A Feature Set of Small Size for the PDF Malware Detection

Ran Liu
Univ. of Maryland, Baltimore County
rliu2@umbc.edu

Charles Nicholas
Univ. of Maryland, Baltimore County
nicholas@umbc.edu

ABSTRACT

Machine learning (ML)-based malware detection systems are becoming increasingly important as malware threats increase and get more sophisticated. PDF files are often used as vectors for phishing attacks because they are widely regarded as trustworthy data resources, and are accessible across different platforms. Therefore, researchers have developed many different PDF malware detection methods. Performance in detecting PDF malware is greatly influenced by feature selection. In this research, we propose a small features set that don't require too much domain knowledge of the PDF file. We evaluate proposed features with six different machine learning models. We report the best accuracy of 99.75% when using Random Forest model. Our proposed feature set, which consists of just 12 features, is one of the most conciseness in the field of PDF malware detection. Despite its modest size, we obtain comparable results to state-of-the-art that employ a much larger set of features.

CCS CONCEPTS

• Security and privacy → Malware and its mitigation; **Artificial immune systems**; **• Computing methodologies → Machine learning**; **• Applied computing → Document management and text processing**;

ACM Reference Format:
Ran Liu and Charles Nicholas. 2023. A Feature Set of Small Size for the PDF Malware Detection. In *M. Gaur, E. Tsamoura, S. Sreedharan, S. Mittal, Proceedings of the Third ACM SIGKDD Workshop on Knowledge-infused Learning (KDD KiL 2023). Long Beach, California, USA, August 6, 2023. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The flexibility and portability of PDF files makes them a popular target for malware attacks. Over time, different approaches have been proposed to detect PDF malware. Machine learning and neural network based models have particularly shown promise in these detection tasks. However, the performance of the model relies on the quality of the feature set chosen[5]. Features used in malware detection are grouped into two categories: dynamic and static. Dynamic features are obtained from monitoring program execution, such as APIs called, instructions executed, or IP addresses accessed. Conversely, static features are obtained through static analysis. Both categories have some limitations. Dynamic features need to be executed in a sandbox environment, where some sophisticated malware can detect the sandbox environment and consequently alter their behaviors. Static features, on the other hand, can be obfuscated by attackers using evasion techniques, making the detection challenging. This raises concerns that some commonly used features have been thoroughly investigated by attackers. If attackers have exploited these features to perform a successful evasive attack, PDF malware detection systems built on the same or similar features set might become vulnerable. This highlights the importance of the usage of PDF-specific features, which may reduce the attack surface. Earlier research, including PDFRate, have employed some PDF-specific features such as the number and the occurrence of a specific PDF objects for model training, which obtained promising accuracy in PDF malware detection[9][7]. Nevertheless, most of these features requires a large amount of domain knowledge to extract. Moreover, their feature sets are large and complex, which may potentially lead to over-fitting. Consequently, it's desirable to have a simple and small PDF-specific features set that may achieve detection accuracy comparable to more complex features.

In this paper, we limit the scope of PDF-specific features to those that are unique to PDF files, hence excluding most dynamic features such as system call sequences, API call sequences, and some static features such as binary code. Furthermore, we exclude features that need extensive domain knowledge for PDF files, which means that most keyword-based features, including JavaScript code in PDF files, are not used in our research.

PDF files can be viewed as a set of interconnected objects. Some work, such as Hidost's, extracts the tree structure of the PDF and uses the binary counts for these paths as features[15]. Our previous work showed that such tree structures contain sequential relationships and can be used to train the Time Series Model for PDF malware detection. In this paper, we propose a novel set of graph features to accurately detect PDF malware. We investigated multiple types of graph tree features by parsing a PDF file into tree representative features. For specific feature types, our research demonstrated statistical differences between benign and malicious PDFs. Using the proposed feature set to train a machine learning model, we show empirically that the model can successfully detect 99.75% of PDF malware samples with only 12 features. The primary contributions of our study are the introduction of one of the smallest PDF specific feature sets. We have conducted a thorough investigation and performance analysis of the ML - models based on these proposed features. Furthermore, we benchmarked our results against state-of-the-arts, indicating that our feature set is promising.

1.1 PDF file Structure

A PDF file is structured using interconnected modules known as objects, which are made up of four parts: a header, a body, a cross-reference table, and a trailer, as shown in Figure 1.

- Header: The header contains information about the PDF version and is marked with the '%' symbol.
- Body: The body, as the primary section of the PDF file, consists of objects that define all the operations performed by the file. These objects, which include both indirect and direct objects, characterize the functionality of each object using

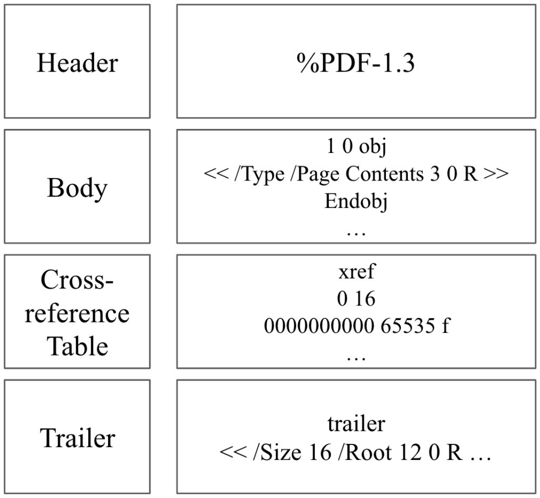
keywords marked with ’/’. For example, ’/Length’ and ’/Filter’ are such keywords. Indirect objects, which start with a numeric identifier like ’4 0 R’, contain information in a directory and can be referenced by other objects. For example, an object starting with ’1 0 R’ can be referred to by other objects using ’1’, the sequence number. This structure allows for the interconnection of objects. The generation number, usually set to 0, is represented by the second digit, although it can be other number in some cases. Objects are usually end with the ’endobj’ marker. A specific type of object, known as a stream, starts with the keyword ’stream’ and ends with ’endstream’ and ’endobj’. The content of the stream object, which includes elements like images and texts, is encoded using filters.

- Cross-reference table: This table contains the location references for each object. A PDF parser uses this table to find the object reference in the memory for parsing. The cross-reference table, marked by ’Xref’ followed by numbers, indicates the total number of objects in the references with its last number. For instance, ’0 16’ indicates a total of 16 objects in the cross-reference table.
- Trailer: trailer contains information about the file such as the number of objects using keyword ’/Size’. It also contains a reference to the root object using keyword /Root and metadata using keyword /Info. The file structure organizes the logical access order for the PDF file. When a PDF reader application accesses a PDF file, it first locates the trailer to find the root object. Then the parser uses a cross-reference table to parse each indirect object to decompress all data. In this way, the content of the PDF is made visible to the user. When a modification happens in the PDF file, like inserting a page into the PDF, a new body, trailer, and cross-reference table will be appended to the original file accordingly, and a new version number will be generated. However, because the cross-reference table sets a strict boundary for each object, removing objects from the previous version can cause errors. Thus, an attacker is more likely to add features instead of removing features.

Each object is labeled with a number, allowing it to be referenced by other objects. Object information is stored in the cross-reference table, while the trailer indicates the root object’s number and the cross-reference table’s location. By querying the cross-reference table, catalog objects can be found. The catalog object serves as the entire document’s root object, containing the PDF document’s outline and the page group object’s reference.

1.2 Related Work

In the field of machine learning-based PDF malware detection, two primary types of features are commonly used - static and dynamic features. The dynamic features are obtained by running the PDF in a controlled environment, which allows for the collection of PDF running behaviors such as sequences of system calls and API calls[11][6]. However, most dynamic features are typically not distinctive to PDF files, and the building of a robust sandbox environment increases the complexity of the detection process. Despite the promising results in malware detection tasks using dynamic features, our work primarily focuses on the use of static features. Previously used dynamic and static features require significant amounts of domain knowledge for feature extraction. In contrast, our goal is to employ features that demand minimal domain knowledge and are unique to PDF files.



Header	%PDF-1.3
Body	1 0 obj << /Type /Page Contents 3 0 R >> Endobj …
Cross-reference Table	xref 0 16 0000000000 65535 f …
Trailer	trailer << /Size 16 /Root 12 0 R …

Figure 1: PDF Structure

There are three categories of static features. The first type of static features are obtained from a keyword-based analysis, which involves searching for predefined keywords such as ’/Javascript’, ’/OpenAction’, ’/GoTo’, ’/URI’, and ’/RichMedia’. These keywords are often associated with malicious code injection. Features can include the number of keywords or simply their presence. The malicious payloads are usually inserted into objects associated with such keywords, making them useful features for PDF malware detection. The second type of static features are obtained through a tree structure-based analysis. This method constructs the object tree representation to capture the connections between items. The tree structure can provide insights into the hierarchical connections between the objects of the PDF file, which may reveal malware-related behaviors. Lastly, static features can be obtained using code-based analysis, which focuses on malicious strings and functions in Javascript code. As PDF malware often manipulates Javascript code to execute malicious activities, the presence of specific code strings or functions may indicate the presence of malware. PDF malware detectors may employ one or more of the features described above. One such example is Hidost[15], a system that uses the Poppler PDF parser[2] to extract tree structural paths of objects in a PDF file, which are then used as features in the classification process. Hidost is implemented with two different models: Support Vector Machine (SVM) and Random Forest (RF)[14][15]. SVM is a supervised learning model that creates an optimal hyperplane to separate different labels. RF is a meta-estimator that integrates different decision trees to improve classification accuracy. The researchers trained their model using a dataset of 10,000 randomly selected files, maintaining a malicious-to-benign ratio of 1:1. The complete PDF dataset comprised 407,037 benign and 32,567 malicious files. The Hidost system has 99.8% accuracy and less than 0.06% false positive rate for both models.

The PDFrate classifier is implemented using an RF algorithm with 99% accuracy and 0.2% false positive rate over the Contagio malware dataset[9]. PDFrate uses the metadata, which includes the names of the files' authors, the size of the file, its location, and the amount of certain keywords, and the content of the PDF files as features. The authors manually define the feature set, which has 202 features in all, including counts for different keywords and specific fields in the PDF. Examples include the number of characters in the author field, the quantity of "endobj" keywords, the total number of pixels in all the photos, the quantity of JavaScript markers, etc. The Mimicus implementation of PDFrate, claiming to get a close approximation, only makes use of 135 of these features. The two versions of PDFrate, PDFrate-v1 and PDFrate-v2, each use a different machine learning model[18][12]. The classifiers in PDFrate-v2 use mutual agreement to implement an ensemble technique. The term "uncertain" is introduced into the classifier voting, where rates of 25–50% are regarded as benign uncertainty and rates of 50–75% as malicious uncertainty.

PjScan is a tool that concentrates on examining JavaScript code[4]. It uses Poppler[2] as a parser to extract tokens from JavaScript code, and a one-class SVM as a classifier. PjScan achieved 85% detection accuracy. Malware Slayer is available in two variants[8][7]. The original Slayer extracts keyword features from PDF files using a pattern recognition method and labels samples using a random forest algorithm. Slayer NEO uses the PeePDF[16] and Origami[13] parsers to extract structural data as features and the AdaBoost algorithm for classification.

Maryam et al. proposed a PDF malware detection system based on stacking learning[3]. Their approach is based on the idea that combining different classifiers could produce improved accuracy as each classifier operates based on unique data assumptions. Their feature set included ten general features, such as PDF size and title character count, as well as structural features such as the amount of keywords and objects. These extracted features were initially fed into a base layer consisting of SVM, Random Forest, MLP, and AdaBoost. The prediction outputs from this layer were subsequently fed into a meta-layer featuring Logistic Regression, K-Nearest Neighbors, and Decision Trees. Their reported metrics on the hybird dataset including Contagio dataset were impressive: an accuracy of 99.98%, precision of 99.84%, recall of 99.89%, and an F1 score of 99.86%.

2 STATISTICAL ANALYSIS OF A FEATURE SET

We now introduce our proposed feature set. We use the Contagio dataset, which includes 9,000 benign and 10,982 malicious PDF samples, to extract features[9]. This dataset was chosen due to its accessibility and its large number of labeled samples. We used the pdfrw library to extract tree structure paths for each file, and while some samples were corrupted, we were able to successfully extract path objects from 7,396 benign PDF files and 10,814 malicious PDF files[17]. Our feature selection strategy is to minimize required domain knowledge during feature extraction. Consequently, despite the promising results achieved by keyword-based features in other research, we chose not to use them. We have already noted that a PDF can be represented as a tree structure of objects, prompting our investigation into graph features. Our selection of features was facilitated by comparing mean values, standard deviations with 95% CI and Quantiles. This led us to select the following features:

- Distribution of children per node: the average (avg children), median (median children) and variance of children per node (var children).
- Number of leaves in the tree (num leaves).
- Number of nodes (num nodes).
- The depth of the tree (depth).
- Average degree (avg degree).
- Degree assortativity coefficient (degree assortativity).[1]
- The average shortest path length (avg shortest path).
- How nodes in a graph tend to cluster together (avg clustering coefficient).
- Graph density (density).

We applied statistical analysis to investigate the proposed features of benign and malicious PDF files. The key statistical metrics in our investigation were: the 75% quartile, median 50%, 25% quartile, 95% CI mean and 95% CI standard deviation.

Table 1: Quantiles for the Proposed Feature Set for Benign PDFs

Benign PDF		Quantiles	
	75%	50%	25%
avg children	1.9416	1.5218	1.44
avg clustering coefficient	0.0640	0.0185	0.097
avg degree	3	3	2
avg shortest path	1.0667	0.7118	0.5421
degree assortativity	-0.3781	-0.4298	-0.5569
density	0.0215	0.0083	0.0068
depth	4	4	4
median children	0	0	0
num edges	459	309	189
num leaves	188	158	71
num nodes	260	199	135.25
var children	158.2680	94.4440	51.1347

Note that Table 1, Table 3, Table 2 and Table 4 show a significant difference in the proposed feature set between benign and malicious PDFs. The statistical difference between benign and malicious PDFs is further visualized in the box plots (Figure 2 and Figure 3), providing clear graphical representations of these differences. We excluded the median degree, depth, and median children from the box plots since they are discrete values.

3 EXPERIMENTAL RESULTS

3.1 PDF Malware Detection Results

Having shown a significant difference in the proposed feature set between benign and malicious PDFs, we move on to demonstrate the feature set's utility in experimental evaluation. we use the Contagio dataset and the pdfrw library to extract tree structure paths

[1] Degree assortativity refers to the tendency for nodes of high or low degree to be connected to other nodes of high or low degree, respectively.

Table 2: Quantiles for the Proposed Feature Set for Malicious PDFs

Malicious PDF	Quantiles		
	75%	50%	25%
avg children	1.7619	1.5758	1.5714
avg clustering coefficient	0.0698	0.0413	0.0374
avg degree	3	3	3
avg shortest path	0.6	0.5595	0.4542
degree assortativity	-0.2821	-0.3126	-0.3405
density	0.0881	0.0786	0.0686
depth	4	4	4
median children	1	1	1
num edges	40	37	33
num leaves	10	10	10
num nodes	27	21	21
var children	4.1814	4.0272	3.6734

Table 3: 95% CI Mean Comparison for the Proposed Feature Set

	Benign	Malicious
	95% CI Mean	95% CI Mean
avg children	1.8601 - 1.9044	1.6359 - 1.6469
avg clustering coefficient	0.0372 - 0.0392	0.0501 - 0.0512
avg degree	3.1385 - 3.2299	2.8545 - 2.8762
avg shortest path	0.8079 - 0.8279	0.5430 - 0.5498
degree assortativity	-0.4589 - -0.4541	-0.3071 - -0.3033
density	0.0205 - 0.0213	0.0764 - 0.0772
depth	4.2561 - 4.2831	3.9594 - 3.9692
median children	0.1506 - 0.1942	0.9405 - 0.9612
num edges	441.5762 - 471.6670	43.8435 - 45.6364
num leaves	152.1234 - 157.9772	12.2298 - 12.7902
num nodes	201.8999 - 209.8423	25.4634 - 26.2143
var children	114.7908 - 122.0534	4.8081 - 5.34606

for each file, and (as mentioned above) we were able to successfully extract path objects from the 7,396 benign PDF files and 10,814 malicious PDF files[17]. We use scikit-learn to evaluate the model's performance[10]. The confusion matrix we used is shown as Table 5

We apply 5-fold cross validation to 6 machine learning classifiers trained with the proposed 12 graph features. To show that our results are statistically significant, we report 95% Confidence Interval (95% CI) of the Precision, Recall in Table 6 and 95% CI F1 score in Table 7. The recall value indicates the ratio of samples have been correctly classified. We report the best recall value for the malicious class with Random Forest, indicating that 99.73% to 99.77% of the malicious samples are correctly detected, while the benign class's recall value is 0.9987 - 0.9991, which means that 99.87% to 99.91% of benign samples are correctly classified.

We report the Accuracy, True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR) and True Negative Rate (TNR) in Table 8. The result indicates our proposed features have good overall performance. We report the best accuracy of 0.9975 when using Random Forest model.

Table 4: 95% CI Standard Deviation Comparison for the Proposed Feature Set

	Benign	Malicious
	95% CI SD	95% CI SD
avg children	0.9561 - 0.9874	0.2881 - 0.2959
avg clustering coefficient	0.0427 - 0.0441	0.0274 - 0.0282
avg degree	1.9738 - 2.0385	0.5662 - 0.5815
avg shortest path	0.4358 - 0.4500	0.1759 - 0.1806
degree assortativity	0.1097 - 0.1133	0.1005 - 0.1032
density	0.0230 - 0.0238	0.02047 - 0.0210
depth	0.5867 - 0.6059	0.2579 - 0.2648
median children	0.9462 - 0.9772	0.5416 - 0.5563
num edges	649.6877 - 670.9714	46.9322 - 48.2001
num leaves	126.3856 - 130.5260	14.6701 - 15.0664
num nodes	171.4772 - 177.0948	19.6561 - 20.1871
var children	156.8088 - 161.9459	14.0821 - 14.4626

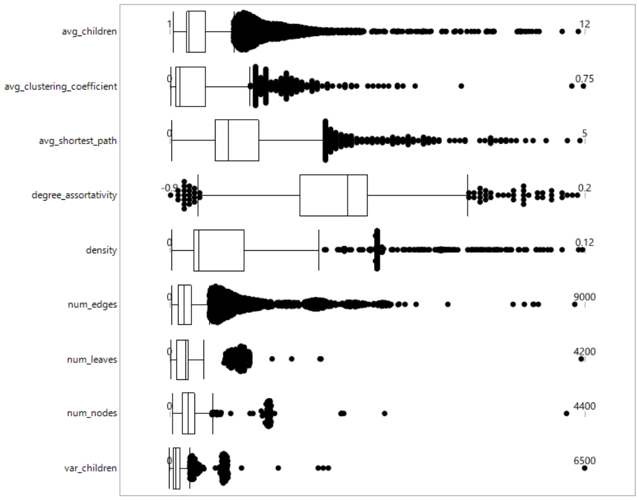


Figure 2: Box plot of the benign PDFs. Features are on the y-axis.

Table 5: Confusion Matrix.

	Detected as Benign	Detected as Malicious
Benign	True Positives (TP)	False Negatives (FN)
Malicious	False Positives (FP)	True Negatives (TN)

3.2 Comparison with Other Works

For model performance comparison, we select models with reported results in the literature for direct comparison.

We report the best result obtained by applying the Random Forest model. Table 9 presents the comparison of the evaluation metrics

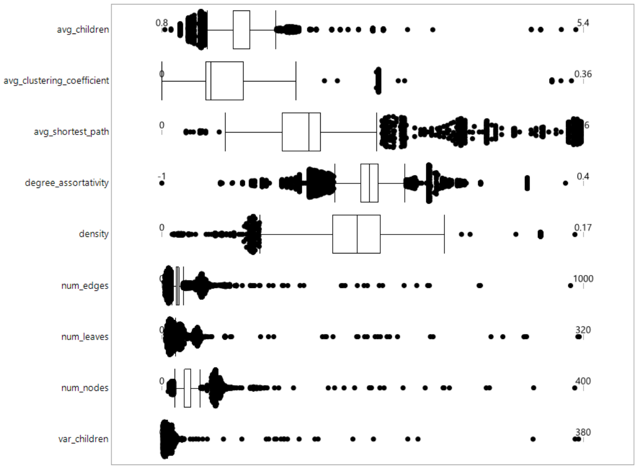


Figure 3: Box plot of the malicious PDFs. Features are on the y-axis.

Table 6: 95% CI Precision and Recall for each label.

Classifier	Label	Precision	Recall
XGBoost	Malicious	0.9939 - 0.9977	0.9958 - 0.9966
	Benign	0.9982 - 0.9987	0.9973 - 0.9991
Naive Bayes	Malicious	0.8769 - 0.9564	0.8000 - 0.9750
	Benign	0.8698 - 0.9683	0.7850 - 0.9824
Multi-layer Perceptron	Malicious	0.9871 - 0.9911	0.9811 - 0.9940
	Benign	0.9946 - 0.9963	0.9946 - 0.9963
Decision Tree (J48)	Malicious	0.9963 - 0.9973	0.9946 - 0.9982
	Benign	0.9972 - 0.9980	0.9959 - 0.9986
Random Forest	Malicious	0.9966 - 0.9982	0.9973 - 0.9977
	Benign	0.9987 - 0.9991	0.9987 - 0.9991
Simple Logistic	Malicious	0.9739 - 0.9879	0.9820 - 0.9824
	Benign	0.9878 - 0.9885	0.9831 - 0.9917

Table 7: 95% CI F1 Score for each label.

Classifier	Label	F1 Score
XGBoost	Malicious	0.9953 - 0.9968
	Benign	0.9980 - 0.9986
Naive Bayes	Malicious	0.8712 - 0.9234
	Benign	0.8671 - 0.9227
Multi-layer Perceptron	Malicious	0.9861 - 0.9906
	Benign	0.9946 - 0.9963
Decision Tree (J48)	Malicious	0.9959 - 0.9972
	Benign	0.9970 - 0.9979
Random Forest	Malicious	0.9970 - 0.9979
	Benign	0.9987 - 0.9991
Simple Logistic	Malicious	0.9781 - 0.9849
	Benign	0.9854 - 0.9901

Table 8: Accuracy, True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR) and True Negative Rate (TNR) with the Proposed Features Set.

Classifier	Accuracy	TPR	FPR	FNR	TNR
XGBoost	0.9964	0.9980	0.0042	0.0020	0.9958
Naive Bayes	0.9006	0.7850	0.0268	0.2150	0.9732
Multi-layer Perceptron	0.9926	0.9919	0.0088	0.0081	0.9912
Decision Tree	0.9967	0.9959	0.0019	0.0041	0.9981
Random Forest	0.9975	0.9980	0.0023	0.0020	0.9977
Simple Logistic	0.9822	0.9817	0.0180	0.0183	0.9820

Table 9: Comparison with other models.

Model	Precision	Recall	F1-Score	Accuracy
Model[18]	0.9970	0.9970	0.9970	0.9965
Model[1]	0.9880	0.9890	0.9885	0.9884
Model[3]	0.9888	0.9887	0.9877	0.9869
Our work	0.9973	0.9974	0.9974	0.9975

to other work. The results show that our work beats other work while being significantly smaller than the feature set they use.

The main weaknesses of the feature set we propose is that it is vulnerable to some evasive attacks. Detectors that employ this feature set can be compromised through the insertion, deletion, or alteration of a subtree. In order to enhance the robustness of the detector, one potential strategy is to enrich the feature set and diversify the feature types employed. We observe that the parser did not successfully parse all objects for some malware specimens. The effectiveness of the our approach is influenced by the quality of parsed PDF objects.

4 CONCLUSION

In this work, we introduced a new features set for PDF malware detection that based on PDF tree structure. Our work aimed to address the need of finding a small features set without needing too much domain knowledge of the PDF file. Our work might serve as a baseline for the future investigation. We do not expect our work can replace the current used static and dynamic features now or in the future, but our work might inspire researchers with an alternative way to build a malware detection model. In the future task, we plan to explore the other features to improve the overall performance and enhance the robustness.

REFERENCES

[1] Qasem Abu Al-Haija, Ammar Odeh, and Hazem Qattous. 2022. PDF Malware Detection Based on Optimizable Decision Trees. *Electronics* 11, 19 (9 2022), 3142. https://doi.org/10.3390/electronics11193142

[2] FreeDesktop.org. 2018. Poppler.

[3] Maryam Issakhani, Princy Victor, Ali Tekeoglu, and Arash Lashkari. 2022. PDF Malware Detection based on Stacking Learning. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 562–570. https://doi.org/10.5220/0010908400003120

[4] Pavel Laskov and Nedim Šrndić. 2011. Static detection of malicious JavaScript-bearing PDF documents. In *Proceedings of the 27th Annual Computer Security*

Applications Conference. ACM, New York, NY, USA, 373–382. https://doi.org/10.1145/2076732.2076785

[5] Ran Liu, Maksim Eren, and Charles Nicholas. 2023. Can Feature Engineering Help Quantum Machine Learning for Malware Detection? (5 2023).

[6] Ran Liu and Charles Nicholas. 2023. *IMCDCF: An Incremental Malware Detection Approach Using Hidden Markov Models*. Technical Report. MALWARE TECHNICAL EXCHANGE MEETING (MTEM) 2021.

[7] Davide Maiorca, Davide Ariu, Igino Corona, and Giorgio Giacinto. 2015. *A structural and content-based approach for a precise and robust detection of malicious PDF files; A structural and content-based approach for a precise and robust detection of malicious PDF files*. Technical Report. http://www.mozilla.org/rhino

[8] Davide Maiorca, Giorgio Giacinto, and Igino Corona. 2012. A Pattern Recognition System for Malicious PDF Files Detection. 510–524. https://doi.org/10.1007/978-3-642-31537-4{_}40

[9] Mila Parkour. [n. d.]. Malicious Documents Archive for Signature Testing and Research - Contagio Malware Dump.

[10] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2012. Scikit-learn: Machine Learning in Python. (1 2012).

[11] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. [n. d.]. *Automatic Analysis of Malware Behavior using Machine Learning*. Technical Report. http://www.iospress.nl,

[12] Charles Smutz and Angelos Stavrou. 2016. When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors. In *Proceedings 2016 Network and Distributed System Security Symposium*. Internet Society, Reston, VA. https://doi.org/10.14722/ndss.2016.23078

[13] Sogeti ESEC Lab. 2015. Origami.

[14] Nedim Šrndic and Pavel Laskov. 2013. Detection of Malicious PDF Files Based on Hierarchical Document Structure. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium*. Citeseer, 1–16.

[15] Nedim Šrndić and Pavel Laskov. 2016. Hidost: a static machine-learning-based detector of malicious files. *EURASIP Journal on Information Security* 2016, 1 (12 2016), 22. https://doi.org/10.1186/s13635-016-0045-0

[16] Trevor Tonn and Kiran Bandla. 2013. PhoneyPDF.

[17] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers. In *Proceedings 2016 Network and Distributed System Security Symposium*. Internet Society, Reston, VA. https://doi.org/10.14722/ndss.2016.23115

[18] Suleiman Y. Yerima, Abul Bashar, and Ghazanfar Latif. 2022. Malicious PDF detection Based on Machine Learning with Enhanced Feature Set. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 486–491. https://doi.org/10.1109/CICN56167.2022.10008374