# A novel detection technique based on benign samples and one-class algorithm for malicious PDF documents containing JavaScript

Jiaxiang Gu[a], Rui Kong[a], He Sun[b], Honglin Zhuang*[a], Fan Pan[b], Zhechao Lin[a]
[a]Academy of Military Sciences PLA China, Beijing, China; [b]Unit 96901, PLA China, Beijing, China

## ABSTRACT

PDF (Portable document format) documents are widely used in information publishing, academic exchanges and daily business. Phishing attacks with malicious PDF documents have become an important means of APT (Advanced Persistent Threat) organizations. Researchers have found that more than 90% of malicious PDF documents launch attacks by JavaScript code. The current detection models' generalization is not enough to detect unknown malicious samples. This paper proposes a method for detecting malicious PDF documents based on benign samples. The method uses benign PDF documents as training data, and uses features at the semantic level of JavaScript code. The JavaScript keywords and usage methods frequently used in malicious PDF documents are taken as important features to improve the robustness of the model. Then, we use One-class SVM (Support Vector Machine) machine learning algorithm to detect malicious PDF documents containing JavaScript code. Compared with the detection model trained with malicious PDF documents, the method proposed in this paper improves the generalization performance while maintaining a higher detection rate.

**Keywords:** PDF document, malicious code detection, JavaScript, one-class SVM

## 1. INTRODUCTION

In recent years, the public's awareness of network security has gradually increased, and they have been able to remain vigilant about executable programs, web pages and Office documents. But they tend to ignore the security issues of PDF documents that are often used in work. In fact, PDF documents have evolved from static pages to compound documents with scripts, multimedia content, interactive forms, and other functions. Malicious codes can be embedded in PDF documents. PDF documents are widely used. In 2020, 250 billion PDF documents have been processed by Adobe software[1]. PDF readers have a large number of vulnerabilities. Among them, the number of PDF vulnerabilities based on Adobe's Acrobat Reader accounts for 59.07% of the document type[2]. High-risk vulnerabilities are still being exposed. Many APT organizations have implemented phishing attacks through PDF documents. And the public literature database had also been contaminated by malicious PDF documents[3]. It is very important to strengthen the detection and protection of malicious codes in PDF documents.

A malicious PDF document refers to a PDF document containing malicious content. The malicious content can be malicious codes, files containing malicious codes, or phishing links. With the help of malicious PDF documents, attackers can achieve data theft and even arbitrary code execution. Conventional anti-virus software mostly uses signatures and simple heuristic rules to detect malicious PDF documents, but the detection rules are simple and lagging. This method is difficult to find unknown malicious PDF documents.

In recent years, the research on the detection of malicious PDF documents based on artificial intelligence has made great progress. The main method is to obtain PDF documents metadata, structural features, JavaScript codes and behavioural characteristics through the parser, and use machine learning or deep learning algorithms for detection. Among the existing malicious PDF documents detection methods, the detection models based on byte features have low accuracy, and the detection models based on JavaScript features are not comprehensive enough. Although the accuracy of the detection models based on structural features and metadata have been high, the detection models have low robustness[4] and insufficient detection capabilities for unknown malicious PDF documents[5].

*zhlxsjl@163.com.

This paper proposes a detection technology for malicious PDF documents containing JavaScript based on benign samples. This technology uses benign PDF documents as training samples, uses one-class SVM machine learning algorithm to establish a decision model. The main contributions are as follows:

- The detection model has strong generalization ability, and the detection rate of newly discovered malicious PDF documents reaches 96.91%.
- By using semantic-level features of JavaScript code, the interpretability of the detection model is improved.
- The detection model uses JavaScript keywords and usage methods frequently used in malicious PDF documents as components of features, making feature selection more targeted and improving robustness.

## 2. BACKGROUND

PDF documents support JavaScript codes execution, which provides conditions for attackers to use programming languages to implement flexible and diverse attacks. Its complex structure provides convenience for effective concealment of malicious code.

### 2.1 JavaScript codes in pdf document

Since PDF 1.3, in order to enrich the functions, PDF documents introduced JavaScript codes for interactive form, digital signatures, and display of 3D objects. The JavaScript codes can exist directly in the object, or it can be stored in the stream after being encoded, or it can be stored in an external file. The JavaScript interpreter integrated in the PDF reader can compile and execute JavaScript codes. The JavaScript codes in the PDF document can be executed automatically when it is opened, or it can be executed dynamically.

### 2.2 Attack method based on JavaScript code

There are three main attack methods for malicious PDF documents: JavaScript code-based attacks, embedded file-based attacks, and form submission attacks. Among them, JavaScript code-based attacks are the main method.

JavaScript code-based attacks can generally trigger vulnerabilities through incorrect JavaScript code API calls or data formats, or through embedded malicious files. Then these attacks use heap spraying technology to make the execution target jump to the memory address of the shellcode, and execute arbitrary malicious code through the shellcode[6]. In addition, the control process of other attacks can also be realized through JavaScript code. For example, downloading executable files from the Internet through JavaScript codes to launch attacks on user terminals.

### 2.3 Obfuscation of JavaScript codes in malicious pdf documents

In order to circumvent the detection of anti-virus software, the attackers take measures to obfuscate the JavaScript codes to increase the success rate of the attack. To obfuscate JavaScript codes in malicious PDF documents, conventional methods can be used, such as white space randomization, comment randomization, variable name randomization, string obfuscation, function name obfuscation, etc. It can also be combined with the PDF document structure to achieve targeted confusion. Attackers can split the malicious JavaScript codes through multi-level references, and encode the malicious codes in the stream. This method can change the original appearance of the malicious codes, making it impossible for the parser to obtain. Attackers also use methods such as file header confusion, object damage, and stream damage to cause PDF document parsing errors, making the parser unable to obtain JavaScript codes. However, malicious PDF documents can still be opened in reader, and retain malicious functions.

## 3. MOTIVATION

Among the existing malicious PDF documents detection methods, the detection accuracy of Hidost[7] and PDFRate[8] based on structural features has reached more than 99%. But the structural feature itself is not necessarily the essential feature of malicious code. Dey et al.[9] attacked the malicious PDF document detection model based on structural features, and realized the evasion of the detection model. Falah et al.[5] tested representative detection methods based on structural features. The results showed that these methods were highly dependent on data sets, and the generalization performance of the model was low.

More than 90% of malicious PDF documents achieve malicious behaviours through JavaScript codes[10]. If the detection model can accurately detect malicious PDF documents containing JavaScript code, it can greatly alleviate the harm of malicious PDF documents. Lemay et al.[11] analysed the content of JavaScript codes in malicious and benign PDF documents and found that there are obvious differences in the use of functions and keywords, which provides a basis for the use of JavaScript features to detect malicious PDF documents. Attacks based on JavaScript codes have certain similarities in the attack process, but the methods of triggering various vulnerabilities are not the same. Malicious JavaScript codes can also deform in a random way or by adding other code unrelated to the attack. Therefore, there are still big differences between malicious JavaScript codes. However, the functions implemented by using JavaScript code in benign PDF documents are relatively fixed, and the codes have greater similarities.

Based on the above considerations, this method attempts to use the JavaScript codes in the benign PDF documents as the training object and use one class algorithm to establish detection model. The main research objective is to achieve accurate detection of malicious PDF documents containing JavaScript code, and to make the model have high generalization performance.

## 4. DETECTION METHOD

### 4.1 The overall structure of the detection model

The overall structure of this method is mainly divided into three parts, as shown in Figure 1, one is preliminary detection, the second is feature extraction, and the third is classification. Since this method is based on the extracted JavaScript codes for detection, the prerequisite for successful detection is that the correct extraction of the JavaScript codes in the PDF documents can be completed. However, some attackers hide the JavaScript code so that the ordinary parser cannot extract it. This will invalidate the method proposed in this article. Therefore, it is first necessary to detect whether the PDF documents contain methods that prevents the ordinary parser from obtaining JavaScript codes. In the features extraction stage, the first task is to extract the JavaScript code, convert the JavaScript code into a token sequence. Then the token sequence is encoded by the N-gram model. In the classification stage, the one class SVM algorithm is used for classification.

### 4.2 JavaScript codes detection and extraction

Since JavaScript codes are not intentionally hidden in benign PDF documents. If hidden JavaScript code is detected, it is directly determined that the PDF document is a malicious PDF document.

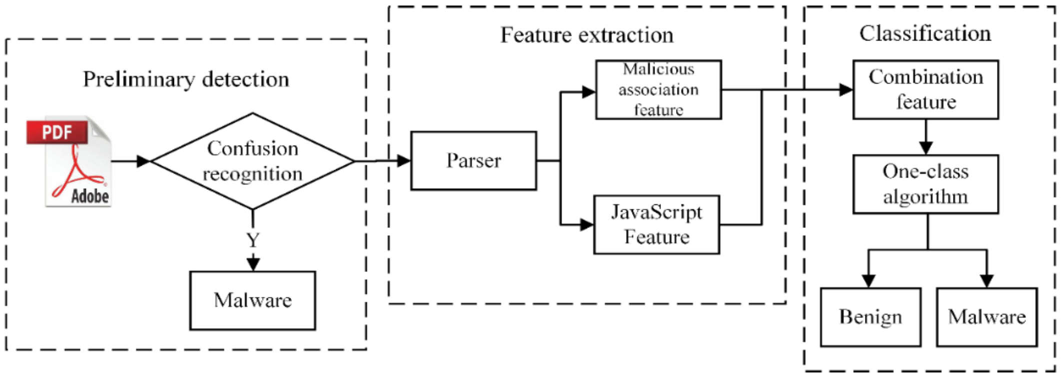

Figure 1. The overall structure of the detection model.

This method uses two parsers to test whether the PDF documents contains JavaScript codes. If the JavaScript codes in the PDF document is not hidden, the ordinary parser can identify it. If there are different results, it means that the PDF document uses hidden means. PJscan[10] and Peepdf[12] are used as comparative parsers. PJscan implemented the function of JavaScript codes extraction through the Poppler[13]. Peepdf is a PDF audit tool in Kali Linux. It can extract the JavaScript codes embedded in the PDF documents, and can also identify the hidden JavaScript codes in the PDF documents.

### 4.3 Feature structure

The features are constructed at the semantic level and consists of two parts, one part is a token sequence directly converted from all JavaScript codes. The other part is the tokens related to malicious behaviour. These tokens are designed based on JavaScript keywords and usage methods frequently used in malicious PDF documents. Then combine the two parts and encode them through the N-gram model. Spider Monkey[14] can convert JavaScript into a token sequence. It defines 86 tokens to represent keywords, operators, data, etc. in the JavaScript codes. The JavaScript codes can be converted into these tokens. For example:

```
function alloc(bytes) {
    padding.substr(0, (bytes - 6) / 2);
}
```

The above JavaScript code can be converted into the following tokens. The corresponding relationship is shown in Table 1.

34 29 27 29 28 25 48 29 22 29 27 30 3 27 29 16 30 28 18 30 28

Table 1. Correspondence between JavaScript code and tokens.

| JavaScript code | Token | Value | Description |
|---|---|---|---|
| function | TOK_FUNCTION | 34 | Function keyword |
| Alloc, bytes, padding, substr | TOK_NAME | 29 | Identifier |
| ( ) | TOK_LP, TOK_RP | 27, 28 | Left and right parentheses |
| { } | TOK_LC, TOK_RC | 25, 26 | Left and right curly bracket |
| . | TOK_DOT | 22 | Member operator (.) |
| 0 | TOK_NUMBER | 30 | Constant |
| , | TOK_COMMA | 3 | Comma operator |
| - | TOK_MINUS | 16 | Minus |
| / | TOK_DIVOP | 18 | Divide operator |

Malicious association feature are features closely related to malicious behaviours, which can be divided into two categories. One is malicious features related to obfuscating JavaScript codes, such as JavaScript codes execution function eval(), decoding and encoding string function unescape(), string manipulation functions substr(), substring(), replace(), keyword return, etc. The other is malicious features related to exploits, such as the length() function that can calculate the length of the buffer, and the long string that stores the shellcode, the number of two-digit number related to the existence of shellcode, etc. Compared with PJscan, we add more corresponding tokens based on malicious association features, as shown in Table 2.

Table 2. Malicious association tokens.

| Token | Description |
|---|---|
| TOK_EVAL | Eval keyword |
| TOK_UNESCAPE | Unescape keyword |
| TOK_SUBSTR | Substr keyword |
| TOK_REPLACE | Replace keyword |
| TOK_SUBSTRING | Substring keyword |
| TOK_RETURN | Return keyword |
| TOK_LENGTH | Length keyword |
| TOK_STR_LEN | A string of length > 100 |
| TOK_Number | Number of two-digit number > 20 |

### 4.4 One class SVM algorithm

This method uses the SVDD model (Support Vector Data Description) proposed by Tax[15] et al., which is a type of One Class SVM algorithm. It is usually used for anomaly detection and shows good results. In the training process, the data is first mapped to a new feature space through the kernel function. In the new feature space, a hypersphere that can contain as much data as possible is constructed. Through training the support vector machine, it calculates the center and radius of the hypersphere to minimize the radius of the hypersphere. In the decision-making process, the distance between the test sample feature and the center of the hypersphere is calculated. If the distance is less than the radius, it is judged as positive, and if the distance is greater than the radius, it is judged as negative. The optimization function is as follows:

$$\min_{R,a} R^2 + C\sum_{i=1}^{n}\zeta_i$$

$$s.t. \| x_i - a \|^2 \le R^2 + \zeta_i, \zeta_i \ge 0, i = 1, \dots, n$$

In the above formula, *C* is the penalty coefficient, used to control the tolerance of error, $\zeta$ is the relaxation variable, *a* is the center of the hypersphere, and *R* is the radius of the hypersphere.

### 4.5 Evaluation criteria

In order to comprehensively and accurately verify the effect of the method, four indicators, $TPR_{KN}$, $TPR_{UN}$, $FPR_{JS}$, and Accuracy are designed to evaluate the method.

- $TPR_{KN}$: True positive rate on known malware PDF documents. We regard the proportion of malicious PDF documents found four years ago detected as malicious PDF documents as the true positive rate on known malicious PDF documents.

- $TPR_{UN}$: True positive rate on unknown malware PDF documents. We regard the proportion of malicious PDF documents found in the past four years detected as malicious PDF documents as the true positive rate on unknown malicious PDF documents.

- $FPR_{JS}$: False Positive Rate on benign PDF documents containing JavaScript codes. The proportion of benign PDF documents that are classified as malicious PDF documents. These benign PDF documents contain JavaScript codes.

- Accuracy: Due to the large gap between the number of benign and malicious samples in the experimental data set, the value of Accuracy is the average of $TPR_{KN}$ and $TNR_{JS}$. $TNR_{JS}$ is the true negative rate on benign PDF documents containing JavaScript codes.

## 5. EXPERIMENTAL RESULT AND ANALYSIS

### 5.1 Data collection

The experimental data comes from Virus Total, Contagio[16] and Microsoft Bing search engine, the specific number is shown in Table 3. We downloaded malicious PDF documents for nearly four years from Virus Total, and obtained 11,106 malicious PDF documents and 9196 benign PDF documents from Contagio. We used the Bing search engine to search the entire Internet using common keywords, and collected 31,245 benign PDF documents. The benign PDF documents containing JavaScript codes were tested using Virus Total, and the results showed that none of them were malicious documents.

Table 3. Data set collection statistics.

| Source | Malware | Benign |
|---|---|---|
| Virus Total | 11090 | 0 |
| Contagio | 11106 | 9196 |
| Microsoft Bing | 0 | 31245 |
| Total | 22196 | 40441 |

### 5.2 JavaScript codes detection results

We used PeePDF and PJscan respectively to detect whether all the collected malicious and benign PDF documents contain JavaScript codes. The detection results of malicious PDF documents are shown in Table 4, and the detection results of benign PDF documents are shown in Table 5.

Table 4. Malicious PDF documents contain JavaScript code detection results.

| Source | Number | PeePDF | PJscan | Uncertain | Proportion | Detected as malicious |
|---|---|---|---|---|---|---|
| Virus Total | 11090 | 10295 | 8509 | 171 | 92.83% | 1786 |
| Contagio | 11106 | 10830 | 9347 | 68 | 97.51% | 1483 |
| Total | 22196 | 21125 | 17856 | 239 | 95.17% | 3269 |

Table 5. Benign PDF documents contain JavaScript code detection results.

| Source | Number | PeePDF | PJscan | Proportion |
|---|---|---|---|---|
| Contagio | 9196 | 392 | 392 | 4.26% |
| Microsoft Bing | 31245 | 235 | 235 | 0.75% |
| Total | 40441 | 627 | 627 | 1.55% |

The results show that this method can achieve the effect of preliminary screening. We detected a total of 3269 malicious PDF documents with hidden methods. In benign documents, the results of the two parsers were consistent, which also showed that no hidden methods are used in benign PDF documents. At the same time, it verified that the proportion of malicious PDF documents containing JavaScript code was indeed very high, reaching 95.17% in these two data sets. In benign PDF documents, the use of JavaScript codes was less. Since the benign PDF documents in Contagio is filtered, the proportion of JavaScript codes is higher. The benign PDF documents collected from Microsoft Bing can better reflect the true ratio. It was about 0.75%.

## 5.3 One class algorithm experimental results

The selection of parameters in the experimental algorithm was determined through experiments. The results are shown in Figures 2-4. Among them, N is the value of N in N-Gram, the horizontal axis represents the regularization parameter C of the SVDD algorithm, the vertical axis of Figure 2 represents the true positive rate, the vertical axis of Figure 3 represents the false positive rate, and the vertical axis of Figure 4 represents accuracy. After comprehensive comparison, N in N-Gram was selected as 4, and the regularization parameter C of SVDD algorithm was selected as 0.05.

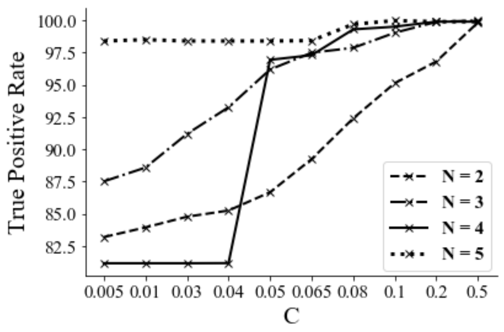

Figure 2. True positive rate experimental result.

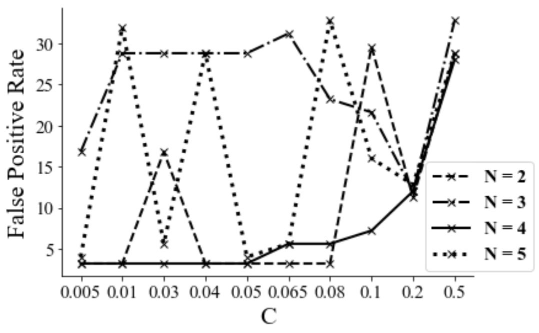

Figure 3. False positive rate experimental result.

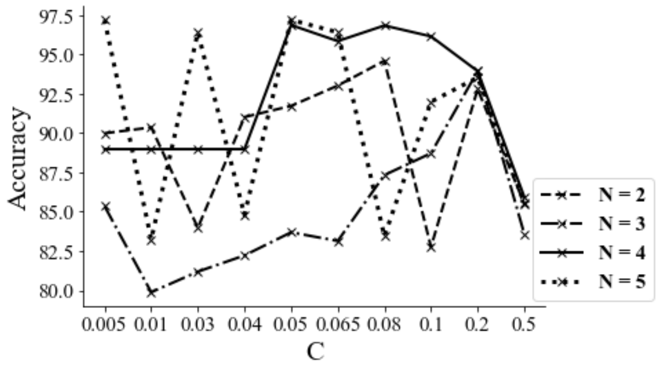

Figure 4. Accuracy experimental result.

In the experiment, after merging the two normal samples, we performed 5-fold Cross Validation 5 times. Since the training set of this method did not contain malicious samples, all malicious samples were directly used as the test set. The 10,830 malicious samples collected in Contagio were collected before 2016, and this part was regarded as a known test sample. The 10,295 malicious samples obtained in Virus Total in the past four years were regarded as unknown test samples. We selected PJscan and Hidost, which were open source and widely quoted, as the comparison objects. We followed the parameters of the original paper and used the data set used in this article for training and testing. The results are shown in Table 6.

Table 6. Comparison with the detection results of PJscan and Hidost.

| **Detection method** | **$TP_{KN}$** | **$TP_{UN}$** | **$FP_{JS}$** |
|---|---|---|---|
| Method of this article | 96.93% | 96.91% | 3.2% |
| PJscan | 80.23% | 74.06% | 1.28% |
| Hidost | 97.39% | 92.67% | 4.31% |

It can be seen from the results that PJscan has a low false positive rate, but its ability to detect malicious PDF documents is insufficient. Hidost has a high detection rate of known malicious PDF documents, reaching 97.39%, but its false positive rate is high. Both PJscan and Hidost's ability to detect unknown malicious PDF documents has decreased significantly. This shows that the detection model trained with existing malicious PDF documents has the problem of insufficient generalization ability. The method proposed in this paper has a detection rate of 96.93% for known malicious PDF documents and 96.91% for unknown malicious PDF documents. The results prove that the method has a high detection rate and strong generalization ability. Since there are fewer benign PDF documents containing JavaScript codes, the 3.2% false positive rate of this method is also acceptable.

## 5.4 Robustness test

The detection model in this paper is obtained by training using benign samples. In theory, attackers can perform an anti-imitation attack or an imitation attack on the model. Among the attacks against this model, the anti-imitation attack is to inject malicious JavaScript code into a benign PDF document containing JavaScript code, and the imitation attack is to inject benign JavaScript code into the malicious PDF document. If the original malicious PDF documents do not contain JavaScript codes, after injecting normal JavaScript codes, the detection model can be evaded. This is a defect of this method.

To verify the robustness of the detection model, we combined benign and malicious JavaScript code features to construct attack samples to attack the model. In order to increase the intensity of the attack, we selected benign samples far away from the decision boundary. We used the JavaScript codes features extracted from the benign samples to combine with the JavaScript codes features of 10,830 malicious samples collected in Contagio. This operation constructed ordinary malicious samples into attack samples. The attack samples were used to attack the common detection model and the detection model with malicious correlation features. The test results are shown in Table 7.

Table 7. Robustness test results.

| **Decision boundary radius** | **Average radius of benign samples** | **Evasion rate of common detection model** | **Evasion rate of detection models with malicious association features** |
|---|---|---|---|
| 0.949227 | 0.882454 | 50.05% | 42.37% |

The evasion rate of attack samples against common detection models is 50.05%, and the evasion rate against detection models that use malicious correlation features is 42.37%. It can be seen from the results that although the detection model with malicious association features still has the risk of being attacked, the evasion rate is reduced. The use of malicious correlation features improves the robustness of the detection model. The robustness of the detection model can be further improved by adding more malicious correlation features to the detection model and using more benign samples to train the model.

## 6. RELATED WORK

A large number of researches on the detection of malicious PDF documents based on artificial intelligence have emerged. The features used include PDF documents byte features, JavaScript features, metadata and structural features, and combinations of these features. Byte feature was an earlier feature used to solve the problem of malicious PDF documents detection. The main advantage of using byte feature was that it did not require human design features, the extraction process was simple, but the model was not interpretable, and the detection accuracy was not high. In order to detect malicious PDF documents more comprehensively, based on the consideration of the structural differences between malicious PDF documents and benign documents, researchers began to use metadata and structural features to construct detection models, but the robustness and generalization were insufficient.

Early detection methods based on JavaScript features were mainly based on behavioral features. SHELLOS[17], MDScan[18] and Lux0R[19] adopted JavaScript behavioural features. SHELLOS extracted memory access sequence features. MDScan extracted code execution sequence features. Lux0R extracted API call features. The main advantage of this method was that it was robust and not easy to be evaded, but it relied on the execution of JavaScript and had low efficiency. The accuracy depends on whether malicious JavaScript was executed.

In recent years, JavaScript feature-based detection methods mainly used JavaScript static features. Feng et al.[20] used the N-gram model to represent JavaScript byte features. Since this method acquired features at the byte level, there are unexplainable problems. PJScan used one class SVM algorithm to detect malicious PDF documents. This method used malicious PDF documents as training data. The detection model relied on known malicious PDF documents and had insufficient detection capabilities for unknown malicious documents. Yu[21] et al. used a logistic regression model to detect malicious PDF documents by combining JavaScript features and structural features. Its detection accuracy was better, but it had not been tested on large-scale data, and its robustness had not been tested.

## 7. CONCLUSIONS AND FUTURE WORK

Due to the difference between the JavaScript codes in the malicious PDF documents and the normal PDF documents, we took the benign PDF document as the training object and used the one-class SVM algorithm to design the malicious PDF documents detection model. This method did not completely rely on prior knowledge, can improve the generalization ability of the detection model, and had high detection accuracy. In addition, we identified JavaScript keywords and usage methods frequently used in malicious PDF documents, and used them as important features to make features selection more targeted and improved the robustness of the detection model. Experiments showed that the method achieved the expected results and had a high detection rate for unknown malicious PDF documents. However, it is necessary to further increase the number of benign samples used to enrich the features and further improve the robustness of detection model.

In the future, the robustness of the detection model can be further improved by using comprehensive features. In terms of algorithm selection, the latest deep learning single classification algorithm can be used in this method to improve the ability of the detection model. Because the attacker can design malicious PDF documents based on the vulnerabilities of the parser, the detection model may not be able to extract features. It is also necessary to build a more complete parser to realize the identification of more obfuscation means.

## REFERENCES

[1] Adobe Fast Facts, (2020). https://www.adobe.com/about-adobe/fast-facts.html

[2] Development Trend of Vulnerability, NFSFOUCS TIANJI Lab, (2020).

[3] Nissim, N., Cohen, A., Wu, J., et al., "Scholarly digital libraries as a platform for malware distribution," Systems Approach to Cyber Security, 15, 107-128 (2017).

[4] Xu, W., Qi, Y. and Evans, D., "Automatically evading classifiers: A case study on PDF malware classifiers," Network and Distributed System Security Symp., San Diego, (2016).

[5] Falah, A., Pan, L., Abdelrazek, M., et al., "Identifying drawbacks in malicious PDF detectors," 4th Int. Conf. on Future Network Systems and Security, Paris, 128-139 (2018).

[6] Maiorca, D., Biggio, B. and Giacinto, G., "Towards adversarial malware detection: Lessons learned from PDF-based attacks," ACM Computing Surveys, 1, (2019).

[7] Srndic, N. and Laskov, P., "Detection of malicious PDF files based on hierarchical document structure Network and Distributed System Security Symp., San Diego, (2013).

[8] Smutz, C. and Stavrou, A., "When a tree falls: using diversity in ensemble classifiers to identify evasion in malware detectors Network and Distributed System Security Symp., San Diego, (2016).

[9] Dey, S., Kumar, A., Sawarkar, M., et al., "EvadePDF: Towards evading machine learning based PDF malware classifiers," 2nd ISEA Int. Conf. on Security and Privacy, Jaipur, 140-150 (2019).

[10] Laskov, P. and Šrndić, N., "Static detection of malicious JavaScript-bearing PDF documents," Proc. of the 27th Annual Computer Security Applications Conf., New York, 373-382 (2011).

[11] Lemay, A. and Leblanc, S. P., "A study of JavaScript in PDF malware," Int. Conf. on Malicious and Unwanted Software, IEEE Press, Piscataway, 13-22 (2018).

[12] PeePDF http://eternaltodo.com/tools/peepdf-pdf-analysis-tool.

[13] Poppler https://poppler.freedesktop.org.

[14] Spider Monkey https://spidermonkey.dev.

[15] Tax, D. M. and Duin, R. P., "Support vector data description," Machine Learning, 54, 45-66 (2004).

[16] Contagio http://contagiodump.blogspot.com.

[17] Snow, K. Z., Krishnan, S., Monrose, F., et al., "SHELLOS: enabling fast detection and forensic analysis of code injection attacks," Proc. of the 20th USENIX Conf. on Security, USENIX Association, San Francisco, 123-138 (2011).

[18] Tzermias, Z., Sykiotakis, G., Polychronakis, M., et al., "Combining static and dynamic analysis for the detection of malicious documents," Proc. of the 4th European Workshop on System Security, ACM Press, New York, (2011).

[19] Corona, I., Maiorca, D., Ariu, D., et al., "Lux0R: detection of malicious PDF-embedded JavaScript code through discriminant analysis of API references," Proc. of the Workshop on Artificial Intelligent and Security Workshop, ACM Press, New York, 47-57 (2014).

[20] Feng, D., Yu, M., Wang, Y., et al., "Detecting malicious PDF files using semi-supervised learning method," 5th Int. Conf. on Advanced Computer Science Applications and Technologies, 1-9 (2017).

[21] Yu, M., Jiang, J., Li, G., et al., "A unified malicious documents detection model based on two layers of abstraction," 5th IEEE Int. Conf. on Data Science and Systems, IEEE Press, Piscataway, 2317-2323 (2019).