Hindawi
Security and Communication Networks
Volume 2022, Article ID 7611741, 10 pages
https://doi.org/10.1155/2022/7611741

*Research Article*

# Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files

**Sultan S. Alshamrani**

*Department of Information Technology, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*

Correspondence should be addressed to Sultan S. Alshamrani; susamash@tu.edu.sa

Received 5 July 2022; Revised 26 August 2022; Accepted 7 September 2022; Published 21 September 2022

Academic Editor: Muhammad Faisal Amjad



Modern day antivirus software, which is available commercially, is incapable of providing the protection from the malicious portable document format (PDF) files and thus considered as a threat to system security. In order to mitigate the same to some extent, a new PDF malware classification system based on machine learning (ML) is introduced in this paper. The novelty of this system is that it will be inspecting the given PDF file both statistically and dynamically, which in turn will increase the accuracy of finding the correct nature of the document. This method is nonsignature-based and hence can possibly distinguish obscure and zero-day malware. The experiment is carried out for this system by deploying five different classifier algorithms to find out the best fit for the system. The best fit approach is analyzed by calculating the true positive rate (TPR), precision, false positive rate (FPR), false negative rate (FNR), and F1-score for each of these classifier algorithms. Comparison of this work is carried out with previously existing PDF classification systems. A malicious attack on to the proposed system is also implemented, which will in turn obfuscate the malicious code inside the PDF file by making it hidden during the parsing phase by the PDF parser. It has been inferred that the proposed approach achieved F1-measure of 0.986 by using the random forest (RF) classifier in comparison to state-of-the-art where F1-measure was 0.978. Thus, our approach is quite effective in the identification of the malwares when embedded in the PDF file in comparison to the existing systems.

## 1. Introduction

In the current generation of the digital world, most of the activities are centric towards the usage of the Internet, and thus, it becomes more important to safeguard our applications, data, and information in the presence of various attackers who are always trying to devise new malicious codes and attacks to compromise the resources. Hence, malware analysis becomes one of the prime concerns today as various malwares are generated by attackers and even their properties are changing very rapidly day by day. Nowadays, malware is not the same one that was there before as they change its signatures with time and thus difficult to trace. So, identification and classification of the latest malware is one of the most sought-after areas of research. There are majorly two ways for malware identification: one is a signature-based detection technique and the another one is behavior-based. The signature-based technique is quick and efficient only for identifying known malware and the behavior-based technique is able to identify unknown and complex malware to some extent using machine intelligence and other approaches, but the behavior-based technique is a complex one. None of the methods can detect all kinds of malware, especially when the count of malware is increasing day by day. In the signature-based approach, unique signature is created by using the attributes of the underlying object. The presence of digital signature is efficiently detected by scanning the object by the algorithm. In the behavioral-based approach, intended actions of objects are evaluated before such actions are carried

out. This approach analyses the potential behavior of any actions carried by objects before the actual execution of behavior. The older malware was easy to detect as they were able to hide their features, but the malware uses different techniques like obfuscation [1–4] to hide their identity for a longer span and even they can bypass the firewall and other security checks present in the network or system. Also, multiple types of malwares are used to launch the attack, so the effects are more devastating.

There are majorly two ways for analyzing the malware: static and dynamic. In the static approach, malware is inspected without running the code it embeds, whereas in the dynamic approach, it is inspected by running its code [5, 6]. Thus, malware identification is one of the foremost steps in the malware analysis process. The static analysis does not require executing any malware samples and is very simple. There is no need to cover each phase of the process while performing static malware analysis. Dynamic analysis involves the detailed analysis for malware detection. A complete behavior of actions is thoroughly analyzed, while the process remains under execution. This analysis requires detailed monitoring for processes. Classification of malware is also important as identification is. The various categories of malware are viruses, trojan horse, worms, rootkits, ransomware, and key logger. There are various ways by which the classification can be carried out for malware such as feature-based techniques [7, 8] and image classification where the binary values are transformed to image. So, for better classification, more information will be fruitful [9, 10]. For better classification, good classifiers are required to be developed for the better and accurate classification of malware using some latest machine intelligence techniques.

One of the most widely used document formats is PDF. Despite the general public's ignorance, it quickly emerged as a critical attack vector for computers. Hackers may take over a victim's computer using dozens of flaws found in adobe reader. In addition, antivirus software developers have a difficult time protecting PDF files from assaults because of the file's complex internal structure and the vast variety of obfuscation techniques already in use [11]. Most of us send attachments in the PDF format since it is recognized for its mobility and small weight. However, we have no idea what kinds of assaults these files may be used for or propagated to. The three primary forms of PDF malware are vulnerabilities, phishing, and exploitation of PDF features. Vulnerability in the PDF reader's API is exploited by exploit kits, allowing the attacker to run an arbitrary code on the compromised machine. In most cases, JavaScript code is included in the file to do this. However, in phishing assaults, an unsuspecting file is used to trick the user into clicking on an infected link. These campaigns have just lately been uncovered, and they are significantly more difficult to recognize. A malicious program may be downloaded or a website's login credentials may be stolen by any of these assaults.

The static analysis makes use of some techniques for identification such as file format inspection, string extraction, fingerprinting, which primarily used hash code values, antivirus scanning, and disassembly where the machine code is changed to assembly language [12]. Static methods are the time taking ones and also more based on behavior analysis of malware. But in the dynamic approach, the behavior is monitored, while the file is executing for any malware identification. So, it has more leverage to identify the malware. Similarly, detection of malware is primarily done through two ways: one is signature-based where the predefined signatures, if there are any, of malware are used for detection and the other one is a heuristic-based approach where multiple factors are used that contrasts the malicious behavior. One of the challenges in the signature-based approach is that attackers develop the malware by changing their signatures so many times that they are hard to trace. Hence, the heuristic approach is more favorable owing to its capacity to identify polymorphic and some latest attacks.

Using heuristics, sequences of code, and string comparison, signature-based algorithms may determine if a PDF is benign or malicious. However, this has not been demonstrated to be effective against stealth assaults of the present day. If one wants to find the hidden malicious behaviors of a particular file, dynamic approaches are more successful since they run the file in a supported environment and analyze the process it goes through as well as the API calls it makes and build a thorough record of its activities. One may learn a lot about a file's characteristics by looking at the execution log. Because the attributes that are employed to identify malware vary depending on the approach, this is true for all methods of malware detection. Like in a signature-based byte sequence, all methods of malware detection,such as Dynamic link libraries (DLL), behavior-based API and system calls, heuristic used operation code, context-free grammars, and some new techniques such as mobile-based used android permissions and system calls are used [13].

The novelty of the proposed approach given in this paper is signature-less driven criteria. The suggested model will evaluate the API calls processes inside the PDF file and will thus look for the activities that will be performed throughout the file's processing. The detection may be dependent on the system calls and JavaScript files inside the PDF file that have been evaluated. The data mining technique is used in this system to collect information from API requests. It is possible to categorise a particular file as being either "Ordinary (O)" or a "Potentially Malicious (PM)" based on the retrieved characteristics and statistics. Finally, these results are sent via the classification block which maps the gathered information with the algorithm's findings and classifies the file as "correct" or a "malicious" file.

The main highlights of this paper are as follows:

(1) It provides a novel ML-based malware identification approach for the PDF files

(2) It provides the training and testing implementation of the proposed model under the various ML approaches

(3) It also highlights the efficiency of the proposed approach under the simulated malicious attack

The paper is divided into the sections as per their relevance. The work already done in the context of malware identification using the ML and PDF file based has been

elaborated in Section 2. The proposed techniques have been defined in Section 3, which is followed by the dataset details under Section 4. Results and the inference drawn have been defined in Section 5. Also, the comparative analysis with existing models has been done in Section 6. Conclusion is highlighted in Section 7.

## 2. Related Work

In this part of the paper, the researchers' main efforts are in detecting and classifying malware using machine learning (ML) and other approaches. Also, the work done pertaining to the file types that are utilized for the malware identification has also been expressed.

*2.1. Malware Identification Related Work.* Malware analysis focuses on finding the operation modus of malware and how it affects the programs and systems. Historically, signature-based identification approaches were widely used. This technique works against known malware quickly and effectively but does not work with respect to the zero-day malware properly [14, 15]. A malware identification framework oriented on the genetic algorithm (GA) and signature generators [16] was proposed by authors. While the authors claim that this methodology may identify unknown malware, the paper does not include significant information for the proposed framework, such as testing results, the amount of malware studied, and a comparison to other current studies. Fukushima et al. have defined [17] a behavior-based detection method. New and encrypted malware may be detected using the proposed approach on Windows OS. In [18], a supervised ML method is suggested. The model utilized an SVM kernel basis that weighs the frequency of each library call for the detection of Mac OS X malware.

Recently, with the advent of intelligence techniques, ML has also become one prominent way in malware analysis. Deep learning is an ML subcomponent, which is a heritage from artificial neural networks (ANNs). It is a novel method and is widely utilized for the analysis of images and autonomous cars, but is not enough for virus detection. Although it quite effectively and significantly decreases the area for features, it does not prevent assaults from evasion. Shabtai et al. [19] proposed taxonomy for malware identification by reporting certain sorts of functions and selecting features in the literature, using ML methods. They focus largely on the selection of features. In [20], author has provided a detailed survey of ML for malware analysis. They have mentioned the challenges of datasets and the ways to overcome them. Image transformation with ML is used for malware identification by the author where the convolution neural network (CNN) is utilized [21]. Similarly, the work in the direction of tools usage and framework representation for the malware analysis has been carried out by the researchers recently [22–25].

*2.2. File-Based Malware Identification Related Work.* In [26], authors examined PDF design and JavaScript information included in PDFs from top to bottom. With regard to design and metadata, they created an extensive set of capabilities, such as the count of bytes per second, the encoding scheme, object names, catchphrases, and comprehensible strings in JavaScript. Also, when the characteristics vary, it is difficult to create antagonistic models since little changes are strong for AI calculations. They built up a classification model utilizing discovery type models keeping structures and data features to limit the danger of ill-disposed assaults. To approve the proposed model, they fabricated an adversarial attack. In [27], authors have presented an outline of the PDF; also, the current assaults are used to be carried out on PDF malware through solid assault models gathered in nature. They depicted how to play out a measurable examination of a PDF record to discover the proof of implanted malware utilizing programming strategies. They examined some of the new PDF malware detection apparatuses dependent on AI that can uphold computerized scientific examinations; recognizing dubious documents before a more profound, a more definite statistical evaluation is released. They examined the PDF constraints and other open issues, particularly regarding the misuse of their weaknesses to possibly misdirect resulting measurable investigations. At last, they recommended tips for improving the exhibition of such frameworks enduring an onslaught and sketch promising analysis. In [28], authors have focused on the malware implanted in PDF files as a delegate instance of modern-day cyber-attacks. They started by giving a scientific classification of the various methodologies used to produce PDF malware. To combat PDF malware classifiers based on learning, they have utilized an adversarial AI structure that has been shown effective. For example, this method enables us to identify existing flaws in learning-oriented PDF malware locators and to identify fresh attacks that may jeopardize such frameworks, along with the possibility of protective measures. In [29], authors have planned and executed a novel framework called AIMED, utilizing hereditary calculations to sidestep malware classifiers. Their tests proved that an opportunity to accomplish ill-disposed malware tests can be diminished up to half, contrasted with exemplary arbitrary approaches. Also, they carried out AIMED to create ill-disposed models utilizing individual malware scanners as target and tried the adversarial documents against additional classifiers from both examination and industry. The created models accomplished up to 82% of cross-avoidance rates among the classifiers.

In [30], authors have exhibited how the most pessimistic scenario conduct of a malware classifier regarding explicit vigor properties can be evaluated. Besides, they found that preparation of classifiers that fulfill officially checked vigor properties can build the avoidance cost of unbounded assailants by dispensing with straightforward assaults avoidances. They proposed another distance metric that works on the PDF tree structure and determined two classes of strength properties including subtree inclusions and erasures. They used the best in class irrefutably vigorous for preparing a strategy to construct strong PDF malware classifiers. A PDF malware classifier, PDFrate, is used by the authors later in [31] to evaluate their methods. Using data

from a real network, they demonstrate that high quality classifier arrangements can make the majority of predictions. It is clear that the classifier cannot reliably predict the outcomes of most avoidance efforts, including nine focusing on imitation scenarios from two current projects. Over 100,000 PDF files as well as 100,000 Android apps are part of their evaluation. In [32], authors presented "Hidost," the primary static AI based malware discovery framework intended to work on various file extensions. Broadening a formerly distributed and profoundly viable strategy, it consolidates the coherent design of documents with their substance for better identification precision. On account to its specific plan and general list of capabilities, it is extended to differentiate organizations whose coherent design is coordinated as a chain of command.

In [33], authors presented a novel AI framework for automating the discovery of malicious PDF files. Both of the structure and data in the PDF are extracted, and a sophisticated parsing mechanism is included. As a result, a broad range of malware may be distinguished, comprising parsing-based and non-JavaScript malware. Additionally, with a cautious decision of the learning calculation, their methodology has given an altogether higher exactness contrasted with other static examination methods, particularly within the sight of ill-disposed malware control.

To identify JavaScript-induced malware, the authors of [34] employed AI algorithms to get a sample of API references that depict the malicious code. An important application area was examined in this investigation, namely, the placement of the malicious JavaScript code in PDF files. Although their training data contained instances of malware, they demonstrated that their strategy has been able to identify new malware even when it was introduced into an existing system that had not previously been exposed to such malicious code. In [35], authors built up a framework that utilizes various feature selection and AI-induced techniques to set up the attributes of typical JavaScript code.

## 3. The Proposed Approach

PDF documents include a header, body, cross-reference table (CRT), and a trailer. Components in the body include information about the document itself, while the header provides the information about the document's current version. Tables used to connect to objects are included in the CRT. The root object and the table locations of the objects in the body area are included in the trailer part.

The proposed ML-based malware categorization technique is explained here. For the most part, this system is designed to scan the PDF file being inspected, sort out its internal code, and determine if it is good or dangerous. The hacker's attempts to obfuscate file headers have also been found to be blocked by the mechanism in place. This technique does not identify the malware family contained inside a particular file, but it does accurately classify the file's type [36]. System's categorization procedure of the proposed work is shown in Figure 1. Even if this is a high-level system design, it provides a better idea of how the classifier is

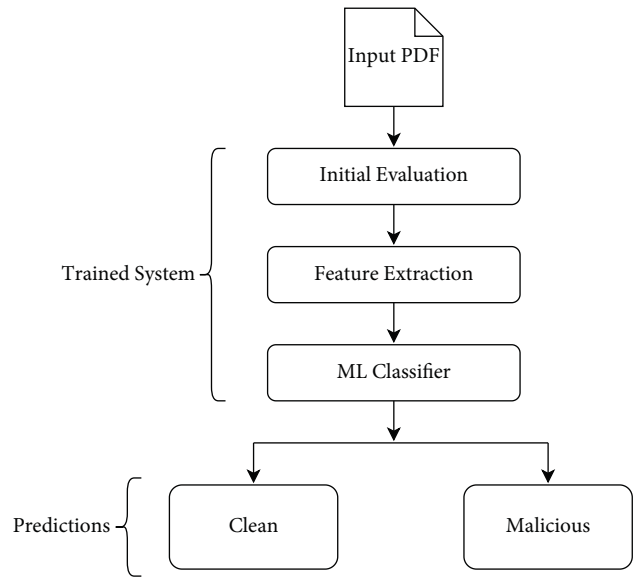


Figure 1: Proposed layout working of the model.

implemented. To begin the inspection procedure, the PDF file document must be uploaded to the system. After the document is submitted, it is first analyzed for its information and structure. It is tagged for additional assessment if it follows the pattern of known harmful files. This saves a lot of time and improves speed. In other circumstances, when it does not fit the pattern of hostile instances, the feature extraction module analyses the whole file structure and derives the features from it. It is then given to the classifier component for evaluation once characteristics have been extracted from the PDF. The main ML algorithm is located in the classifier component, and it is this algorithm that is responsible for thoroughly examining the information provided by the feature extractor component [37]. The classifier will categorise the PDF file as either a "correct" or an "infected" file after doing the necessary data analysis.

There may be some suspicious API references in the code that can only be discovered via the dynamic code assessment, which can only be done through a new static analysis. The static and dynamic code inspection both employed the same monitoring method as with the PhoneyPDF to keep an eye out for any API references. SpiderMonkey and Rhino are two examples of open-source facilitators that have been used to conduct dynamic investigations in the past. The JavaScript ECMA standard is seen by these translators, but they are unable to comprehend JavaScript connections to the Acrobat PDF format, unless Adobe DOM duplication occurs [38].

A reference design is developed by selecting a subset of API reference that depicts the harmful JavaScript code. Using a collection of PDF files that are either clean or malicious, our system can automatically build a specific set. Acrobat PDF API perceives all JavaScript objects, strategy, and capacity constants as part of the "$H$" arrangement. This enables us to define "$\Phi$" as the arrangement of all JavaScript objects, strategy, and capacity constants as well as constants. The total number of harmful and nonmalicious files is equal

to $M$. The following equation depicts the characteristic set provided by all of the references.

$$\sum_{i=0}^{M} (\Phi(H, i)) > \text{threshold}. \quad (1)$$

Also, it is to be noticed that $\Phi(H, i)$ may be holding two values and signifies −1 and +1. Also, if result comes out +1, then it signifies malicious PDF. If the value is −1, then it signifies that the file is a safe one.

## 4. Dataset Details

An overview of the dataset is provided in this section that is used in this proposed research. Following the benign set, the malicious dataset that we analyzed was provided. A total of 1200 PDF samples, both malicious and safe, have been obtained for the investigation. An 800-sample training set has been employed, and 400 samples have been used for testing as depicted in Table 1. It must have been important to have a ratio of good files to malicious files in the training and testing sets of 1 : 1. The majority of the samples are based on genuine cyber-attacks that have been made public. Samples are gathered from a variety of locations over the Internet.

Because the JavaScript code is included in many of these PDF files, some classifiers believe they are all malicious because of the file's large size. The approach in this work, on the other hand, does not use file size as a criterion for determining whether or not a file is harmful. To demonstrate this, the harmless JavaScript code is purposely inserted into nondangerous PDF files in order to make them seem as though they included the malicious JavaScript code. All dangerous and safe PDFs have been analyzed independently and the average size of safe and malicious files was of only approximately 800 kB difference, as shown in Table 2.

## 5. Implementation and Results

In this section, the various approaches that have been executed for the analysis and implementation of the proposed model are described. Here, the training and testing part is done consequently and the results inferred are discussed. A variety of methods have been used to study and implement this suggested approach, and they are all discussed in this section. This system has been trained on 800 PDFs using several ML classification techniques. With five alternative algorithms, including stochastic gradient boosting (SGB), random forest (RF), decision tree (DT), support vector classifier (SVC), and logistic regression (LR), a comparison is carried out to check how well the system performs under these algorithms.

The effectiveness of the proposed work is reflected by confusion matrix parameters obtained after classification. The confusion matrix comprises of training, testing, validation, and a combined matrix that reflects $\text{true}_{\text{positive}}$, $\text{true}_{\text{negative}}$, $\text{false}_{\text{positive}}$, and $\text{false}_{\text{negative}}$ outcomes. These parameters are further used to calculate the performance parameters like precision, recall, and F1-score using the following equations, respectively.

$$\text{Precision} = \frac{\text{true}_{\text{positive}}}{\text{true}_{\text{positive}} + \text{false}_{\text{positive}}}, \quad (2)$$

$$\text{Recall} = \frac{\text{true}_{\text{positive}}}{\text{true}_{\text{positive}} + \text{false}_{\text{negative}}}. \quad (3)$$

$$\text{F1 score} = 2 * \frac{(\text{precision} * \text{recall})}{(\text{precision} + \text{recall})}. \quad (4)$$

Figure 2 signifies that deploying RF, LR, and DT takes the least amount of time possible, and thus, they perform the classification in a faster manner. In comparison, the SGB's efficiency is average, whereas the SVC's is poor.

The "True Positive Rate (TPR)" is computed by placing the various classifiers during testing, and the trend line is produced. The system should have a higher TPR score in order to be the optimal match. Figure 3 shows that the RF approach has the highest TPR score among all the options. Moreover, the SVC seems inappropriate for file functional testing since it has the lowest TFR score value.

The same holds true when this suggested system's "Precision Score (PS)" under various classifiers was tested. Using Figure 4, it can be concluded that the RF is providing an average PS ratio of approximately 96 percent and that the SVC is providing the lowest PS at roughly 72 percent.

When calculating the "False Positive Rate (FPR)" when testing the system, it is deduced from Figure 5 that the RF has the lowest FPR score, which is preferred, and the SVC has the highest FPR score, which demonstrates its ineffectiveness.

During the calculation of the "False Negative Ratio (FNR)" score, Figure 6 shows that the RF, DT, and SGB all have the lowest FNR scores; hence, they come strongly recommended. SVC and LR, at the other hand, have a high FNR score.

In addition, RF has shown the best overall F1-score on the dataset when compared to the SGB. The DT's F1-score remained similarly moderate. RF has shown to be the best fit for our proposed system, whereas SVC had the worst results when tested with our system.

A number of other PDF malware classification techniques, created by a variety of authors, have been tested. It is evident that our system has the best fit when utilizing the RF classifier based on previous parts. Extraction of features relies on API calls performed by the document as well as the JavaScript code included inside its contents. The F1-score of the various classifiers is determined, and it is inferred that for the proposed classification method, it is higher in contrast to other classifiers as mentioned in Table 3. The numbers (F1-score) shown in the table are derived utilizing the same dataset, through which the testing was executed earlier.

## 6. System Analysis under Attacks

Malicious samples are developed to resist our system after it had been built, and it is supported by developing a

Table 1: File count for testing and training.

| | Safe files | Malicious files | Total files |
|---|---|---|---|
| Count for training | 396 | 404 | 800 |
| Count for testing | 175 | 225 | 400 |
| Total count | | | 1200 |

Table 2: File size (kB) details of dataset for evaluation.

| Category type of file | Maximum size | Minimum size | Average size |
|---|---|---|---|
| Safe | 21,365 | 2 | 10,657 |
| Malicious | 22,061 | 2 | 11,321 |

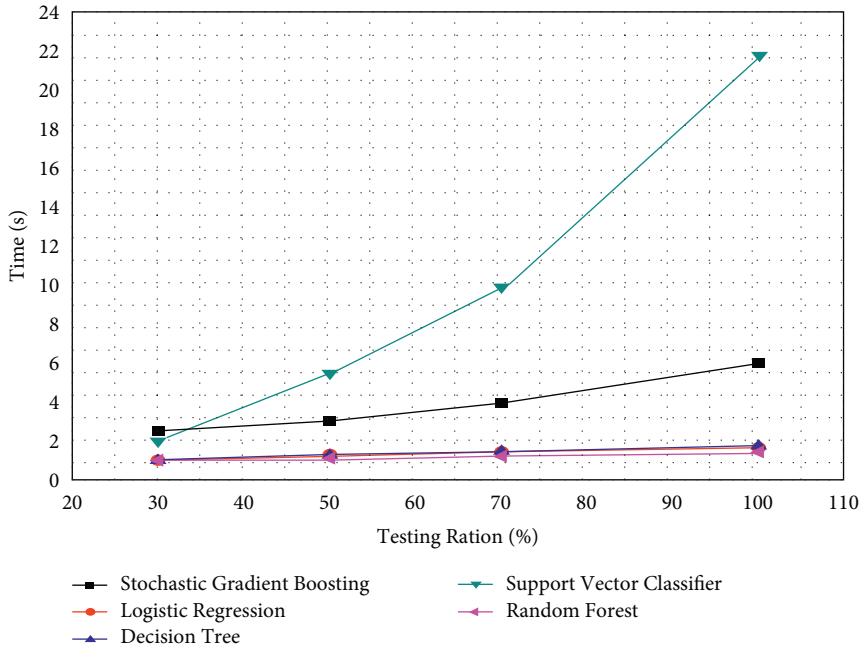


Figure 2: Performance evaluation under various classifiers.

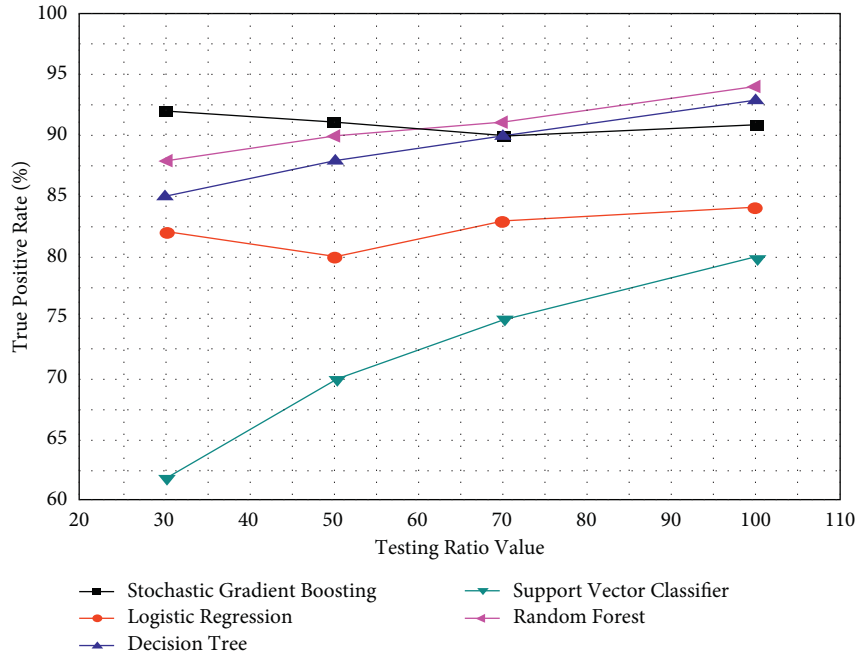


Figure 3: Analysis of TPR under the testing phase.

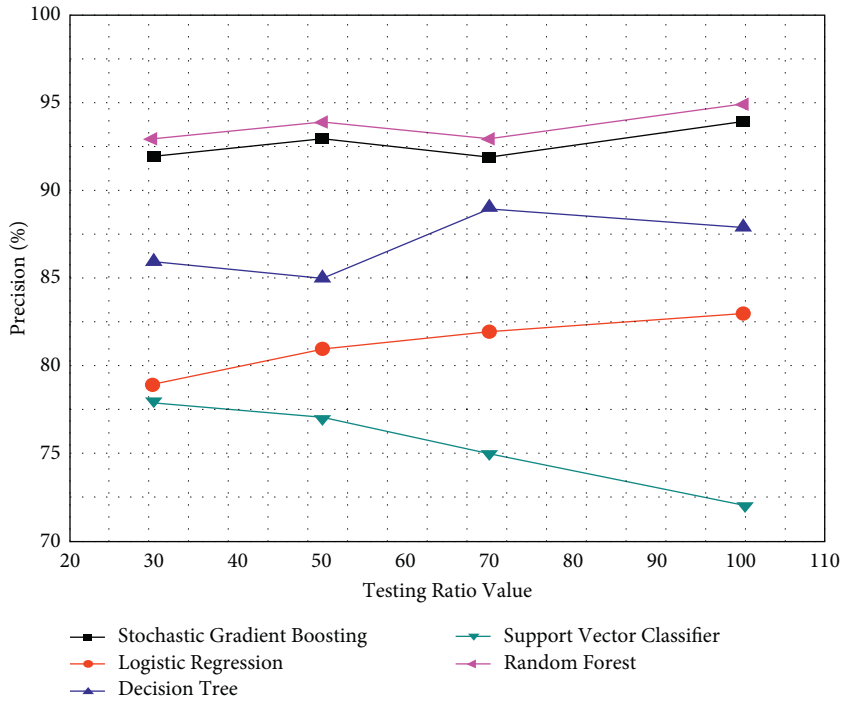


Figure 4: Precision evaluation under various classifiers.

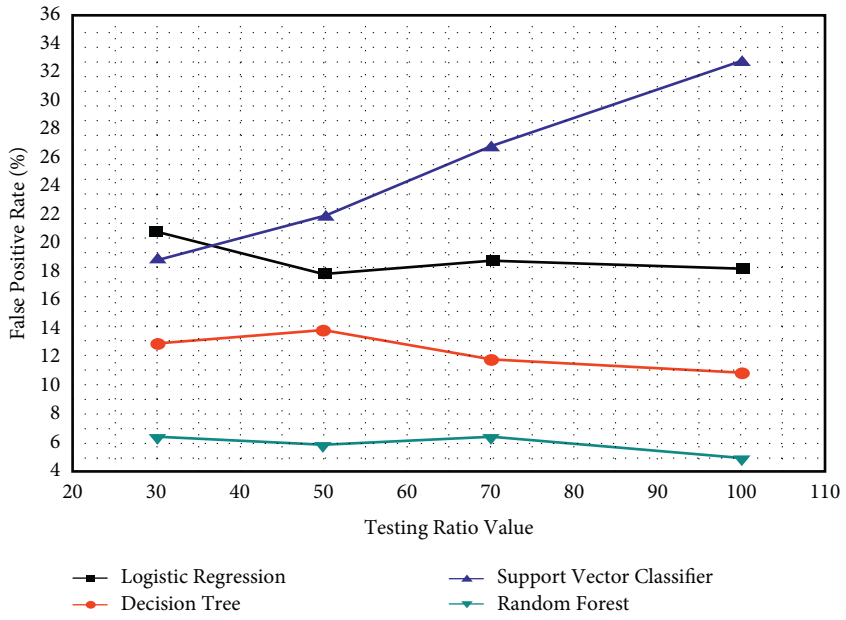


Figure 5: FPR evaluation under various classifiers.

mechanism to recognize those types of attackers during the testing step. As a general rule, while parsing PDF files, the parser first travels to the trailer and retrieves the location of the first item in the list of items in the body. When the first object has been entirely parsed, the program returns to the cross-reference table (CRT) and receives the second item's address. Since the harmful code is not processed or read when a PDF reader is requested, this work deleted the references to the body section objects that contain the dangerous code. Because of this, we may fool the parser into thinking that the file is secure, even if it has a harmful code inside it. If one wants to deceive the system into thinking a malicious file is safe, one may use this method. This is despite the fact that it has been tested using dynamic classifiers, which means that it can be inspected throughout the course of its execution. This code does not execute because it does not include any references to the portions of the body mentioned above. As a result, we may also send the

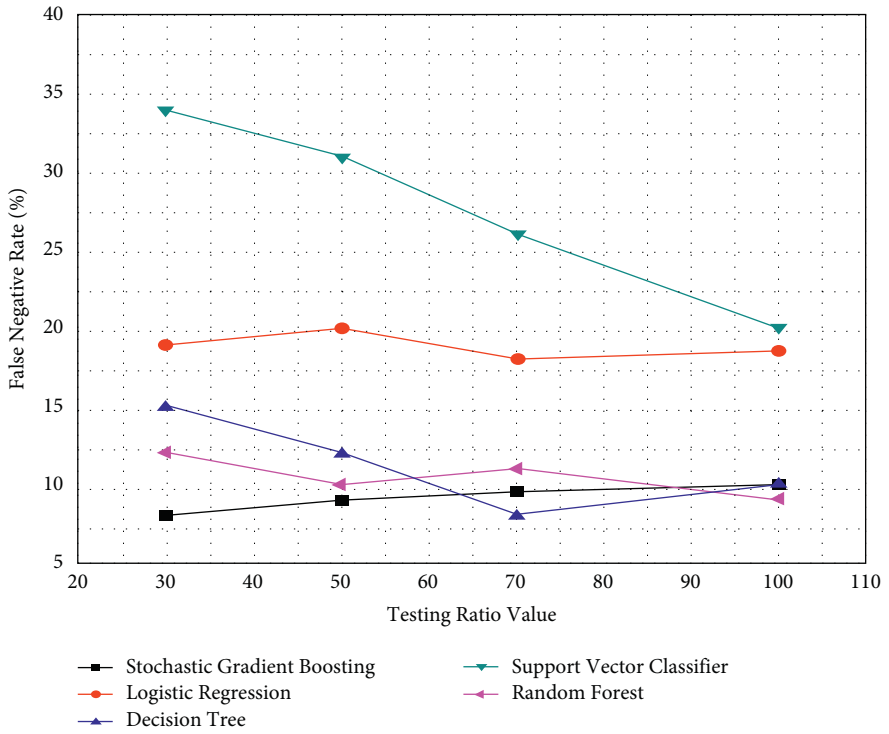


Figure 6: Analysis of FNR under the testing phase.

Table 3: Comparative analysis of the proposed model with existing based on F1-score.

| Tool reference details | Classifier used | F1-score |
|---|---|---|
| [39] | SVM | 0.828 |
| [40] | RF | 0.982 |
| [41] | RF | 0.778 |
| [32] | RF | 0.818 |
| [34] | RF | 0.980 |
| [33] | AdaBoost | 0.960 |
| [31] | DT | 0.658 |
| [35] | Bayesian | 0.978 |
| Proposed model | RF | 0.986 |

Table 4: Document classification under the malicious attack on the proposed system.

| Type of file | Normal stage | | Executing the malicious attack | |
|---|---|---|---|---|
| | Classified | Not classified | Classified | Not classified |
| Safe | 109 | 16 | 119 | 6 |
| Malicious | 122 | 53 | 170 | 5 |

malicious code-infected PDF file during dynamic analysis. The classification of documents under the malicious attacks is given in Table 4.

## 7. Conclusion and Future Scope

A ML model that can identify JavaScript and malicious API calls attacks in PDF files is provided in this paper. This work also tried out a number of alternative classifiers, including DT, RF, LR, SVC, and SGB, on the dataset to see how they performed. The RF classifiers within this work have produced the best results. A comparison of this approach with other PDF classifiers revealed that this proposed approach has a high F1-score of 0.986, making it 4 percent more efficient than the other most recent PDF classifiers. To further enhance the system's defense against malicious code obfuscation methods, functionality is included to run an object scanner within the PDF document to identify any objects that are not being processed. Unparsed objects containing the malicious code may be easily identified and removed using this approach. Future plans include adding support for other file formats. Use an advanced data mining approach for more detailed insights of documents. The use of ML during the detection and classification phase of malware is highly useful, but it fails against evasion attacks; thus, it must be explored in the future.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The author declares that there are no conflicts of interest.

## Acknowledgments

This research was supported by Taif University Researchers supporting Project number (TURSP-2020/215), Taif University, Taif, Saudi Arabia.

## References


[1] K. M. A. Alzarooni, "Malware Variant Detection," Doctoral Dissertation, UCL (University College London), London, England, 2012.

[2] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer Security: Principles and Practice*, pp. 978–980, Pearson Education, Upper Saddle River, NJ, USA, 2012.

[3] S. Alam, R. N. Horspool, I. Traore, and I. Sogukpinar, "A framework for metamorphic malware analysis and real-time detection," *Computers & Security*, vol. 48, pp. 212–233, 2015.

[4] A. Mehtab, W. B. Shahid, T. Yaqoob et al., "AdDroid: rule-based machine learning framework for android malware analysis," *Mobile Networks and Applications*, vol. 25, no. 1, pp. 180–192, 2020.

[5] Y. Alosefer, *Analysing Web-Based Malware Behaviour through Client Honeypots*, Doctoral dissertation PhD Thesis, Cardiff University, Cardiff, Wales, 2012.

[6] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Technical Report, Purdue University, West Lafayette, IN, USA, 2007.

[7] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 646–656, 2013.

[8] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proceedings of the 2015 10th International Conference on Malicious And Unwanted Software (MALWARE)*, pp. 11–20, IEEE, Fajardo, PR, USA, October 2015.

[9] L. Nataraj and B. S. Manjunath, "SPAM: signal processing to analyze malware [applications corner]," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 105–117, 2016.

[10] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security*, vol. 14, no. 1, pp. 1–14, 2015.

[11] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious JavaScript in pdf through document instrumentation," in *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 100–111, IEEE, Atlanta, Georgia, June 2014.

[12] K. Chumachenko, *Machine Learning Methods for Malware Detection and Classification*, Bachelor's Thesis Information Technology, Xamk Kouvolan kampus, Kouvola, Finland, 2017.

[13] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[14] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 3–22, 2018.

[15] A. R. Javed, M. O. Beg, M. Asim, T. Baker, and A. H. Al-Bayatti, "Alphalogger: detecting motion-based side-channel attack using smartphone keystrokes," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2020, pp. 1–14, 2020.

[16] M. F. Zolkipli and A. Jantan, "A framework for malware detection using combination technique and signature generation," in *Proceedings of the 2010 Second International Conference on Computer Research and Development*, pp. 196–199, IEEE, Kuala Lumpur, Malaysia, May 2010.

[17] Y. Fukushima, A. Sakai, Y. Hori, and K. Sakurai, "A behavior-based malware detection scheme for avoiding false positive," in *Proceedings of the 2010 6th IEEE Workshop on Secure Network Protocols*, pp. 79–84, IEEE, Kyoto, Japan, October 2010.

[18] H. H. Pajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, "Intelligent OS X malware threat detection with code inspection," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 3, pp. 213–223, 2018.

[19] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on opcode patterns," *Security Informatics*, vol. 1, no. 1, pp. 1–22, 2012.

[20] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[21] D. L. Vu, T. K. Nguyen, T. V. Nguyen, T. N. Nguyen, F. Massacci, and P. H. Phung, "A convolutional transformation network for malware classification," in *Proceedings of the 2019 6th NAFOSTED Conference on Information And Computer Science (NICS)*, pp. 234–239, IEEE, Hanoi, Vietnam, December 2019.

[22] S. K. Sasidharan and C. Thomas, "ProDroid—an Android malware detection framework based on profile hidden Markov model," *Pervasive and Mobile Computing*, vol. 72, Article ID 101336, 2021.

[23] Y. Jian, H. Kuang, C. Ren, Z. Ma, and H. Wang, "A novel framework for image-based malware detection with a deep neural network," *Computers & Security*, vol. 109, Article ID 102400, 2021.

[24] Y. Li, X. Wang, Z. Shi, R. Zhang, J. Xue, and Z. Wang, "Boosting training for PDF malware classifier via active learning," *International Journal of Intelligent Systems*, vol. 37, no. 4, pp. 2803–2821, 2022.

[25] A. R. Javed, W. Ahmed, M. Alazab, Z. Jalil, K. Kifayat, and T. R. Gadekallu, "A comprehensive survey on computer forensics: state-of-the-art, tools, techniques, challenges, and Future Directions," *IEEE Access*, vol. 10, pp. 11065–11089, 2022.

[26] A. R. Kang, Y. S. Jeong, S. L. Kim, and J. Woo, "Malicious PDF detection model against adversarial attack built from benign PDF containing javascript," *Applied Sciences*, vol. 9, no. 22, p. 4764, 2019.

[27] D. Maiorca and B. Biggio, "Digital investigation of pdf files: unveiling traces of embedded malware," *IEEE Security & Privacy*, vol. 17, no. 1, pp. 63–71, 2019.

[28] Y. Chen, S. Wang, D. She, and S. Jana, "On training robust {PDF} malware classifiers," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20*, pp. 2343–2360, Berkeley CA. USA, August 2020.

[29] C. Smutz and A. Stavrou, "When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors," in *Proceedings of the 23rd Annual Network and Distributed System Security Symposium, NDSS 2016*, San Diego, CA, USA, February 2016.

[30] N. Šrndić and P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," *EURASIP Journal on Information Security*, vol. 2016, no. 1, pp. 22–20, 2016.

[31] D. Maiorca, D. Ariu, I. Corona, and G. Giacinto, "A structural and content-based approach for a precise and robust detection of malicious PDF files," in *Proceedings of the 2015 International Conference on Information Systems Security and Privacy (Icissp)*, pp. 27–36, IEEE, Angers, France, February 2015.

[32] I. Corona, D. Maiorca, D. Ariu, and G. Giacinto, "Lux0r: detection of malicious pdf-embedded javascript code through

discriminant analysis of api references," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pp. 47–57, Scottsdale, ARI, USA, November 2014.

[33] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *Proceedings of the 19th International Conference on World Wide Web*, pp. 281–290, Raleigh North, CAR, USA, April 2010.

[34] A. Demontis, M. Melis, B. Biggio et al., "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 711–724, 2019.

[35] A. Pektaş and T. Acarman, "Malware Classification Based on API Calls and Behaviour Analysis," *IET Information Security*, vol. 12, no. 2, 2017.

[36] P. Panda, I. Chakraborty, and K. Roy, "Discretization based solutions for secure machine learning against adversarial attacks," *IEEE Access*, vol. 7, pp. 70157–70168, 2019.

[37] B. Chen, Z. Ren, C. Yu, I. Hussain, and J. Liu, "Adversarial examples for cnn-based malware detectors," *IEEE Access*, vol. 7, pp. 54360–54371, 2019.

[38] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[39] P. Laskov and N. Šrndić, "Static detection of malicious JavaScript-bearing PDF documents," in *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 373–382, Orlando, FL, USA, December 2011.

[40] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious pdf files detection," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 510–524, Springer, Berlin, Heidelberg, 2012.

[41] C. Smutz and A. Stavrou, "Malicious PDF detection using metadata and structural features," in *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 239–248, Orlando, FL, USA, December 2012.