Malware detection based on TF-(IDF&ICF) method

Bin Qin¹, Junpeng Zhang¹,²* and Hongyu Chen¹,²

¹Shenzhen University Information Center, Shenzhen, Guangdong, 518060, China
²School of Electronics and Information Engineering, Shenzhen University, Shenzhen, Guangdong, 518060, China
2070436054@email.szu.edu.cn

Abstract: As the level of information technology continues to increase, information security problems caused by malware are becoming more and more serious. An important method to detect malware is to analyze software behavior information, such as permissions, API call sequences and system calls. In this paper, API call sequences are used as the research object for malware detection, and the traditional feature extraction methods are not ideal for API call sequences. In this paper, a TF-(IDF&ICF) feature extraction method is proposed by mathematical analysis, which combines document and category level features. Experiments show that using the feature extractor proposed in this paper, followed by training, the performance is improved in four different machine learning models, and the F1 can reach 0.979, while the system response time is significantly reduced, which has good practical value.

1. Introduction

According to the Rising 2020 China Network Security Report, Rising's "Cloud Security" system intercepted 148 million virus samples and 352 million virus infections in 2020, an increase of 43.71% over the same period in 2019, and in the case of ransomware, it has shifted from attacks on ordinary users to attacks on medium and large governments, enterprises and institutions [¹]. Therefore, for malware fast and accurate identification becomes one of the current challenges.

Malware search algorithms can be divided into two categories, dynamic and static, depending on the information collected. In static analysis, the identification of malware is achieved by the analysis of software code and bytecode and instructions, based mainly on the assembly code and properties of executable files. Dynamic analysis algorithms work with already running programs or run them in a virtual environment, exposing what happens during the work: analyzing the behavior, code and data parts of their programs and monitoring the resource consumption [²]. In recent years, some research results on malware feature extraction by researchers are worth of studying, for example, the literature [³] describes the use of features extracted in pre-trained AlexNet and Inception-v3 deep neural networks on the Malimg malware software dataset with segment-based fractal texture analysis (SFTA) using images representing malware code obtained by combining features with features obtained using segment-based fractal texture analysis (SFTA) of images representing malware code, the accuracy for malware detection is up to 99.3%; literature [⁴] proposes mapping malware binaries onto images, followed by extracting features with strong differentiation ability using principal component analysis, and finally using a support vector machine model for malware detection and classification with a model accuracy of up to 99.8%; literature [⁵] proposes a space-filling curve mapping (SFCM) method to visualize the features of byte sequences, and then combined with Markov point diagram (MDP) method to visualize the double-gram features of bytes and their statistical information sequences as the
location information of pixels to achieve the effective extraction of malware features, and finally the support vector machine method is used to achieve the high-performance detection of malware. In summary, [3-5] feature extraction approach is feasible for malicious code analysis and can be effective for effective extraction of malware features. The exploratory work on feature extraction helps in malware behavior feature extraction, but there are still shortcomings that lead to some information loss in the behavior feature extraction process. For example, API sequences are used as malware behavior detection features without considering the impact of non-dimensional features in API sequences on the detection results. The main work of this paper achieves to improve the overall performance of malware detection by proposing a feature extractor called TF-(IDF&ICF) to effectively fuse the features of different dimensions of API sequences.

2. Related work

2.1 TF-(IDF&ICF) algorithm

This section is firstly a brief introduction for the existing classical feature extractors and then leads to the idea and derivation process of the feature extractor TF-(IDF&ICF) proposed in this paper.

Among the feature extraction algorithms, TF-IDF (term frequency-inverse document frequency) is one of the most classical algorithms [6], which consists of two parts, respectively, the word frequency $tf_{i,j}$ as shown in equation (1) and the inverse document frequency $idf_i$ as shown in equation (2). The main idea of the TF-IDF algorithm is that if a word appears more frequently in a particular document, but the number of documents it has appeared in is small, then the word is given good distinguishing power, and its expression is shown in equation (3):

$$
tf_{i,j} = \frac{n_{i,j}}{\sum_{i}n_{k,i}} \quad (1)
$$

$$
idf_i = \log_2(1 + \frac{|D|}{|\{d \in D : i \in d\}|}) \quad (2)
$$

$$
TF-IDF = tf_{i,j} \ast idf_i \quad (3)
$$

Where $tf_{i,j}$ is the word frequency, which is the ratio of the number of occurrences $n_{i,j}$ of a word $j$ in a document to the total number of words $\sum_{i}n_{k,i}$ in the current document, $idf_i$ is the inverse document frequency, which is the logarithm of the ratio of the total number of documents $|D|$ to the number of documents $|\{d \in D : i \in d\}|$ in which the current word has appeared, and '1' is the smoothening term. TF-IDF is an unsupervised feature extraction method and does not consider label information. To address the drawback that TF-IDF does not utilize labels, Wang et al. [7] proposed to introduce the inverse category frequency $icf_i$ into the term weighting scheme, and TF-ICF replaces $icf_i$ with $idf_i$, which will tend to give more weight to terms that have appeared in fewer categories than those that have appeared in fewer documents, and its expression is shown in equation (5):

$$
icf_i = \log_2(1 + \frac{|C|}{cf(t_i)}) \quad (4)
$$

$$
TF-ICF = tf_{i,j} \ast icf_i \quad (5)
$$

Where $tf_{i,j}$ is the word frequency, which has the same meaning as the word frequency in TF-IDF; $|C|$ is the total number of categories; $cf(t_i)$ is the number of categories in which $t_i$ occurs.

Above we briefly introduced TF-IDF and TF-ICF, below we illustrate the idea of TF-(IDF&ICF) by extending the derivation of TF-IDF in Dr. Jun Wu's "The Beauty of Mathematics" [8]. The first thing to understand is the concept of information entropy, which is an important concept in information the-
ory, and it refers to the expectation of the amount of information brought by all possible events, which is expressed as shown in equation (6):

\[ H(x) = -\sum_{x \in X} P(x) \log_{10}(P(x)) \]  

(6)

The concept of TF-IDF is the cross entropy from the probability distribution of keywords under specific conditions. The weight of each keyword \( \omega \) in a given document should reflect how much information the keyword provides for the current document, which means that the amount of information of each word can be used as its weight, as expressed in equation (7):

\[ I(\omega) = -P(\omega) \log_{10}(P(\omega)) = -\frac{TF(\omega)}{N} \log_{10}\left(\frac{TF(\omega)}{N}\right) = \frac{TF(\omega)}{N} \log_{10}\left(\frac{N}{TF(\omega)}\right) \]  

(7)

Where \( TF(\omega) \) is the word frequency of \( \omega \) in the whole corpus; \( \omega \) is the keyword in the document; and \( N \) is the size of the whole corpus. Since the probability value is independent of the constant, \( N \) can be omitted, and then equation (7) can be reduced to equation (8):

\[ I(\omega) = TF(\omega) \log_{10}\left(\frac{N}{TF(\omega)}\right) \]  

(8)

When there are two-word frequency \( TF(\omega) \) at the same time, one is a common word in a document, while the other is scattered in multiple documents, the weight of the two words is the same in terms of equation (8), in fact, we can intuitively conclude that the word that appears in only one document should be given more weight, so that equation (8) does not reflect the resolution of keywords.

If two ideal assumptions are made:

1) Each document is essentially the same length, all with \( M \) words, then we have \( M = N/D = \sum_{\omega} TF(\omega)/D \).

2) If once a keyword appears in a document, the contributions are all the same, then this word appears \( Q(\omega) \) times in all documents, i.e., \( Q(\omega) = TF(\omega)/D(\omega) \), then from equation (8) we can obtain equation (9):

\[ I(\omega) = TF(\omega) \log_{10}\left(\frac{MD}{Q(\omega)D(\omega)}\right) = TF(\omega)(\log_{10}\left(\frac{D}{D(\omega)}\right) + \log_{10}\left(\frac{M}{Q(\omega)}\right)) \]  

(9)

Where \( Q(\omega) \) is the word frequency of \( \omega \) in the current document, \( TF(\omega) \) is the word frequency of \( \omega \) in the whole corpus, \( D \) is the number of all documents, and \( D(\omega) \) is the number of documents in which \( \omega \) occurs. We know that the definition of TF-IDF is the product of the word frequency and the logarithm of the inverse document frequency, then we can write equation (9) as:

\[ TF - IDF(\omega) = I(\omega) - TF(\omega) \log_{10}\left(\frac{M}{Q(\omega)}\right) \]  

(10)

Equation (10) can clearly see the relationship between TF-IDF and information entropy, the greater the value of the current word \( \omega \) information, the greater the value of TF-IDF; when \( Q(\omega) \) is larger, that is, the higher the frequency of a word in a document, the greater the value of TF-IDF, which is a good solution to the problem of equation (8), and these conclusions are fully consistent with information theory. But this only makes consideration from the document level of the word, so we analyze that on the basis of equation (9), considering the linkage information between the feature word and the label will have a better feature extraction effect, and the specific expression is shown in equation (11):

\[ I'(\omega) = TF(\omega) \log_{10}\left(\frac{D}{D(\omega)} + \frac{C}{C(\omega)}\right) + TF(\omega) \log_{10}\left(\frac{M}{Q(\omega)}\right) \]  

(11)
Where \( C(\omega) \) is the number of categories in which \( \omega \) has appeared and \( C \) is the total number of categories. Equation (12) is obtained from equation (11):

\[
TF(\omega) \log_{10}\left(\frac{D}{D(\omega)} + \frac{C}{C(\omega)}\right) = I'(\omega) - TF(\omega) \log_{10}\left(\frac{M}{Q(\omega)}\right)
\]

(12)

From equation (12), we can see that the more informative a word \( \omega \) is, the larger \( TF(\omega) \log_{10}\left(\frac{D}{D(\omega)} + \frac{C}{C(\omega)}\right) \) is. And through analysis we can see that the size of \( TF(\omega) \log_{10}\left(\frac{D}{D(\omega)} + \frac{C}{C(\omega)}\right) \) is determined by the number of categories \( C(\omega) \) in which the current \( \omega \) has appeared together with the number of documents \( D(\omega) \) and from the number of times it appears in the current document \( Q(\omega) \), obviously the size of the feature weight for the keyword gives more relevant information, and theoretically it will have better feature extraction capability. We define the first half of equation (12) as TF-(IDF&ICF), then we have equation (13):

\[
TF - (IDF & ICF) = TF(\omega) \log_{10}\left(\frac{D}{D(\omega)} + \frac{C}{C(\omega)}\right)
\]

(13)

Considering that in general the number of documents is much larger than the number of categories, i.e., \( \frac{D}{D(\omega)} \gg \frac{C}{C(\omega)} \) in most cases, resulting in the inverse document frequency becoming dominant, a logarithmic smoothing operation is done for the inverse document frequency to obtain equation (14):

\[
TF - (IDF & ICF) = TF(\omega) \log_{10}\left(\ln\left(\frac{D}{D(\omega)} + \frac{C}{C(\omega)}\right)\right)
\]

(14)

Finally, we express TF-(IDF&ICF) in the same defined form of TF-IDF and TF-ICF to obtain equation (15):

\[
TF - (IDF & ICF) = tf_{i,j} \log_{10}\left(1 + \frac{|C|}{cf(t_i)} + \ln\left(\frac{|D|}{|\{d \in D : i \in d\}|}\right)\right)
\]

(15)

Where ‘1’ is the smoothing term, which is retained here, \( tf_{i,j} \) is the later optimization that solves the different document lengths on the original word frequency; \( cf(t_i) \) is the number of categories in which \( t_i \) appears; \( |C| \) is the total number of categories; \( |\{d \in D : i \in d\}| \) is the number of documents in which \( t_i \) has appeared; \( |D| \) is the total number of documents. From equation (15), it can be seen that the weight of a certain keyword is determined by both the inverse category frequency and the inverse document frequency, which will give a richer feature to different keywords and theoretically will make the keyword importance differentiation more explicit.

3. Experiment and evaluation

3.1 Experiment platform
All the experimental platform information in this paper as follows: Processor: Intel(R)Core(TM) i79700CPU@3.00GHz-3.0GHz; RAM: 16.0GB; System type: 64-bit OS; x64-based processor; System version: win10 professional; PyCharm: 2020.2.2 64-bit; Python: 3.8; Scikit-learn: 0.23.2.

3.2 Evaluation metrics
In this paper, F1 (reconciled mean) were selected as indicators for the evaluation of classification effectiveness, and the formulae (16-18) were calculated as follows:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

(16)
\[ \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \]  
\[ \text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]  

Where TP refers to the number of positive samples predicted to be positive by itself; FN refers to the number of positive samples predicted to be negative by itself; FP refers to the number of negative samples predicted to be positive by itself; TN is the number of negative samples predicted to be negative by itself.

3.3 Data construction and processing
The datasets in this paper are mainly derived from log files of malware and benign software runs. The malware is mainly collected from Malekal malware sample website and CILPKU08 and Henchiri-Dataset databases provided by Peking University, with 1561 malware in total. The benign software is mainly common applications derived from window system files, with a total of 1051. We put the software samples into the Cuckoo sandbox to run, then intercepted the program running within 2 minutes, and finally extracted the API call sequences in the running log as the behavioral characteristics of the software. When running in the sandbox, because some malware has the technique of using anti-virtualization, when it detects that the environment is not a normal running environment, it will terminate the operation, resulting in some API call sequences are very short, which is not a reference value, for such samples we will remove them, finally the remaining 1391 malware API call sequences, 1051 benign software API call sequences. In addition, since the API call sequences have a lot of redundant information, we refer to Lu et al.\cite{9} and perform the de-duplication of the API sequences to get the final dataset used for training.

3.4 Parameter Settings
To clearly see the effectiveness of the proposed feature extractors in this paper, all the parameters of the feature extractors are set as shown in Table 1.

| Feature Extractor | TF-IDF | TF-ICF | TF-(IDF&ICF) |
|-------------------|--------|--------|--------------|
| Ngram_range       | (1,2)  | (1,2)  | (1,2)        |
| Other variables   | Default| Default| Default      |

3.5 Comparison of experimental results
The models used in this paper are: 1)LR:(logistic regression); 2)SVC:(support vector classification); 3)DT:(decision tree classifier); 4)QDA:(quadratic discriminate analysis); 5)Light-BGM (Light Gradient Boosting Machine); 6) MLP:(Muti-Layer Perception).

In this paper, after feature extraction with different feature extractors, the Stacking model is used for training, and the primary model at the Stacking model uses Light-BGM and MLP, and the secondary model uses four different machine learning models. By conducting experiments on the pre-processed dataset, the following results are obtained, where the evaluation methods are the reconciled mean F1, and the time, and the specific results are compared as shown in Table 2.
Table 2. Comparison table of F1 results

| Training programs | LR   | SVC  | DT   | QDA  |
|-------------------|------|------|------|------|
| TF-IDF            | 0.972| 0.973| 0.972| 0.972|
| TF-ICF            | 0.974| 0.972| 0.975| 0.974|
| TF-(IDF&ICF)      | **0.979** | **0.976** | **0.977** | **0.976** |

The experimental results of verifying the effectiveness of our proposed feature extractor by building malware detection models using different feature extraction schemes on each of the machine learning models are represented in Table 2. The feature extraction is first performed by different feature extractors and then trained using the Stacking model. From Table 2, we can see that the TF-(IDF&ICF) feature extractor proposed in this paper performs well on all four different training schemes, with a 0.4% to 0.7% improvement in F1 compared to other feature extractors, which verifies the superiority of the TF-(IDF&ICF) feature extractor for feature extraction. In addition, for the detection system, we should not only focus on F1, but also on the system response time, so we compared the training time used, as shown in Table 3.

Table 3. Comparison table of training time

| Training programs | LR   | SVC  | DT   | QDA  |
|-------------------|------|------|------|------|
| TF-IDF            | 130.5s | 143s | 86.6s | 169.7s |
| TF-ICF            | 36.8s | 26.4s | 41.1s | 60.1s |
| TF-(IDF&ICF)      | **23.5s** | **16.3s** | **18.6s** | **33.8s** |

Where the evaluation criterion is Time, it can be seen from Table 3 that using TF-(IDF&ICF) as a feature extractor not only has a great advantage over TF-IDF, but also has a reduction of 29.6% to 54.7% compared to TF-ICF. To clearly see the advantage of TF-(IDF&ICF) for feature extraction system response, we visualized the training result as shown in Figure 1.

![Figure 1. Comparison of training time](image)

It is obvious from Figure 1 that the use of TF-(IDF&ICF) feature extractor contributes more to the system response time reduction of malware compared to the use of TF-IDF and TF-ICF feature extractors, verifying that TF-(IDF&ICF) feature extractor has better utility compared to traditional feature extractors.

4. Conclusion
Malware detection has become increasingly important in the face of the continuous outbreak of malware. The TF-(IDF & ICF) feature extractor proposed in this paper achieves an effective fusion of document-level and class-level API call sequence features, and then the extracted features are trained using the Stacking model. The experimental results show that the TF-(IDF & ICF) feature extractor is
of good practical value for improving the performance improvement of malware detection. Finally, since only API call sequences are considered as detection features in this paper and no other features are considered, the next research work will explore features that are more distinct from the nature of the software identified by the model to make the malware detection model perform better.

Acknowledgments
Fund project: "5G Indoor Spatial Penetration, Enhancement and Comprehensive Demonstration" Project No. 2020YFB1806405, Key R&D Program of Ministry of Science and Technology; Shenzhen University 2020 Graduate Education Reform Project (No.860-000001050503)

References
[1] Rising 2020 China Cyber security Report. (2021) Information Security Research, 7(02):102-109.
[2] Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR), 44(2), 1-42.
[3] Nisa, M., Shah, J. H., Kanwal, S., Raza, M., Khan, M. A., Damaševičius, R., & Blažauskas, T. (2020). Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. Applied Sciences, 10(14), 4966.
[4] Ghouti, L., & Imam, M. (2020). Malware classification using compact image features and multiclass support vector machines. IET Information Security, 14(4), 419-429.
[5] Ren, Z., Chen, G., & Lu, W. (2019). Malware visualization methods based on deep convolution neural networks. Multimedia Tools and Applications, 1-19.
[6] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5), 513-523.
[7] Wang, D., & Zhang, H. (2010). Inverse-category-frequency based supervised term weighting scheme for text categorization. arXiv preprint arXiv:1012.2609.
[8] Wu J. (2012) The beauty of mathematics. People's Post and Telecommunications Publishing House, Beijing.
[9] Lu X F. (2019)"ASSCA API sequence and statistics features combined architecture for malware detection." Computer Networks,157: 99-111.