International Journal of Computational Intelligence Systems
In Press, Uncorrected Proof
DOI: https://doi.org/10.2991/ijcis.d.190905.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Expdf: Exploits Detection System Based on Machine-Learning

Xin Zhou*, Jianmin Pang

State Key Laboratory of Mathematical Engineering and Advanced Computing, Henan, Zhengzhou, China

ARTICLE INFO

Article History
Received 02 Jan 2019
Accepted 29 Aug 2019

Keywords
Malware
Exploit
Pdf
Machine learning

ABSTRACT

Due to the seriousness of the network security situation, as a low-cost, high-efficiency email attack method, it is increasingly favored by attackers. Most of these attack vectors were embedded in email attachments and exploit vulnerabilities in Adobe and Office software. Among these attack samples, PDF-based exploit samples are the main ones. In this paper, we proposed Expdf, different from existing research on detecting pdf malware, a robust recognition system for exploitable code-based machine learning. We demonstrate the effectiveness of Expdf on the dataset collected from Virus Total filtered by the labels of multiple antivirus software. With the experimental evaluation compared to Hidost, Expdf demonstrates its superiority in detecting exploits, reaching the accuracy rate of 95.54% and the recall rate of 97.54%. Additionally, as the supplementary experiment, Expdf could identify specific exploit vulnerability types.

1. INTRODUCTION

Over the past few years, the advent of the informatization has had a far-reaching effects on our life. With the development of office automation, due to the high availability of attack tools, the cost of developing malicious software will also reduce [1].

According to the "Tencent Security 2017 Internet Safety Report," the malware with exploitable code are mainly distributed on Windows, Linux, and Android platforms. According to the statistics with samples collected, it could be seen that the Windows platform accounts for the most, of which the PE type vulnerability samples reached 25.21% and the non-PE type vulnerability samples reached 65.60%. And the total number of Windows platform vulnerability samples could reach 90.81% of the total samples of all platform-wide vulnerability.

Since the first vulnerability exposed in Adobe Reader in 2008, the malware as Portable Document Format (PDF) was turning into the favorite file type exploited by hackers [2]. According to the Internet Security Threat Report [3], the PDF malware ranked the top 7 in spear fishing emails in 2014. In accordance with the Cisco 2018 Annual Cyber Security Report, the Cisco researchers conducted analysis on the 3.9 million suspiciously downloaded documents, of which 34% are PDF malware and 31% are Microsoft Office documents.

Through sites such as virus share, we collected 4,709,478 malicious samples with corresponding virus total information. Based on the antivirus software labels in the virus total, we filtered a total of 15,274 samples with exploitable code. On the basis of these samples, we used the type of samples and the time when the virus total was first submitted to do statistical analysis, as can be seen in Figure 1. The PDF malware with exploitable code have emerged since 2008 and are on the rise, which have gradually become the main type of malware with exploitable code.

In recent years, Advanced Persistent Threat (APT) has received high attention from society, government, and enterprises. Known as Operation Aurora, the attack was extremely widescale and was believed to have targeted 34 organizations [4]. The cyber threats are mainly caused by persistently attempt to gain and maintain access, that is, covert long-term penetration attacks on specific targets.

According to the TrendLabs report [5], 91% of targeted attacks in an APT attack involved spear-mail attacks. By means of a low-cost, high-efficiency email attack method, most of the attack vectors were embedded in email attachments, and exploited vulnerability on Adobe and Office software. In January 2017, FireEye Company exposed the APT28 organization's cyber attacks on U.S. government agencies and interfered with the U.S. presidential election [6]. The events, such as the U.S. election at the end of 2016, the mail door incidents of Hillary and the Democratic National Committee of the United States (DNC), made the public first witness the profound influence of the APT attack on geopolitics and even the state power. In May 2017, ESET, FireEye and other security agencies issued reports that they found APT28 interfered with the French presidential election with spear-mail attacks [7]. In October 2017, Kaspersky issued a report [8] that BlackOasis spread Spyware FinSpy by exploited Adobe 0 day vulnerability CVE- 2017-11292. In December 2017, FireEye claimed that they found APT34 used the just-repaired vulnerability CVE-2017-11882 to attack the Middle East government [9].

*Corresponding author. Email: qf_zhouxin@126.com.
This research was supported by the project of the National Science Foundation of China under Grant No. 61472447, 61802435.

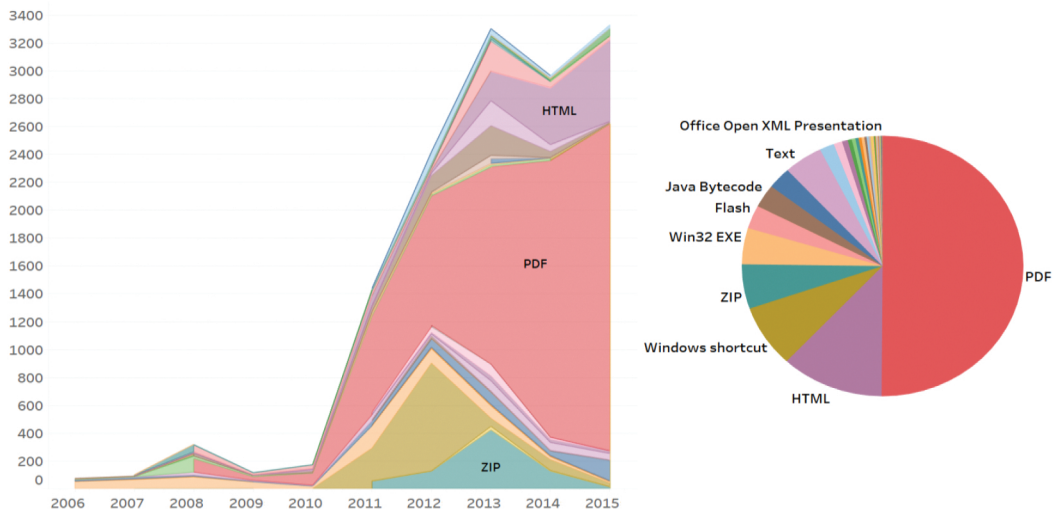

Figure 1 Distribution of different types of malicious sample with exploitable code.

In March 2018, an unknown hacker group inadvertently leaked two 0-day vulnerabilities (CVE-2018-4990 and CVE-2018-8120) while uploading a PDF file for attack to the public malware scan engine. ESET researchers demonstrated they found malicious PDFs in the public malware repository (VirusTotal), and manually analyzed that these two 0-day vulnerabilities were used together to form a chain of attacks [10]. And the sample did not contain a final payload, which may suggest that it was caught during its early development stages.

As an inspiration for system development, we should pay more attention to the cause of the incident. Before hackers use the tools they developed, they worked on fine-tuning their exploits by submitting them to the public malware scan engine for checksum verification. Therefore, we should increase the level of attention to the detection of exploits of malicious samples, and thus get the 0-day vulnerability by analyzing new samples.

In this case, we designed and studied Expdf to detect and identify exploitable samples in pdf malware. It could be seen that APT and other hacker organizations frequently use spoiler mail attacks by exploiting office software vulnerabilities such as Adobe, among which PDF-based exploit samples are the main ones. Therefore, we chose PDF as the research object to establish Expdf for quickly and effectively identifying PDF samples with exploits. By identifying the exploit samples in the malware, the unknown samples can be effectively responded and repaired in time, providing the knowledge base for the analysis.

At present, the research of pdf exploitable code detection is scarce, almost by means of the signature code. However, the problem of pdf malware detection has more research recently. In paper [11], Nedim et al. proposed Hidost, a malware detection system for detecting malicious pdf and SWF files by path analysis of pdf files. In paper [12], PDFrate parses PDF files to retrieve structural features and uses the random forest (RF) as classification model to detect malware. Meanwhile, there are also many researches use JavaScript extracted from the PDFs as the features, such as [2] and [13].

In our work, we obtained and evaluated the Expdf in detecting pdf malware with exploitable code. Different from existing research on detecting pdf malware, Expdf aims to design a robust recognition system for exploitable code. So as to Expdf, we combine bioinformatics and genetics to propose the pdf exploitable malware gene for detection whether the exploits are exploited in the pdf malware.

Our contributions can be summarized as follows:

1. We proposed the Pdf Exploitable Malware Gene that could be used to detect pdf samples with exploits, that can be automated and efficiently extracted statically.
2. We used the RF to build the Expdf that could detect the existence of pdf exploits.
3. We collected the experimental dataset using virus total to ensure the ground truth.
4. We evaluated the Expdf through experiments on pdf malware with exploitable code detection and classification.

2. OVERVIEW

The structure of this paper is as follows: we describe the overview in Section 2. And in Section 3, we introduce the relevant theory of pdf document. And in Section 4, the system model and algorithm will be elaborated. The results of the experiment and evaluation will be shown in Section 5. Finally, we make a summary of the work and prospects in Section 6.

2.1. Machine Learning for Classification

Machine learning [14] refers to learning the inherent regularity information in the data through computer, and gaining new experience and knowledge to improve the intelligence of the computer and enable the computer to make decisions like people. The goal is to build a mathematical model based on a certain network structure, select the corresponding learning method and training method, learn the data structure and internal model of the input data, continuously adjust the model parameters, and solve the model's optimal predictive feedback through mathematical tools. Improve generalization ability and prevent over-fitting.

The basis and key to train the model of machine learning is the extraction of features. In a machine learning system, samples are represented by feature extraction modules mapped into feature vectors in the feature space. For example, for the maliciousness determination of the malware, the function call sequences of the malicious code could be extracted as feature vectors, and each call sequence is used as a dimension in the feature space, so that different samples are mapped one-to-one into different vectors.

However, due to the classification algorithm has certain computational constraints on the dimensions of the feature, and the effect of the corresponding model are not in direct proportion. Because of the feature of large-scale dimensions, only a part of it may be have positive effects, while the other are equivalent to the existence of noise in the classification model. Therefore, it is crucial to reduce the dimensions of high-dimensional features.

Machine learning is divided into supervised learning and unsupervised learning based on whether the training dataset have artificially set tags. In our work, we will train the model based on supervised learning, in other words, the training dataset set with the category label for each sample. Through training, the computer learns from the sample data and builds a learning model, and uses the model to predict the category corresponding to the new sample.

2.2. Detection Model

The detection model is essentially trained by constructing a mathematical model through the calculation of the input data. The test dataset is input into the model, and the predicted value corresponding to different categories is calculated. The predicted value is a number (a real number between 0 and 1), corresponding to the different categories is compared with the threshold (default is 0.5).

2.3. Obtain Exploit Samples

The collection of the dataset and the guarantee of its ground truth are especially critical for the construction and verification of the model. Virus Total [15] is an information aggregator, it aggregates many antivirus products and online scan engines to check for viruses that the user's own antivirus may have missed, or to verify against any false positives. And in the security world, using Virus Total has certain Credibility. Therefore, we analyzed the results of the samples scanned using Virus Total, and use massive labels set by antivirus software to ensure the ground truth of the dataset.

In our work, we obtained a large number of malware from Virus Share [16] and hybrid-analysis, filtered according to the and the results of the antivirus software. Based on this, we collected 754 pdf malware with exploitable code, which could be ensured ground truth, and 1368 pdf malware without exploitable code.

3. PDF RELATED INTRODUCTION

In our work, we selected the PDF malware as the research object, so the focus of our work is how to find and analyze the PDF malware with the exploitable code from the malware. In this section, we introduced the information about the pdf format.

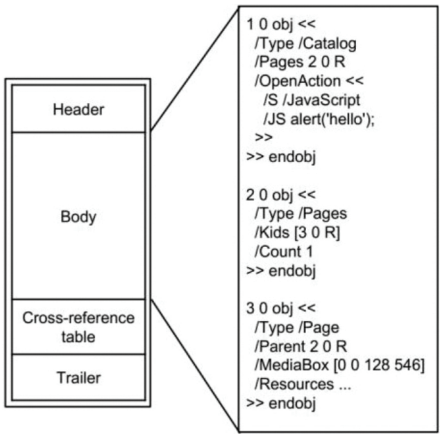

Figure 2 The structure of Portable Document Format (PDF) document.

The structure of a PDF can be understood from two aspects: file structure and logical structure. The file structure of a PDF refers to the physical organization of its files, and the logical structure refers to the logical organization of its contents.

According to the PDF format standard [17], the file structure could be divided into four parts (as shown in Figure 2): header, body, cross-reference table, Trailer. The file header indicates the version number of the PDF specification that the file complied with, which appears in the first line of the PDF file. The body is also called the collection of file object. The main part of the PDF file is composed of a series of objects. Cross-reference table, an address index table of an indirect object established by random access to an object. (The object address is actually stored in offset and index). Trailer declares the address of the cross-reference table, which indicates the root object of the file body (Catalog), so that the location of each object in the PDF file could be found to achieve random access. In addition, in the trailer, security information such as encryption of PDF files is saved.

The PDF logical structure is embodied in a hierarchical structure. A structured pdf document is composed of a number of "object" modules.

The trailer indicates the object number of the root object and indicates the location of the cross-reference table. The directory object (Catalog) can be found by querying the cross-reference table. This directory object is the root of the PDF document and contains both the outline of the PDF document and the page group reference. Therefore, we can map the logical structure of the PDF document into a tree structure by parsing the keywords. As shown in the Figure 3, it is a partial tree structure diagram obtained by parsing a certain pdf malware.

The nodes in the tree structure diagram are identified by the keywords of the pdf document, and the Table 1 gives the explanation corresponding to each keyword. The /Filter keyword saves the encoding information of the stream, such as /FlateDecode, /ASCIIHexDecode and /RunLengthDecode. By parsing the keywords /JS, /JavaScript, we can extract the JavaScript code. Therefore, through the analysis of the PDF structure, the internal information of the PDF could be preserved in the form of a tree structure, which is complete and without missing information.

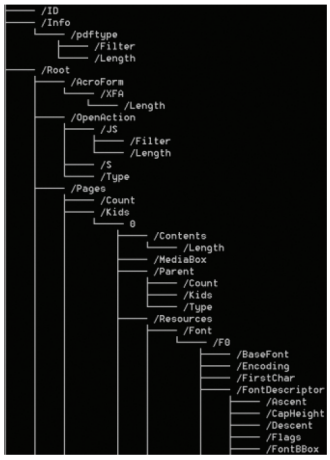

Figure 3 The partial tree structure diagram of pdf malware.

Table 1 The meaning corresponding to each keyword.

Keyword	Meaning
obj endobj	The sign of object indicated the start and end
stream endstream	The sign of stream indicated the start and end
xref	The sign of cross-reference table started
trailer	The sign of trailer started
startxref	The sign of cross-reference table ended
/Page	Number of pages
/Encrypt	Indicates whether the file is encrypted
/ObjStm	Number of objectstream
/JS	Embedded with JavaScript code
/JavaScript	Embedded with JavaScript code
/AA /OpenAction /AcroForm	Open objects automatically for specific features
/URI	Embedded url links
/Filter	Indicates the following stream is encrypted.
/RichMedia	Rich text
/Launch	The number of times the OpenAction is executed

4. EXPDF SYSTEM DESIGN

Authors should discuss the results and how they can be interpreted in perspective of previous studies and of the working hypotheses. The findings and their implications should be discussed in the broadest context possible. Future research directions may also be highlighted.

4.1. System Design

Expdf has been designed as the exploits samples detection on pdf malware, exploits detection system capable of learning to discriminate between malware with exploitable code and without it. As shown in the Figure 4, the model of Expdf contains five main stages: Filtered with VirusTotal, Gene extraction, Gene Vector, Traing Model, Model Validation. With the labels of VirusTotal, the pdf malware filtered on whether the samples contained exploitable code or not. Through the gene extraction, we transformed it into a tree diagram, and extract genes based on the path of the leaf node and the corresponding value. Gene vectorization we used transformed the genes into numeric vectors for processing by machine learning models. Finally, based on the machine learning, the training model we built to makes a decision whether the samples contained

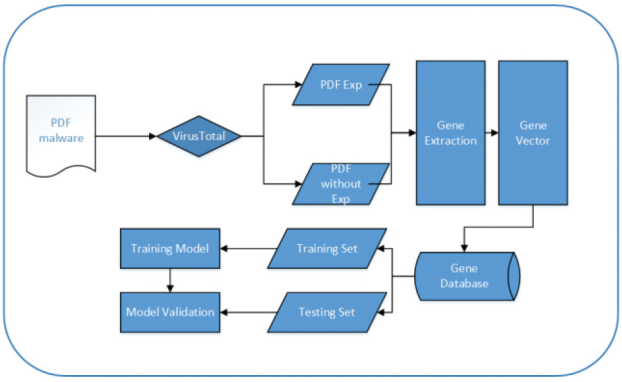

Figure 4 Expdf system design.

exploitable code or not. Meanwhile, we used the 10-fold cross validation to test the model performance. In the following subsections, the main stages of our approach are presented in detail.

4.2. Gene Definition and Extraction

In our work, we proposed the pdf exploitable malware gene, which could be utilized to detect and classify pdf exploitable malware. The perfection of the theory of genetics and bioinformatics has greatly promoted the development of malware gene theory research. Similar to the definition of biological genes, software genes [18] are defined as follows:

Definition: software genes are code fragments of programs that carry functional information.

As a result of the pdf exploit malware is a special kind of pdf malware, we defined the pdf malware gene first [19]. And then, we filtered the pdf malware gene according to the experiment to obtain the pdf exploit malware gene. Owing to the specificity of pdf malware file type, we defined the pdf malware gene based on their structure and behavioral information. Through the parse of pdf malware, we defined the pdf malware gene based on the properties of the software gene, so that it could simultaneously express the structural information and behavior information of the sample.

The method of extraction of pdf exploitable malware gene could be seen in Algorithm 1.

Since the intrinsic logic of PDF is indexed by cross-reference table, the logical structure of PDF could be regarded as a circular graph with orientation rooted. Therefore, in order to normalize the graph, we map it to the root tree by limiting the number of loops. As shown in the Figure 2, /Pages/Kids/Parent and /Pages point to the same object. In this case, there will be a case of directed infinite loops. Therefore, even if the pdf document is parsed by cross-reference table, two different path structures may point to the same object. Moreover, many of the pdf malware have problems with missing or incorrect structural information now. Based on these difficulties, we used the python-based open source parsing library pdfrw, version 0.4. The parsing library could solve the problem that pdf malware structure information is missing or incorrect.

Meanwhile, pdfrw automatically handles indirect references on reading in a PDF file. When pdfrw encounters an indirect PDF file

Algorithm 1: Pdf exploitable malware gene extraction

Input:
Pdf malware;
Output:
Sequences of pdf malware gene;
1: **Traverse** the cross-reference table composed of the identity of objects:
2: **for** *parsing keyword identifier of the pdf malware* **do**
3: **if** *dr* > *dm* **then**
4: Terminate the subpath calculation;
5: **end if**
6: **if** *Multiple keywords* are the same **then**
7: Node identification mark distinction;
8: **end if**
9: Convert pdf logical structure to directed rooted loop graph;
10: **end for**
11: Map pdf malware into rooted trees;
12: **Traverse** the leaf node composed of the information and path:
13: **for** *path of the leaf node* **do**
14: Extract the structure path and stream;
15: **end for**
dr: /Kids keyword node depth
dm: The depth limited

object, the corresponding Python object it creates will have an 'indirect' attribute with a value of True. Therefore, when we parse the pdf data, we just need to make sure that circular references are broken up by putting an attribute named 'indirect' rather evaluations to True on at least one object in every cycle.

As shown in the Figure 5, Expdf solve the structure of the pdf document, draw it into a tree diagram, and extract genes based on the path of the leaf node and the corresponding value. To solve the problem that /Pages and /Pages/Kids/Parent indirect reference to the same object, we took the method of limiting the tree depth of the Keys keyword node, truncating the loop branch with dr>dm. In our work, we set dm = 3. Since the stream in /Kids keyword is parsed as the type of array, instead of dict in python. Therefore, Expdf use the method of naming the Kids and the subscript splicing string to split the path at the node, as shown by /Kids0 in the Figure 3.

Different from existing research on detecting pdf malware, such as Hidost, Expdf aims to design a robust recognition system for exploitable code. For Hidost, the gene extraction reduces the dimension of genes by merging paths to solve the path polymorphism problem. For example, /Pages/Kids/Resources/Font/ path represents the font of PDF, and other fonts such as F0 and F2 are the same type of feature to some extent. In Hidost, the path of this type was merged at first. However, this method has certain defects in the identification of exploits. For example, CVE-2017-8454, remote attackers could exploit vulnerabilities to obtain sensitive information or execute arbitrary code by constructing PDF document fonts. In CVE-2010-1797, FreeType is a font function library that allows remote attackers to execute arbitrary code or cause denial of service (memory corruption) by means of specially crafted CFF opcodes in embedded fonts in PDF files. It could be seen that extracting genes using path merging is not suitable for detecting exploitable code.

Therefore, different from Hidost, Expdf constructed a library of pdf exploitable malware genes by parsing the samples. If the pdf exploitable malware gene appears in the sample, the feature is converted to 1. Set to 0 if it does not exist.

However, due to the ample related to the exploit is more critical for the encryption and decryption related genes, the feature just only mapped into binary values is not enough. Therefore, on this basis, we performed entropy analysis on the stream information. By means of the entropy calculation with stream and feature fusion with path feature, we mapped the exploitable genes into new vectors.

A mathematical tool for measuring the entropy of information is required to apply the concept of entropy measurements, and Shannon's formula was devised as Eq. (1):

$$\mathrm{H}(x) = -\sum_{i=1}^{n} p(i) \cdot \log_b p(i) \tag{1}$$

where H(x) is the measured entropy value and $p(i)$ is the probability of the i-th unit of information in the series of n symbols of event x. The base number of the logarithm (b) can be any real number greater than 1. In our work, we set b = 2. In our work, we obtain the path and stream through gene extraction, feature

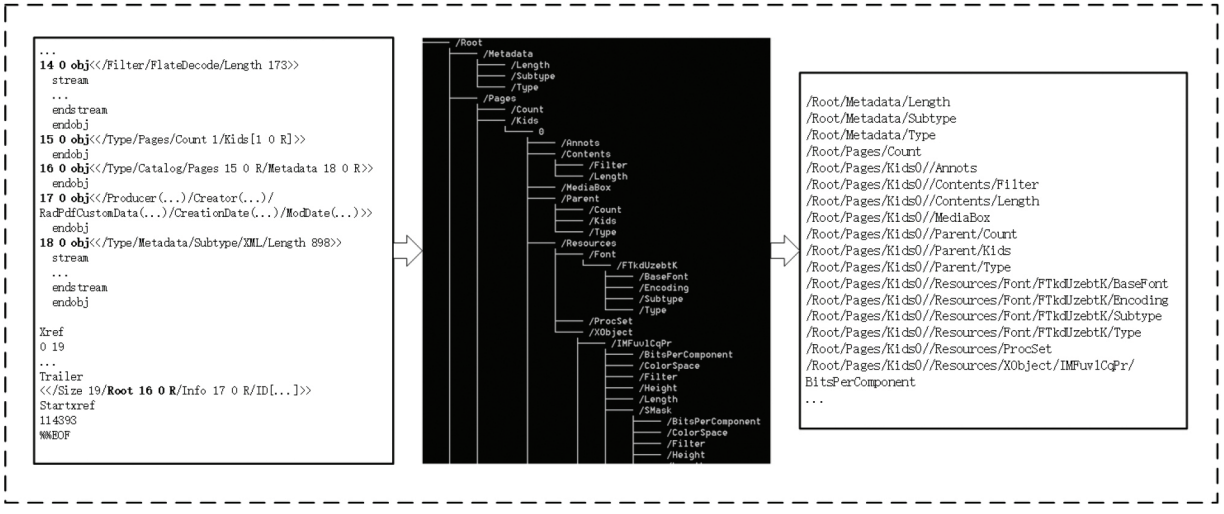

Figure 5 The procedure of gene extraction in the Expdf.

mapping through entropy and path analysis, and form a new feature vector based on feature fusion. And then, we used the Expdf gene vectors as input to the classification model.

4.3. Classification Model

Expdf employed RF [20] as the classification model to distinguish whether there is a pdf malware with exploitable code.

As an ensemble learning method, RF consists of massive independently trained decision trees (the number of which called n_estimators) [21]. In the training step, every tree is learned using CART methodology, but using only a subset of the available training samples. A different subset is generated for every tree by randomly sampling a fixed number of times from the training data, with replacement. When a new decision node is added to a tree, only a randomly chosen subset of max_features features is considered, where max_features is less than the total number of features. A decision is made by majority voting among all decision trees on a given new data point. RFs are known for their excellent generalization ability and robustness against data noise. For experiments presented in this paper, the RF implementation of the open-source scikit-learn [22] Python machine learning library, version 0.19.1, was utilized. The n_estimators and max_features are parameters of RF implemented with scikitlearn.

It is trained by growing a forest of decision trees using CART methodology. Each of the decision tree trees is grown on its own fixed-size random subset of training data drawn with replacement. At every branching of a tree during training, the feature providing the optimal split is selected from a random subset comprising max_features features not previously used for this tree. During classification, the decision of every tree is counted as one vote and the overall outcome is the class with the majority of votes.

5. EXPERIMENTAL EVALUATION

5.1. Experiment Dataset

In our work, we collected a large number of malware from Virus Share and hybrid-analysis, filtered according to the Virus Total and the results of the antivirus software.

In order to ensure ground truth, we considered those pdf malware as exploits samples that were labeled as malicious by at least five antivirus engines and labeled as exploit by at least 60% of the number of malicious labels. Meanwhile, we considered the pdf malware as without exploitable code that were labeled as malicious by at least five antivirus engines and without labeled as exploit. The other samples were discarded from the experiment because of the high uncertainty of their true class label.

On the basis of filtered samples, experiments were run on two datasets. The one we used contained 754 pdf malware with exploitable code and 1368 pdf malware without exploitable code, which we collected between 2017 and 2018. Table 2 lists the distribution of samples in the other dataset. The second dataset contained four types of exploit vulnerability, such as CVE-2010-2883, CVE-2010-0188, CVE-2011-2462, and CVE-2013-0640.

5.2. Experiment Result

5.2.1. Classification performance

In order to test the classification performance of the Expdf, we implemented three groups of experiments. For the first group experiment, we used the gene extraction method based on the Hidost to extract genes from samples. We used Gradient Boosting Decision Tree (GBDT), Support Vector Machine (SVM), extremely randomized trees (ET), and RFs as the classification models, using 10-fold cross validation to test the model performance. As for the second group experiment, we used GBDT, ET, RF, and XGBoost (XGB) as the classification models for test the Expdf classification performance.

In order to better describe the performance of classification model, we use four indicators to measure it: accuracy, precision, F1 score, and recall rate.

As a binary classifier, there are only two categories of input and output, and the indicator description matrix is shown in Table 3.

The accuracy means the proportion of samples classified correctly by classifier in all samples. The precision means the proportion of positive determined by the classifier in all positive samples actually. In all samples determined by the classifier correctly, the proportion of positive actually we called recall rate. As a result of a single accuracy and recall couldn't fully explain the problem, so we introduce F score, as the average of the accuracy and recall rate.

The calculation formula is as follows:

$$
\begin{aligned}
&TPR = TP / (TP + FN)\\
&FPR = FP / (FP + TN)\\
&Accuracy = (TP + TN) / (TP + FP + FN + TN)\\
&Precision = TP / (TP + FP)\\
&Recall = TP / (TP + FN)\\
&F1 = (2 * P * R) / (P + R)
\end{aligned}
$$

The recall is more important than the precision, because missing a exploitable code is considered more dangerous than incorrectly identify a non-exploitable code.

As shown in Tables 4 and 5, the results in the first two groups experiment we used RF as the classification model represented. In our work, we chose Hidost as the comparison model of ExPDF. This is because there are few studies on exploits detection of PDF, and the research on PDF malicious detection is mature. Among them,

Table 2 Distribution of samples of the second dataset.

CVE ID	Number
CVE-2010-2883	25
CVE-2010-0188	49
CVE-2011-2462	25
CVE-2013-0640	21

Table 3 Indicator description matrix.

		Predicted	
		Positive	**Negative**
Actual	Positive	TP	FN
	Negative	FP	TN

Table 4 Evaluation results using 10-fold cross validation-based Hidost.

	MEAN	1	2	3	4	5	6	7	8	9	10
Acc	0.93966	0.959	0.942	0.919	0.919	0.936	0.936	0.959	0.947	0.947	0.929
Precision	0.94723	0.982	0.935	0.948	0.956	0.928	0.942	0.966	0.943	0.936	0.933
F1	0.95704	0.965	0.958	0.944	0.939	0.954	0.953	0.970	0.970	0.962	0.949
Recall	0.9686	0.949	0.983	0.940	0.932	0.983	0.966	0.983	0.983	1	0.965

Table 5 Evaluation results using 10-fold cross validation-based Expdf.

	MEAN	1	2	3	4	5	6	7	8	9	10
Acc	0.95535	0.948	0.960	0.948	0.936	0.953	0.948	0.971	0.965	0.959	0.965
Precision	0.95919	0.966	0.951	0.958	0.965	0.959	0.950	0.975	0.959	0.951	0.959
F1	0.96717	0.961	0.971	0.958	0.957	0.967	0.962	0.979	0.975	0.971	0.971
Recall	0.9754	0.958	0.992	0.958	0.941	0.975	0.975	0.983	0.991	0.991	0.991

Hidost and PDFRate are excellent in malicious PDF detection models. And Hidost is a classification model based on the PDF file structure as feature, but the Hidost model only considers the structure without considering the content information. Therefore, we chose the Hidost model as a comparison model, which could explain that the structure and information are extremely critical for the problem of exploits detection, and ExPDF our proposed has excellent performance in the problem of exploit identification. Although both ExPDF and PDFRate use RF technology, the difference is that the PDFRate summarized 202 features based on human experience and used a special PDF parser to extract specific features. These particular features are not apply to the issue of exploits, so we were not select PDFRate as a comparison model. Compared with the feature extractor by Hidost, the Expdf adopted in our work has the best classification effect, reaching the accuracy rate of 95.54% and the recall rate of 97.54%.

To make the experiment more fully, we built a supplementary dataset, which expand the original dataset. In other words, the samples are manually created based on the metasploit tool and the exploit principle. Since the label of the dataset is manually created, the label of the exploits can be completely convinced. In order to avoid the proportion of manual samples is too large, it would affect the training effect of raw samples we collected on the model. In our work, we built 1000 manual samples based on the metasploit tool. According to the distribution ratio of the raw samples, the manual dataset contained 350 pdf malware with exploitable code and 650 pdf malware without exploitable code. Based on this, we denoted the original collected dataset as raw dataset, and denoted the extended dataset as mix dataset. Therefore, mix dataset has a total of 3122 samples, which contains 1104 pdf malware with exploitable code and 2018 pdf malware without exploitable code. As shown in the Table 6, due to the expansion of the dataset, ExPDF performance is also enhanced, reaching the accuracy rate of 97.40% and the recall rate of 98.39%.

ROC [23] and AUC are also indicators of the evaluation classifier. The ROC curve is drawn by two variables. The abscissa is the False Positive Rate (FPR), and the ordinate is the True Positive Rate (TPR). AUC is the size of the area under the ROC curve, usually between 0.5 and 1. The larger the AUC, the higher the performance of the classifier. As shown in Figures 6, the results in the first two groups experiment were represented. Compared with other classification model, the RF adopted in Expdf has the best classification effect, reaching the AUC of 0.98. By means of the ROC curve, is could be said that using features in Expdf is more stable than features in Hidost. Meanwhile, the feature extracted through Expdf is

Table 6 Compared results with Hidost.

	Raw dataset		Mix dataset	
	Hidost	ExPDF	Hidost	ExPDF
Acc	93.97	95.54	94.01	97.40
Precision	94.72	95.92	94.82	96.51
F1	95.70	96.72	95.92	97.44
Recall	96.86	97.54	97.05	98.39

Table 7 Evaluation results of classification on multiple vulnerabilities.

	Mean (%)
Acc	99.17
Precision	98.33
F1	99.09
Recall	99.99

less than Hidost affected by the different classification models. In the same time, we also could find that the tree-based classification models are more superb than others to classify the pdf malware with exploitable code.

For the third experiment, we evaluated performance of Expdf on different types of vulnerabilities based the second dataset. As shown in Table 7, the accuracy of RF we performed as classification model to differentiate multiple vulnerabilities reaching 99.17%. With the evaluation on CVE-2010-2883, CVE-2010-0188, CVE-2011-2462, and CVE-2013-0640, the recall rate could reaching 99.99%.

RandomForestClassifier (bootstrap=True, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)

Not only detect whether pdf malware have exploitable code or not, as a robust recognition system for exploitable code, Expdf could identify specific vulnerability types.

As shown in the Figure 7, each sub-bitmap in the gene bitmap represents the frequency of pdf exploitable malware genes in different exploit types (black represents the highest frequency and white represents the lowest). Therefore, we could intuitively feel through Figure 7 that the exploitable genes of different exploit types are different. The exploitable genes contained in CVE-2011-2462 are more

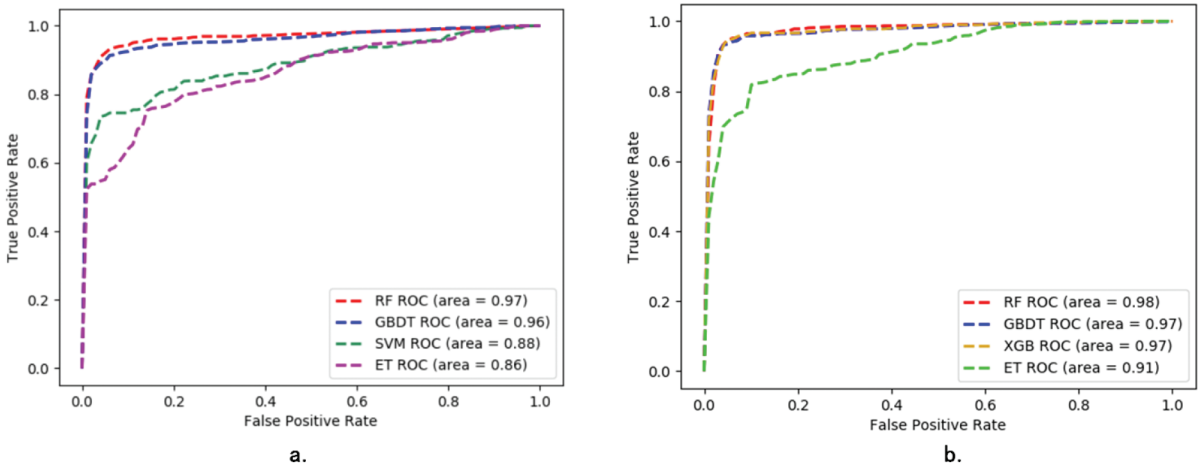

Figure 6 The ROC curve of the first two groups experiments.

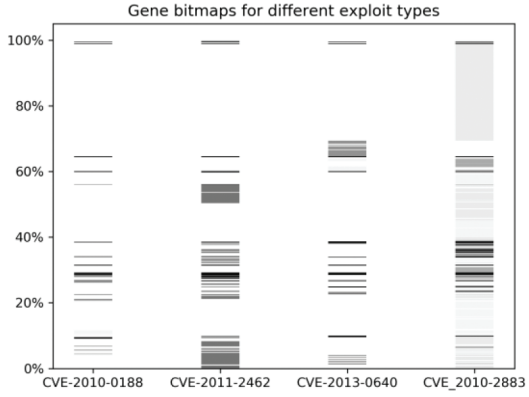

Figure 7 Gene bitmaps for different exploit types.

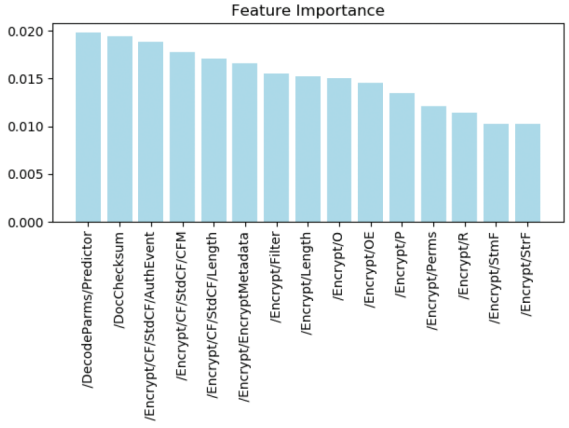

Figure 8 The histogram of features importance ranking.

intensive, while the exploitable genes of CVE-2010-2883 are more dispersed with the clear distinction.

5.2.2. Feature evaluation

In order to evaluate features more effectively, we implemented the following experiment.

Let *VIM* denote variable importance measures and *GI* denote the *Gini* index. Suppose there are m features $X_1, X_2, X_3, \ldots, X_m$, we need to calculate the *Gini* index score $VIM_j^{(Gini)}$ of every feature X_j, $j \in 1, \ldots, m$, namely, the average amount of change of node splitting impurity in all RF decision tree of j-th feature.

The calculation formula of *Gini* index is

$$GI_m = \sum_{k=1}^{|K|} \sum_{k' \neq k} p_{mk} p_{mk'} = 1 - \sum_{k=1}^{|K|} p_{mk}^2$$

where K is the number of categories and p_{mk} is the proportion of category K in node m.

Before and after the node m branches, the *Gini* index change is

$$VIM_{jm}^{(Gini)} = GI_m - GI_l - GI_r$$

where GI_l and GI_r denote the *Gini* index of two new nodes after branching.

If the node where the feature X_j appears in the decision tree i is in the set M, then the importance of X_j in the i-th tree is

$$VIM_{ij}^{(Gini)} = \sum_{m \in M} VIM_{jm}^{(Gini)}$$

If there are n trees in RF, then

$$VIM_j^{(Gini)} = \sum_{i=1}^{n} VIM_{ij}^{(Gini)}$$

Finally, we normalize all the importance scores we have obtained:

$$VIM_j = \frac{VIM_j}{\sum_{i=1}^{c} VIM_i}$$

As shown in Figure 8, we ranked features according to the importance score and showed the top 15 in it. By means of the rankings, we could find that encryption and decryption features are of the highest importance for Expdf, such as /DecodeParms and /Encrypt.

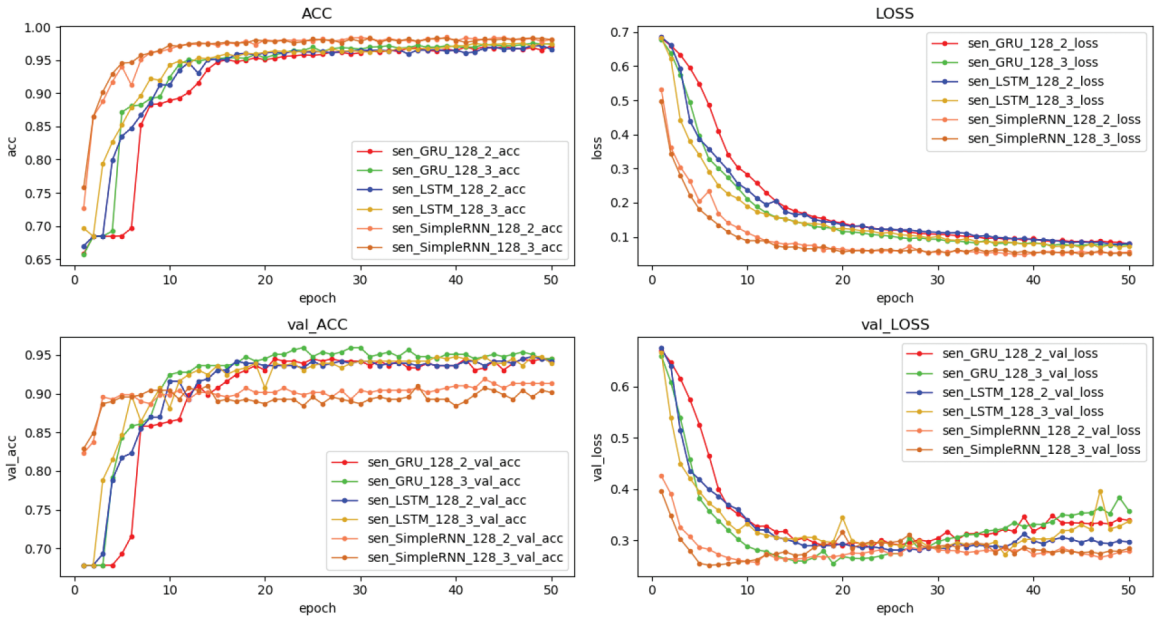

Figure 9 The convergence curve of accuracy and loss on the training set and the validation set during the training process of GRU, LSTM, SimpleRNN. The vertical axis represents the accuracy and loss rate while the horizontal axis represents epoch.

5.2.3. Model comparison

To further demonstrate the validity of the gene in Expdf, the neural network built as a comparative experiment is necessary. An important aspect of training large-scale neural network architectures is the training and testing efficiency. Figure 9 shows detailed experimental results for building GRU [24], LSTM [25], SimpleRNN as classification models, which contained four subfigures. Among the subfigures, it showed the convergence curve of accuracy and loss on the training set and the validation set during the training process of GRU, LSTM, SimpleRNN.

In this section, we implement the GRU, LSTM, and SimpleRNN neural network in Python using Theano together with Keras. We run experiments on a machine with NVIDIA GeForce GTX 1080 GPU. In our experiment, the dropout is set to 0.5, the minibatch stochastic gradient descent together with ADAMAX is used for training with the default learning rate of 1.0. The number of layers are set to 2 and 3. Meanwhile we trained and validated on 50 epochs.

In this work, we extracted the gene of processes using Expdf and generated gene vectors. Compared with GRU, LSTM, SimpleRNN, two layers and three layers the neural network models we trained and validated on 50 epochs. By means of Figure 9, the GRU network of three layers has more stable and better performance in the verification test. Meanwhile, more convincingly, it could be said that the genes of Expdf are excellent in any machine learning model.

6. CONCLUSIONS

In this paper, we introduced Expdf, exploits detection system capable of learning to discriminate between malware with exploitable code and without it. Different from existing research on detecting pdf malware, Expdf is the first robust recognition system for exploitable code-based machine learning. We transformed samples into the tree diagram, and extracted pdf exploitable malware gene based on the path of the leaf node and the corresponding value. For processing by machine learning models, we transformed the genes into numeric vectors. Based on the labels of VirusTotal, we collected samples as dataset. With the experimental evaluation compared to Hidost, Expdf demonstrates its superiority in detecting exploits. Compared with different machine learning models, Expdf proved the validity of gene extraction, independent of the specific type of model. It could be included that in most models, the genes of Expdf have an excellent performance in detecting exploits.

In order to ensure the accuracy of the samples, we have stricter screening of the samples. Based on this difficulty, the size of the data set is not very large. A conceptual design for this application was proposed in this paper. Future work can be developed for Expdf in parsing the JavaScript code extracted from the JS stream and feature fusion based on the large-scale dataset. Therefore, we have reasons to believe that, with the further development of machine learning and exploits parsing, detection and analysis of exploits will be more intelligent and powerful.

REFERENCES

[1] X. Zhou, J. Pang, G. Liang, Image classification for malware detection using extremely randomized trees, in IEEE International Conference on Anti-Counterfeiting, Security, and Identification, IEEE, Xiamen, 2017, pp. 54–59.
[2] P. Laskov, N. Šrndić, Static detection of malicious JavaScript-bearing PDF documents, in Twenty-Seventh Computer Security Applications Conference, Orlando, 2011, pp. 373–382.
[3] Symantec Corporation, Symantec internet security threat report, 2015. https://www.symantec.com/content/en/us/enterprise/other_resources/21347933_GA_RPT-internet-security-threat-report-volume-20-2015.pdf.
[4] C. Tankard, Advanced Persistent threats and how to monitor and deter them, Netw. Secur. 2011 (2011), 16–19.

[5] TrendLabsSM APT, Spear-phishing email: most favored APT attack bait, 2012. https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf.
[6] FireEye iSight Intelligence, APT28: at the center of the storm, special report 2016, 2016. http://www.fireeye.com/reports.html.
[7] C. Bing, Russia-linked hackers impersonate NATO in attempt to hack Romanian government, May 11, 2017. https://www.cyberscoop.com/dnc-hackers-impersonated-nato-attempt-hack-romanian-government/.
[8] GReAT, BlackOasis APT and and new targeted attacks leveraging zero-day exploit, 2017. https://securelist.com/blackoasis-apt-and-new-targeted-attacks-leveraging-zero-day-exploit/82732/.
[9] M. Sa, New Targeted attack in the Middle East by APT34, a suspected Iranian threat group, using CVE-2017-11882 exploit, 2017. https://www.fireeye.com/blog/threat-research/2017/12/targeted-attack-inmiddle-east-by-apt34.html.
[10] Cherepanov, A tale of two zero-days, May 15, 2018. https://www.welivesecurity.com/2018/05/15/tale-two-zero-days/.
[11] N. Šrndić, P. Laskov, Hidost: a static machine-learning-based detector of malicious files, EURASIP J. Inf. Secur. 1 (2016), 22.
[12] C. Smutz, A. Stavrou, Malicious PDF detection using metadata and structural features, in Proceedings of the 28th Annual Computer Security Applications Conference, ACM, Orlando, 2012, pp. 239–248.
[13] D. Liu, H. Wang, A. Stavrou, Detecting malicious javascript in pdf through document instrumentation, in Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on IEEE, Atlanta, 2014, pp. 100–111.
[14] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Lean. 3 (1988), 95–99.
[15] Virus Total, VirusTotal-Free online virus, malware and URL scanner. https://www.virustotal.com/en.
[16] J.M. Roberts, Virus share. https://virusshare.com.
[17] Document Management, Portable document format - Part 1: PDF 1.7. https://www.adobe.com/devnet/pdf/pdf_reference.html.
[18] H. Jin, Z. Rongcai, S. Zhen, *et al.*, Analyzing and recognizing android malware via semantic-based malware gene, in Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, IEEE, Nanjing, 2017, pp. 17–20.
[19] X. Zhou, J. Pang, F. Liu, *et al.*, Pdf Exploitable malware analysis based on exploit genes, in 2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), IEEE, Xiamen, 2018, pp. 16–20.
[20] V. Svetnik, A. Liaw, C. Tong, *et al.*, Random forest: a classification and regression tool for compound classification and QSAR modeling, J. Chem. Inf. Comput. Sci. 43 (2003), 1947–1958.
[21] P. Laskov, Practical evasion of a learning-based classifier: a case study, in IEEE Symposium on Security and Privacy, IEEE, San Jose, 2014, pp. 197–211.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011), 2825–2830.
[23] T. Fawcett, An introduction to ROC analysis, Pattern Recognit. Lett. 27 (2006), 861–874.
[24] J. Chung, C. Gulcehre, K.H. Cho, *et al.*, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv: 1412.3555, 2014.
[25] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997), 1735–1780.