PDF Malware Detection based on Stacking Learning

Maryam Issakhani[1], Princy Victor[1], Ali Tekeoglu[2][a] and Arash Habibi Lashkari[1][b]
[1]*Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB), Fredericton, NB, Canada*
[2]*Johns Hopkins University Applied Physics Laboratory, Critical Infrastructure Protection Group, Maryland, U.S.A.*

Keywords: PDF, PDF Malware, Evasive PDF Malware, Malware Detection, Stacking, Machine Learning, Deep Learning.

Abstract: Over the years, Portable Document Format (PDF) has become the most popular content presenting format among users due to its flexibility and easy-to-work features. However, advanced features such as JavaScript or file embedding make them an attractive target to exploit by attackers. Due to the complex PDF structure and sophistication of attacks, traditional detection approaches such as Anti-Viruses can detect only specific types of threats as they rely on signature-based techniques. Even though state-of-the-art researches utilize AI technology for a higher PDF Malware detection rate, the evasive malicious PDF files are still a security threat. This paper proposes a framework to address this gap by extracting 28 static representative features from PDF files with 12 being novel,and feeding to the stacking ML models for detecting evasive malicious PDF files. We evaluated our solution on two different datasets, Contagio and a newly generated evasive PDF dataset (Evasive-PDFMal2022). In the first evaluation, we achieved accuracy and F1-score of 99.89% and 99.86%, which outperforms the existing models. Then, we re-evaluated the proposed model using the newly generated evasive PDF dataset (Evasive-PDFMal2022)as an improved version of Contagio. As a result, we achieved 98.69% and 98.77% as accuracy and F1-scores, demonstrating the effectiveness of our proposed model. A comparison with state-of-the-art methods proves that our proposed work is more resilient to detect evasive malicious PDF files.

1 INTRODUCTION

Over the years, PDF has been the most widely used document format due to its portability and reliability. Unlike the rest of the document formats, such as doc files that may present the content differently on different platforms, PDF files are platform-independent. Despite the appearance, they have a very complex structure with a combination of binary and ASCII data, and they can be considered a whole programming language of their own, which gets executed upon being viewed. PDFs also support various features and elements such as embedding multimedia, JavaScript functionalities and system command injection. These features contributed a lot to the flexibility and efficiency of PDFs.

PDF popularity and its advanced features,have allowed attackers to exploit them in numerous ways. In addition to that, according to (Nissim et al., 2015) most users only associate danger with executable files and do not think of files such as .doc and .pdf as being potentially malicious. Therefore, spreading malware through PDFs has become a common technique in today's online world, where documents are increasingly sent in PDF format compared to paper. In 2008, (Blonce et al., 2008) analyzed the critical PDF features that an attacker can misuse to deliver a malicious payload. The paper demonstrates that attackers have various options to utilize and combine these features to launch an attack. It is worth mentioning that the majority of PDF attacks target Adobe Reader, one of the most popular PDF readers among users. Despite being a powerful PDF viewer, Adobe has many vulnerabilities, often the target for various PDF exploitation. (Maiorca et al., 2019) listed 27 main types of vulnerabilities found in Adobe Reader, such as API overflow and memory corruption which are often exploited by attackers, mainly through using JavaScript.

A plethora of researches, namely (Zhang, 2018)(Liu et al., 2014), discussed that traditional automated detection methods are inefficient for malicious PDF detection as they are mostly signature-based, which allows the malware authors to evade them sometimes, even by applying a basic obfuscation. Therefore, PDF infection detection still requires

[a] https://orcid.org/0000-0001-7638-0941
[b] https://orcid.org/0000-0002-1240-6433

Issakhani, M., Victor, P., Tekeoglu, A. and Lashkari, A.
PDF Malware Detection based on Stacking Learning.
DOI: 10.5220/0010908400003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 562-570
ISBN: 978-989-758-553-1; ISSN: 2184-4356

a degree of manual analysis using different tools.
Main Contributions: Our contributions in this work are as follows:

- We proposed a novel evasive PDF malware detection approach based on Stacking Learning which results in an improved accuracy and F1-score compared with the relevant works.
- We analyzed the shortcomings of the ”Contagio” PDF dataset which is one of the most popular datasets used for PDF malware detection. Based on the findings, we generated a more comprehensive and stable dataset that addresses the specified flaws, and closer to the real-world distribution of the data.

2 LITERATURE REVIEW

To highlight the novelty of our proposed work, we explored the background on PDF document structure, features that attackers typically exploit, and main PDF malware detection approaches.

2.1 Background

Since PDFs are counted as a Page Description Language, they also follow a certain structure which consists of four components as itemized below, and illustrated in Figure 1. Header is usually less than half a line that appears at the beginning of the PDF file and according to (Itabashi, 2011), ”headers follow a specific pattern consisting of the 5 characters %PDF- followed by a version number of the form 1.N, where N is a digit between 0 and 7”. Body is the largest and most important section of all PDF files, which represents whatever content inside the document in terms of *objects*. Each object in a PDF has an index number followed by its type, such as Page, Font, Image Catalog, etc. Cross-Reference Table(Xref) is a table that allows random access to PDF objects and stores each object’s information and location (Carmony et al., 2016). And finally, trailer specifies the root object and the location of the Xref table.

2.2 Exploiting PDF Features For Delivering Malware

According to (Blonce et al., 2008), PDF features can be exploited in various ways to deliver an attack. Inspired by a couple of references (Liu et al., 2014; Blonce et al., 2008), we provide some detail about common features utilized by attackers in a PDF malware.

1. Headers: Headers are the first part that are inspected by Anti Viruses. (Li et al., 2020) indicates that malicious PDF files tend to have obfuscated and malformed headers, whereas this is not typically observed in benign files.
2. Metadata: Metadata is a section that contains information about the file, such as its creation date or description. It is usually exploited by attackers for hiding parts of the shell code in different sections which they later refer, in their actual malicious payload(Liu et al., 2014).
3. JavaScript : JavaScript is the riskiest feature supported in PDFs which is utilized when a link or button is clicked or while filling out a form inside the PDF. Overall, it is a very common attack vector exploited by attackers (Corona et al., 2014) as it can connect to a malicious URL to drop a malware, or execute a malicious shell code.
4. Streams: Streams are mainly used to store binary data such as image files or page composition objects, (Jeong et al., 2019), and they are often encoded using compression filters. Attackers usually hide their malicious JavaScript code, within streams, as they have no length limitation.
5. ObjStm : This keyword denotes the presence of object streams, that are objects placed inside of streams(Carmony et al., 2016). Attackers can utilize this feature for concealing objects containing malicious code by wrapping them inside object streams (Tzermias et al., 2011).
6. Encoding filters: Encoding or compression filters are applied to PDF streams to reduce their size or protect their sensitive content. However, attackers tend to use one or more of these filters, to hide their malicious content (Brandis and Steller, 2012).
7. Action class : This class has elements that operates on certain events such as clicking, dragging etc, which can be easily subverted by attackers to execute a malware (Blonce et al., 2008).
8. OpenAction/AA object type : This object tag is a risky sub group of Action, that denotes an action or script that gets executed automatically once the file is opened. The combination of this feature with executing a malicious JavaScript, is observed in a variety of exploited PDFs (Cross and Munson, 2011).
9. Embedded Files: As PDFs can attach different file formats such as doc, XML, EXE, etc., attacker may use this technique to bind a malicious file to the PDF. (Stevens, 2011).

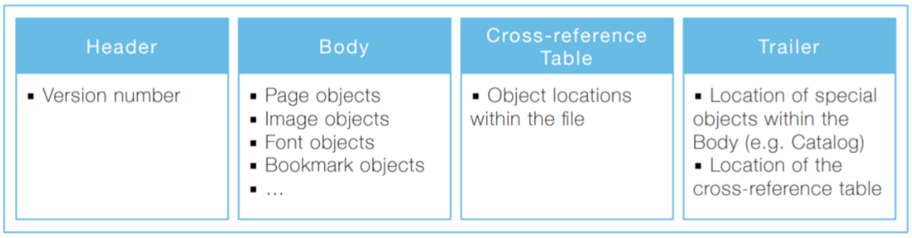
Header
▪ Version number
Body
▪ Page objects
▪ Image objects
▪ Font objects
▪ Bookmark objects
▪ ...
Cross-reference Table
▪ Object locations within the file
Trailer
▪ Location of special objects within the Body (e.g. Catalog)
▪ Location of the cross-reference table

Figure 1: PDF Internal Structure.

10. RichMedia: This feature allows PDFs to embed media files and flash objects inside them, enabling attackers to attach malicious medias to the PDF (Cui et al., 2020).
11. PDF Obfuscation: PDF supports specific obfuscation forms including "Name Obfuscation" where the names and strings inside the PDF can be represented in another form. Hexadecimal Encoding is one example, which enables representing a JS tag as "J#53" or "URI" tag as "#55RI". Attackers tend to utilize this feature, to evade the AV signature detection.

2.3 PDF Malware Detection

Most of the works in PDF malware detection techniques uses two approaches known as static analysis or dynamic analysis.

2.3.1 Static Analysis

Static analysis is the most prevalent approach in the document malware detection, which offers multiple advantages such as a faster detection rate, cheaper implementation cost, simplicity and ease of deployment. Researchers usually perform static analysis by extracting certain features and processing them to classify the input file. Static PDF features can be general features such as size and page number, or they can be extracted from the PDF structure and content. We divided static malware analysis into three subcategories.

1. The first is pure signature-based methods to detect PDF malware, which is the easiest to implement and most anti-viruses rely on for malware detection.
2. The second static analysis type is machine learning in which the relevant features are extracted from the PDF and then fed into the machine learning models for classification (Zhang, 2018). Their proposed model using MLP outperformed eight major AV scanners with 95% TPR and a low FPR of 0.08. Likewise, (Torres and De Los Santos, 2018) compared the performance of three potential classifiers, SVM, Random Forest, and MLP for PDF malware classification. 28 features are extracted from the PDF metadata, structobserved that RF and MLP performed significantly better than SVM.
3. The third static approach focuses on applying deep learning technology to classify PDF malware which appeared in relatively fewer research works, but gaining popularity. In 2020, (Fettaya and Mansour, 2020) proposed a model that uses CNN to detect PDF malware. Their model performs almost on the same level as the best commercial Anti-viruses with no domain knowledge or feature extraction required for development.

An important challenge associated with PDF static detection using learning systems is that they are prone to adversarial attacks, due to which the targeted malware PDF files can evade structural and general features. This is why it is important to apply strategies to mitigate it. (Maiorca et al., 2015) proposes a solution which is specifically robust against evasive attacks by selecting the Adaptive Boosting method to classify the PDF files which trains a set of weak classifiers to build a stronger one. Similarly, (Cuan et al., 2018), proposes SVM as the learning algorithm, and evaluate their model performance with different evasion counter measures. As a result, they completely block the naive evasion attack by setting a feature threshold value.

2.3.2 Dynamic Analysis

Dynamic analysis is another PDF malware detection technique that observes the code's behavior during run time instead of statically analyzing them. Adobe reader has used a sandboxed environment for this purpose where it runs the PDF in the limited locked-down environment to detect any suspicious behavior. Dynamic analysis is generally more reliable than static, and it's usually harder to be evaded by attackers. It is specifically useful to detect the malicious JavaScript code embedded inside the PDF file since malicious JavaScript code is usually obfuscated, making static detection less effective. Unlike static features, dynamic PDF features reveal themselves during runtime. Features such as API calls, network communications, and suspicious memory consumption, which is an indication of heap spraying (Liu et al., 2014), are primary examples. (Liu et al., 2014) combines the static and dynamic analysis by defining a set of both groups of features indicative of malicious activity and their run time detector incurs a negligible delay of 0.093 seconds for each JavaScript code.

3 PROPOSED APPROACH

From the existing works, it is identifiable that static-based approaches are widely used to detect malicious PDFs, and most of them tend to rely on a single machine learning or deep learning model. However, to the best of our knowledge, no research in the literature has detected evasive malicious PDF files by combining different classifiers for predicting the final result, which is a practical approach as identified in (ZhiWei et al., 2017), that applied Stacking for spam email detection.

Unlike a single classifier, stacking uses several classifiers that make unique assumptions on the data to predict the output, leading to better performance when combined in the suitable form. This idea motivated us to propose a detection solution for PDF malware detection based on stacking learning. In this work, 28 static representative features including 10 general features and 18 structural features as shown in the Table 1 are extracted from each PDF file and fed into our proposed model for detection. We provide our testing and implementation in the following sections to demonstrate the efficacy of the proposed approach.

3.1 General Overview

Figure 2 presents the general layout of our proposed framework, which consists of two main modules. The components of the model include:

1. Raw PDF Files: The raw PDF files from the user are taken as input and fed into the Feature Extraction Module.
2. Feature Extraction: The proposed 28 features are extracted from each PDF file regarding the common attack vectors specified in the literature review. In this fashion, 10 general features, 18 structural features are extracted from the PDF, which are specified in Table 1.
3. Detection: The extracted features are passed to the proposed detection model for classification.
4. Prediction Results: The final prediction is a binary label: malicious or benign for each PDF, generated by the model.

3.2 Detection Model

As the main idea behind our approach is based on the Stacking method, we present a brief overview of the Stacking technique before diving into the proposed model. Stacking at the simplest level can be defined as a method that determines the final output based on the outputs of a set of different classifiers known as base learners and meta learners. Base learners predict the output from the original dataset and pass the result into a meta learner, producing the output based on the base learner prediction. These learners can be chosen among any of the machine learning or deep learning classifiers according to their fitness for the problem and other relevant factors.

3.2.1 Proposed Stacking Architecture

In order to determine our proposed architecture, several parameters are considered, and it includes:

1. Number Of Stacking Levels: Given the problem complexity, two stacking levels are preferred, which is a suitable trade-off that could lead to a reasonable accuracy with a low cost.
2. Base Learners: The best stacking practice is to select diverse classifiers so that each classifier brings a different perspective of the data to the table. Given the above facts, 4 base learners are selected, SVM, Random Forest, MLP, and Adaboost. as the potential classifiers. Since stacking is particularly subject to overfitting when the base learners are selected among strong classifiers with a high variance, we selected the base learners among the ones that are intrinsically less likely to overfit, such as Linear SVM and RF and Adaboost. For more complex high variance models such as MLP, we applied the L2 regularized cost function to mitigate the issue. The best combination of these base learners are to be determined in the experimentation phase.
3. Meta Learners: Similar to selecting base learners, the same key factors are considered for selecting the meta-learner as well. The most important one is that the input dataset to the meta learner has one target variable and a minimum of two and a maximum of four features that are generated by the predictions of selected base estimators. Considering key factors such as low complexity of the

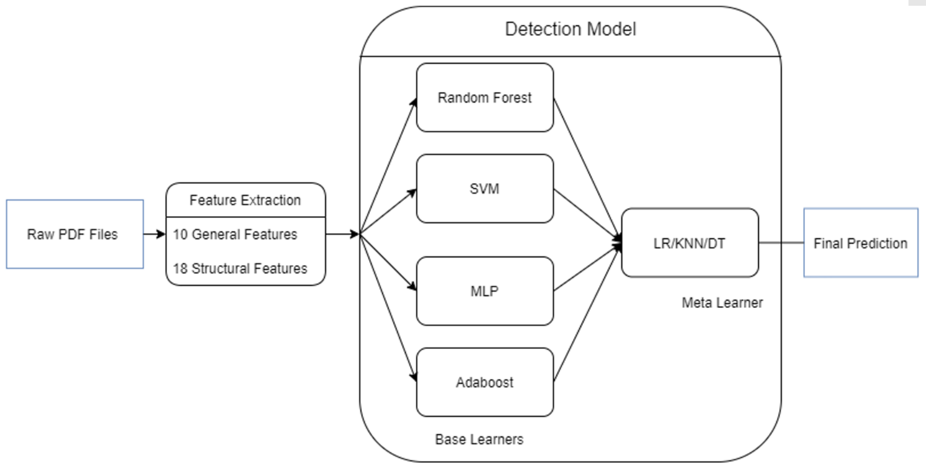

Figure 2: Layout of the proposed architecture.

Table 1: Selected Features.

General Features	Structural Feature
PDF size, title characters, encryption, metadata size, page number, header, image number, text, object number, font objects, number of embedded files, average size of all the embedded media	No. of keywords ”streams”, No. of keywords ”endstreams”, No. of name obfuscations, Total number of filters used No. of objects with nested filters, Average stream size, No. of stream objects (ObjStm), No. of Xref enterie, No. of keywords: ”/JS”, ”/JavaScript”, ”/URI”, ”/Action”, ”/AA”, ”/OpenAction”, ”/launch”, ”/submitForm”, ”/Acroform”, ”/XFA”, ”/JBig2Decode”, ”Colors”, ”/RichMedia”, ”/Trailer”, ”/Xref”, ”/Startxref”

dataset, independence of features, and the binary classification type, three different classifiers, Logistic Regression, KNN, and Decision Tree were shortlisted as potentially fit meta learners for our model. The final selection among the three will be determined according to their performance in our experimentation phase.

4 GENERATING NEW DATASET

In this section, existing datasets for PDF Malware detection are analyzed, and a new evasive PDF malware dataset is generated to improve the old ones. The main objective was to improve the ”Contagio” dataset, which is a well-known PDF repository, and used in the majority of the previous research. Contagio is collected from the Contagio Malware Dump repository, provided by the External Data Source, which consists of 11,173 malicious along with 9,000 benign PDF files. Observing most of the previous research results using Contagio, a notable fact is that most of them reported results with over 99% accuracy, and such a high testing accuracy is not typically achieved. This motivated us to further inspect Contagio dataset to lead to some clues behind the unrealistic high testing accuracy, with the aim of generating a new baseline dataset that overcomes the identified issues, and leads to more accurate analysis results. Analyzing Contagio dataset, the following are two significant issues that we found.

Table 2: Variance and Standard Deviation Comparison Results.

Group	Dataset	Var.	SD
Benign - All	New Contagio	9965.58 591.11	28.27 11.39
Malicious - All	New Contagio	343595.33 7098.43	190.77 23.79
Benign - General	New Contagio	28610.61 1648.5	76.29 27.3
Malicious - General	New Contagio	286987826.99 20405.42	545.23 67.74

Table 3: Base-Learner Results

Base learner	Acc.	Prec.	Rec.	F1
RF	99.87	99.84	99.86	99.86
MLP	99.82	99.83	99.83	99.82
SVM	99.83	99.83	99.83	99.85
Adaboost	99.77	99.74	99.73	99.75

Duplicate Entries: We extracted 28 features from each PDF file in the repository, and based on those features, a high number of duplicate entries are found in the resultant CSV dataset. It was identified that there exist 8,977 duplicate entries, which accounts for 44% of the entire dataset. This indicates a very high similarity between PDF file sample structures used in the Contagio repository.

Lack of Coverage: A good dataset should cover multiple varieties of the samples of interest. In the case of PDF malware, an ideal dataset is one that includes diverse samples among each class of PDF. For testing this, we decided to evaluate the variety of the data points in the Contagio dataset by applying a simple metric referred to as the coefficient of variation. The Coefficient of Variation (CV) is defined as the ratio of standard deviation to the mean of a set of values, and it is used to measure relative variability of a distribution. We applied the CV formula to each of the columns in the dataset, which represent each feature value, and calculated the average of all the results to represent the entire dataset variability with a single number. As a result, we obtained the value of 0.83 for malware samples and the value of 0.79 for benign ones. Since both values are lower than 1, they are considered low variances, indicating a low variation among malicious and benign PDF file classes in Contagio. The result coordinates with what we expected before running our analysis.

Data Bias: Another clear indication of lack of comprehensiveness and proper malware distribution in the Contagio dataset is that more than 0.74 of the malicious data points have a value of more than 0 for the two features (“JavaScript“ and “OpenAction“), whereas only 0.0007 of the benign files have a value of more than 0 for both. This makes the samples almost easily detectable solely on the basis of two features, and it enables a simple linear classifier to produce a minimum accuracy of around (80%) purely based on them. This does not represent the real-world PDF malware exploitation as attacks are getting more sophisticated, and attackers exploit various other features instead of these two to maximize their chances of evasion and hinder being easily detected.

Our main methodology for building the new dataset was to combine the available datasets in a way that the samples meet the criteria that we are looking for. For this, we extracted 28 features from two available dataset resources, VirusTotal and Contagio, and then we deduplicated the records, which left us with 3,223 malicious files and 8,268 benign files from Contagio and 4,868 malicious files from VirusTotal. The final goal in work is to wisely combine the datasets' records into one final file, which results in a more representative dataset of the PDF distribution.

In order to achieve this, we decided to employ K-means, an unsupervised machine learning that clusters the data into two groups by their similarity. The samples falling into the wrong cluster with the malicious label are taken as an evasive set of malicious records, with an intuition that the features of these samples were not so similar with the rest of the class so that they are not clustered with the majority of the same label samples. We applied the same logic for the benign records and finally combined the results with the new ”Evasive-PDFMal2022”. (It is available at https://www.unb.ca/cic/datasets/PDFMal-2022.html)

4.1 Evaluating the New Dataset

To prove the effectiveness of our approach for building a better dataset, certain experiments were conducted and compared it with the existing datasets. The justification on how each of these issues is handled in the new dataset is described as follows.

Duplicate Entries Issue: The first issue that we aimed to address was increasing the dataset quality by reducing the number of duplicate records. For this, all the duplicate records of both source datasets are removed before extracting our records of interest. Hence our dataset consists of 10,022 samples with no duplicate records.

Lack of Coverage Issue: To evaluate how successfully we could tackle the coverage issue and how diverse the new dataset is compared to the existing datasets, we decided to utilize two metrics, Standard Deviation, and Variance. These metrics are used to measure how disperse a set of values are from each other and are computed as described in Equation(1). Higher variance and standard deviation indicates a higher variability among data points.

$$\begin{gathered}\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N}} \\ \sigma^2 = \frac{\sum_{i=1}^{N}(x_i-\mu)^2}{N}\end{gathered} \qquad (1)$$

1. Variance And Standard Deviation Based On All Features: This evaluation is to measure the diversity of our data samples based on the whole feature set and the result comparison between benign samples of each dataset and malicious samples is depicted in the Table 2. We seek to achieve a higher number, as it indicates that the same group files are harder to detect. The outputs indicate that both the variance and standard deviation are considerably higher in our dataset than Contagio.
2. Variance And Standard Deviation Based On General Features: The purpose of this evaluation is to compare the diversity of the PDF files, solely based on their general features. We provided

the result comparison between benign samples of each dataset and the same results for malicious samples in Table 2. As demonstrated, our dataset's variety is remarkably greater than Contagio, which is a good indication that it will be much more difficult to classify our samples based on general features such as size and page number.

3. Ratio of PDF files having JavaScript and OpenAction features: Another lack of coverage indication in Contagio was that 74% of malicious samples contained one or more "JavaScript" and 'OpenAction' features, whereas only a tiny fraction of benign files had both two features. In our new dataset, this ratio is more realistic and balanced at around 49%, which shows that attack varieties are less limited to exploiting two features only.

5 EXPERIMENTS

To finalize our proposed model, we have tuned its parameters and evaluated using two different datasets: Contagio dataset and the newly created Evasive PDF dataset.

5.1 Experimental Setup

We implemented and executed all modules on a Kali Linux OS with one vCPU and 4 GB RAM on a Virtual Machine. We utilized the python PyMuPDF library and PDFid(developed by Didier Stevens) to develop our feature extraction module, along with python sklearn library for developing our classifier.

5.2 Data Repository

We used the Contagio dataset to finalize our proposed model's structure and compare our results with existing research works. Moreover, results are tested on the second dataset, labeled as "Evasive PDF Dataset", to further confirm our model's effectiveness. For the dataset split, we used 5-Fold cross-validation to carry out all of our evaluations.

5.3 Finalizing the Proposed Model

To finalize our learning detection model, we need to determine the best subset of the shortlisted base-learners and the best meta-learner out of the three proposed meta-learners mentioned in section 3. For this, each individual base-learner is tested on the baseline dataset and we reported the results in the Table 3.

Based on the results, it can be identified that the highest scores belong to Random Forest whereas Adaboost has the lowest performance. Surprisingly MLP and Linear SVM performed at the same level. To select the most suitable meta-model, we choose the one that produces the best results for our model and tried stacking multiple combinations of the base-learners. Finally, the best results were obtained by combining 'MLP', 'Linear SVM' and 'RF' as base learners, with using 'Logistic Regression' as the meta learner. The performance results by applying different meta-models are shown in the Table 6, where LR outperfomed the other classifiers.

5.4 Evaluating the Proposed Model

Here we evaluate our finalized model performance by testing it on the Contagio dataset to compare the results with previous relevant works. Moreover, we re-evaluate it on the Evasive PDF Malware Dataset, which was created in section 4.

Table 4: Base-learner results on the new dataset.

Base Learner	Acc.	Prec.	Rec.	F1-s.
RF	98.44	98.65	98.68	98.66
MLP	98.33	98.55	98.53	98.52
SVM	98.21	98.23	98.23	98.28
Adaboost	97.32	97.75	97.73	97.75

Table 5: Proposed model results on our dataset.

Accuracy	Precision	Recall	F1 score
98.69	98.88	98.87	98.77

Table 6: Proposed model results with different meta-learners.

Metamodel	Acc.	Prec.	Rec.	F1.
LR	99.89	99.84	99.88	99.86
KNN	99.86	99.81	99.82	99.81
DT	99.85	99.75	99.70	99.78

Table 7: Comparison of our work with relevant studies.

Research	Acc.	Prec.	Rec.	F1.
(Cuan et al.)	99.68	NA	NA	NA
(Fettaya et al.)	99.83	NA	NA	NA
Proposed	99.89	99.84	99.89	99.86

5.5 Comparing Proposed Model with Relevant Studies

It is generally a challenging task to compare our proposed method with different relevant works as some of the studies did not provide all the measurements. It was specifically difficult to access the exact dataset they trained and tested their model with. However, we shortlisted some papers that purely used Contagio dataset and compared their results in the Table 7. Eventhough the specified research works solely relied on Contagio dataset to evaluate their research, the evaluation method was not specified. Based on the results, it was identified that our approach achieves a higher testing accuracy than the other two researches.

5.6 Evaluating the Proposed Model using New Dataset

As the new dataset proved to be more reliable than Contagio dataset based on a set of criteria, we decided to evaluate our model on our own dataset to validate its effectiveness further. We can observe the results in the Table 5 and the comparison of the individual scores of each classifier in Table 4. From these results, it can be seen that our proposed model outperforms each of the individual scores, which verifies the robustness and validity of our design.

6 CONCLUSIONS AND FUTURE WORKS

Malware PDF files are a severe Cyber risk in today's world. Analyzing the PDF complex structure and powerful features, we can conclude that attackers can deliver malware in multiple ways. Lack of user awareness coupled with the ineffectiveness of common anti-viruses has increased this risk even more.

Several solutions have been proposed for PDF malware detection, with their strengths and weaknesses. In this research, we proposed a stacking-based learning model to detect malicious PDF files. We demonstrated our solution's effectiveness through several experimentation by extracting a set of 28 representative features and stacking three different algorithms. Furthermore, we generated a new dataset (Evasive-PDFMal2022) according to specific data quality cri-teria that lead to more reliable results and better rep-resents the real-world distribution of benign and ma-licious PDF files.

ACKNOWLEDGEMENTS

We thank the Lockheed Martin Cybersecurity Research Fund (LMCRF) to support this project.

REFERENCES

Blonce, A., Filiol, E., and Frayssignes, L. (2008). Portable document format (pdf) security analysis and malware threats. In *Presentations of Europe BlackHat 2008 Conference*.

Brandis, R. and Steller, L. (2012). Threat modelling adobe pdf. Technical report.

Carmony, C., Hu, X., Yin, H., Bhaskar, A. V., and Zhang, M. (2016). Extract me if you can: Abusing pdf parsers in malware detectors. In *NDSS*.

Corona, I., Maiorca, D., Ariu, D., and Giacinto, G. (2014). Lux0r: Detection of malicious pdf-embedded javascript code through discriminant analysis of api references. In *workshop on artificial intelligent and security workshop*, pages 47–57.

Cross, J. S. and Munson, M. A. (2011). Deep pdf parsing to extract features for detecting embedded malware. *Sandia National Labs, Albuquerque, New Mexico, Unlimited Release SAND2011-7982*.

Cuan, B., Damien, A., Delaplace, C., and Valois, M. (2018). Malware detection in pdf files using machine learning.

Cui, Y., Sun, Y., Luo, J., Huang, Y., Zhou, Y., and Li, X. (2020). Mmpd: A novel malicious pdf file detector for mobile robots. *IEEE Sensors Journal*.

Fettaya, R. and Mansour, Y. (2020). Detecting malicious pdf using cnn. *arXiv preprint arXiv:2007.12729*.

Itabashi, K. (2011). Portable document format malware. *Symantec white paper*.

Jeong, Y.-S., Woo, J., and Kang, A. R. (2019). Malware detection on byte streams of pdf files using convolutional neural networks. *Security and Communication Networks*, 2019.

Li, Y., Wang, Y., Wang, Y., Ke, L., and Tan, Y.-a. (2020). A feature-vector generative adversarial network for evading pdf malware classifiers. *Information Sciences*, 523:pp. 38–48.

Liu, D., Wang, H., and Stavrou, A. (2014). Detecting malicious javascript in pdf through document instrumentation. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 100–111. IEEE.

Maiorca, D., Ariu, D., Corona, I., and Giacinto, G. (2015). A structural and content-based approach for a precise and robust detection of malicious pdf files. In *2015 international conference on information systems security and privacy (icissp)*, pages 27–36. IEEE.

Maiorca, D., Biggio, B., and Giacinto, G. (2019). Towards adversarial malware detection: Lessons learned from pdf-based attacks. *ACM Computing Surveys (CSUR)*, 52(4):pp. 1–36.

Nissim, N., Cohen, A., Glezer, C., and Elovici, Y. (2015). Detection of malicious pdf files and directions for enhancements: A state-of-the art survey. *Computers & Security*, 48:246–266.

Stevens, D. (2011). Malicious pdf documents explained. *IEEE Security & Privacy*, 9(1):80–82.

Torres, J. and De Los Santos, S. (2018). Malicious pdf documents detection using machine learning techniques.

In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, pages 337–344.

Tzermias, Z., Sykiotakis, G., Polychronakis, M., and Markatos, E. P. (2011). Combining static and dynamic analysis for the detection of malicious documents. In *Proceedings of the Fourth European Workshop on System Security*, pages 1–6.

Zhang, J. (2018). Mlpdf: an effective machine learning based approach for pdf malware detection. arXiv preprint arXiv:1808.06991.

ZhiWei, M., Singh, M. M., and Zaaba, Z. F. (2017). Email spam detection: a method of metaclassifiers stacking. In *The 6th international conference on computing and informatics*, pages pp. 750–757.