 2020. doi: 10.1007/s11416-019-00346-7.

[7] H. D. Samuel, M. S. Kumar, R. Aishwarya and G. Mathivanan, "Automation Detection of Malware and Stenographical Content using Machine Learning," *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022, pp. 889-894, doi: 10.1109/ICCMC53470.2022.9754063.

[8] S. S. Alshamrani, “Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files,” *Security and Communication Networks,* vol. 2022. Hindawi Limited, pp. 1–10, Sep. 21, 2022. doi: 10.1155/2022/7611741.

[9] K. O. Babaagba and S. O. Adesanya, “A Study on the Effect of Feature Selection on Malware Analysis using Machine Learning,” *Proceedings of the 2019 8th International Conference on Educational and Information Technology. ACM*, Mar. 02, 2019. doi: 10.1145/3318396.3318448.

[10] He, K. and Kim, D.-S. (2019) “Malware detection with malware images using Deep Learning Techniques,” 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) [Preprint]. Available at: https://doi.org/10.1109/trustcom/bigdatase.2019.00022.

[11] S. I. Bae, G. B. Lee, and E. G. Im, “Ransomware detection using machine learning algorithms,” *Concurrency and Computation: Practice and Experience*, vol. 32, no. 18, 2019.

[12] M. S. Akhtar and T. Feng, “Malware analysis and detection using machine learning algorithms,” *Symmetry,* vol. 14, no. 11, p. 2304, 2022.

[13] T. M. Mohammed, L. Nataraj, S. Chikkagoudar, S. Chandrasekaran and B. Manjunath, "Malware detection using frequencydomain-based image visualization and deep learning," *Proceedings of the 54th Hawaii International Conference on System Sciences*, pp. 7132, 2021.

[14] B. Cuan, A. Damien, C. Delaplace, and M. Valois, “Malware detection in PDF files using Machine Learning,” *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications,* 2018.

[15] J. Torres and S. De los Santos, “Malicious PDF documents detection using machine learning techniques - A practical approach with cloud computing applications,” *Proceedings of the 4th International Conference on Information Systems Security and Privacy,* 2018.

[16] M. Cova, C. Kruegel, and G. Vigna, “Detection and analysis of drive-by-download attacks and malicious javascript code,” *Proceedings of the 19th international conference on World wide web - WWW '10,* 2010.

[17] D. Maiorca, D. Ariu, I. Corona, and G. Giacinto, “An evasion resilient approach to the detection of malicious PDF files,” *Communications in Computer and Information Science*, pp. 68–85, 2015.