Research on malicious Web Page Detection algorithm based on Deep Learning

Ruoyang Tan [1], Shuhao Zhang[2], Sibo Zhang[3]

[1] School of Mathematics and Statistics, Chongqing Technology and Business University, Chongqing, China

[2] Academy of Electronics and Information Engineering, Tiangong University, Tianjin, China

[3] College of Computer Science, North China Institute of Aerospace Engineering, Langfang, Hebei, China

Keywords: Deep learning; malicious Web content; web page classification; malicious web page recognition.

Abstract: In recent years, malicious web page detection mainly relies on semantic analysis or code simulation execution to extract features, but these methods are complex to implement, require high computational overhead and increase the attack surface. For this reason, a malicious web page detection method based on deep learning is proposed. Firstly, simple regular expressions are used to extract semantically independent tags directly from static HTML documents. Then the neural network model is used to capture the local representation of the document on multiple hierarchical spatial scales, and the ability to quickly find small malicious code fragments from web pages of arbitrary length is realized. this method is compared with a variety of baseline models and simplified models, and the results show that this method achieves a detection rate of 96.4% with a false alarm rate of 0.1%. Better classification accuracy is obtained. the speed and accuracy of this method make it suitable for deployment to endpoints, firewalls and Web agents.

1. Introduction

At present, attacks using malicious web pages are very common. the security report points out that in the first half of 2018, the Internet Security Center intercepted a total of 16.226 million new malicious web pages, a seven-fold increase compared with the first half of 2017 (2.015 million)[1]. How to effectively identify these malicious web pages is faced with a variety of challenges: first of all, the detection method must be run without the user's perception. Secondly, the detection method must be able to identify all kinds of code obfuscation techniques for the purpose of avoiding detection; finally, the detection method must be able to quickly find out the malicious code fragments embedded in the normal web page in the massive Web web page visits[2].

In order to meet these challenges, this paper proposes a fast detection method of malicious web pages based on deep learning, which uses simple 12-character regular expressions to mark Web content, and then detects these contents on multiple hierarchical spatial scales. hierarchical spatial scale refers to not simply taking the token (token) set on the entire HTML file as input. Instead, it calculates the representation of the token set in multiple local specific subregions: the HTML file is divided into 1max 2, 1max 4, 1max 8 and 1max 16, and then two full connection layers (fully connected layer) are used to extract the representation of the HTML file at these levels. this method is based on simple token flow input to learn high-quality representation of Web content. Small malicious code snippets can be quickly found in web pages of any length.

One of the focuses of malicious web page detection is to use only URL strings to detect malicious Web content. Literature [3] based on blacklist method, first label malicious URL, and then use string matching and other techniques to identify malicious URL. Literature [4] establishes a unified classification model based on URL lexical features and host features, and then identifies malicious URL, according to existing tagging sets, which focus on manual tagging to maximize detection accuracy; reference [5] uses URL as the detection basis, but also contains other information, such as URL quotes in Web links, which manually extract features as the input of SVM and K-nearest classifiers. However, these methods only focus on URL-related information, so they can not take

advantage of malicious semantics in Web content. Although URL-based systems have the advantage of lightweight and can be deployed in an environment without complete Web content, this paper focuses on HTML files because they can detect deeper threats.

Literature [6mur8] attempts to extract features from HTML and JavaScript and provide them to machine learning or heuristic detection systems to detect malicious Web content. Literature [6] proposes a web page malicious code detection method based on machine learning classifier to detect web page malicious JavaScript scripts by analyzing the characteristics of JavaScript scripts. Reference [7] proposes a Web page crawler for JavaScript anti-confusion and analysis, then all extracted JavaScript code is represented by custom basic words, and then three machine learning algorithms are used to detect malicious web pages through anomaly detection model. Reference [8] proposes a manually defined heuristic method that uses static feature extraction to detect malicious HTML documents. These methods are similar to those studied in this paper, except that the parser-free tokenization method is used to calculate the representation of HTML files instead of explicitly parsing HTML, JavaScript or CSS. Parser-free representation of web content even simulates the execution of JavaScript. Parser-free representation of web content allows for minimal assumptions about the syntax and semantics of malicious and benign documents, so that the deep learning model has maximum flexibility in learning the internal representation of Web content.

In addition to Web content detection, scholars at home and abroad have done extensive research in the field of text classification based on deep learning. For example, the single-layer convolution neural network (CNN), proposed in reference [9] uses unsupervised (Word2Vec) sequences and words embedded in (word embedding) sequences, which can provide good performance in sentence classification tasks. The hyperparameters needed for text classification using single-layer CNN are adjusted in reference [10] to further optimize the performance of the model, while the model in reference [11] shows that the CNN method of learning representation directly from character input is more competitive in the problem of text classification. The work of this paper involves these methods, but the difference is that this model runs on HTML web pages with multiple formats of HTML, JavaScript and CSS, which may contain arbitrary source code, attack load and natural language. because the uncertainty of the content of HTML files makes it relatively difficult to define discrete words, this paper does not use word embedding as model input. Instead, a simple, format-independent tagging method is used to represent Web documents hierarchically.

2. Model design

2.1 Design principle

The model proposed in this paper is based on the following particularities of malicious web page detection:

(1) malicious content fragments are usually very small, but the length of HTML documents varies greatly, which means that the length ratio of malicious content in HTML documents is variable. Therefore, the model is required to check documents on multiple spatial scales.

To identify whether a given document is malicious. (2) explicit parsing of HTML documents is actually the execution of HTML, JavaScript, CSS and data, which is not desirable because it may require high computational overhead and open the attack surface within the detector, which may be exploited by attackers.

Simulation execution, static analysis or symbolic execution of JavaScript is also not desirable. Because it is also possible to increase computational overhead and open the attack surface within the detector.

Based on the above problems, the model design follows the following principles:

(1) the model does not perform detailed parsing, static analysis, symbol execution or content simulation on HTML documents, but makes minimal assumptions on the document composition format and makes simple word bag style tags, which is called token bag (Bag of Token in this paper.

BoT).

instead of simply representing the whole document as a single-layer token bag, the model captures the locality representation on multiple scale spaces representing different regions and aggregation levels.

2.2 Design method

The model involves a feature extractor responsible for parsing a series of token (token) from HTML documents. A neural network model that consists of two logical components.

inspectors: apply shared weights (shared-weight) on a hierarchical spatial scale and aggregate the information of the document into 1024-length vectors.

main network: make the final classification decision on the inspector's output.

feature extractor: First, the target document is marked with a regular expression ([^\ X00 -\ x7F] + |\ w+) +, which divides the document along the boundary of non-alphanumeric words. Then the resulting token stream is divided into 16 consecutive blocks of equal length, where the length is defined as the number of tokens. If the number of tokens in the document is not divisible by 16, there are fewer tokens in the last block.Python Code function implemented by tagging and Block function system. The number of TokenizeChunk is as follows.

```
import numpy as np
def TokenizeChunk(data, steps=16, dims=16 384):
data=TokenizeLengthHash (data, steps=steps,
dims=dims)
ret = []
stepsize = int(len(data) / float(steps))
for percent in np.arange(0, 1, 1 / float(steps)):
idx = int(len(data) * percent)
unq, cnt = np.unique(data[idx:idx + stepsize],
return_counts=True)
newarray = np.zeros(dims / steps)
for v, c in zip(unq, cnt):
newarray[v] = c
ret.append(newarray)
return ret
```

Next, use the function TokenizeLengthHash, which uses a modified hashing trick, to create a token bag for each block using 1024 bin.

```
import re
import murmur # the murmur hashing library
def TokenizeLengthHash (data, steps=16, dims=16 384):
feats = re.findall(r"([^\x00-\x7F]+|\w+)", data)
final_feats = []
for feat in feats:
loglength=int(min(8, max(1, math.log(len(feat),
1.4)))) - 1 # 0-7
shash=murmur.string_hash (feat) % (dims/steps/8)
final_feats.append (loglength*(dims/steps/8)+shash)
return final_feats
```

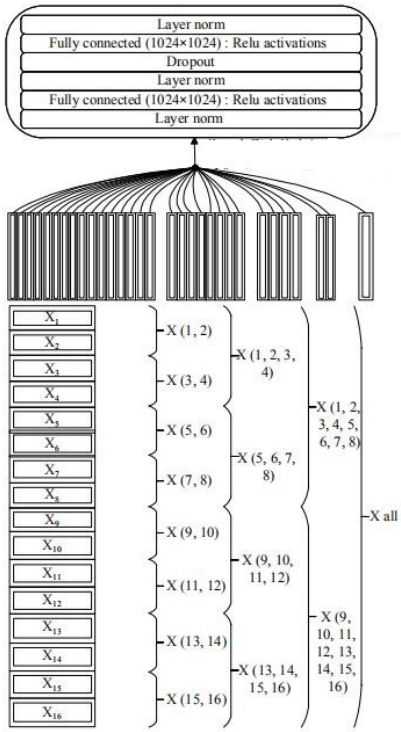


Fig 1 Inspector component logic

The result of the whole process is to first mark the HTML document, then divide the resulting token set into 16 blocks of equal length, and then hash each block into 1024 bin, resulting in a 16 × 1024 tensor, table showing the token bag sequence extracted from the HTML document, where each element in the sequence represents the aggregation on the successive 1max 16 of the input document.

Inspector component: the inspector is responsible for inputting the extracted feature representation into the neural network. as shown in figure 1, the first step is to create a hierarchical token sequence, in which the first 16 token bags are folded into 8 token bags, then the 8 token bags are folded into 4 and 4 into 2. 2 fold into 1. In this way, multiple token bags are obtained on multiple spatial scales to capture the appearance of tokens. The average window length of the whole folding process is 2 and the step size is 2 until a token bag is recursively obtained. by using the method of averaging rather than summation, the norm of each representation level is guaranteed for a given document.

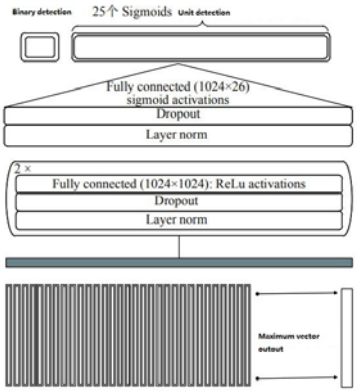


Fig 2 Primary network component logic

Once the inspector has created this hierarchical representation, it continues to access each node in the aggregation tree and calculates the output vector. The inspector is a feedforward neural network

with two fully connected layers, each with 1024 ReLU units. This paper uses layer normalization [12] to prevent gradient dispersion, and uses dropout [13] to normalize inspectors, dropout=0.2.

After the inspector visits each node, in order to calculate the 1024-dimensional vector output, the model obtains the maximum activation from 1024 neurons, and the result is that the final vector of the document is represented by the maximum output of each neuron in the last layer. in short, this enhances the output vector capture patterns, because these patterns best match the known template features used to predict malicious content.

No matter where they appear in the document, or how long the entire document is.

main network component: once the inspector calculates the 1024-dimensional output vector of the target document, the vector is input into the main network of the model. As shown in figure 2, the main network is also a feedforward neural network with two fully connected layers, before each fully connected layer is layer normali zation and dropout, like the inspector's, dropout=0.2.

The last layer of the model consists of 26 Sigmoid units, corresponding to 26 detection decisions made on the document. One Sigmoid is used to determine whether the target document is malicious or benign, and the remaining 25 Sigmoid detect various information tags to determine the malware family of the document, such as determining whether the document is a phishing page or a vulnerability exploitation toolkit. To train the model. In this paper, we use the binary-cross-entropy loss function on each Sigmoid output, and then average the gradient to calculate the parameter update. The model emphasizes the accuracy of evaluating good and bad Sigmoid output, but also takes into account the performance of the model output.

3. Experiment

The vibration and noise tests of the sample vehicle are carried out under the condition of acceleration and uniform speed, and the vibration and noise characteristics of the electric vehicle are analyzed, which provides a basis for the improvement of the vibration and noise of the electric vehicle in the next step.

The goal of this paper is to create a fast Web content detection model that can run on endpoints, firewalls and Web agents. The accuracy and speed of the model are the main assessment indicators.

In this paper, the accuracy of the above model is tested in two ways. first, it is compared with other word bag models, which represent standard document classification methods, and then modify the model in various ways to test the rationality of the model design.

This paper does not directly give the speed of the experimental use case test model. The hierarchical scale space model involves the parsing of web page content, but compared with other methods involving Web content parsing or simulation execution, the detection time of 400000 web pages is only 30% of that of this kind of WAF products, and the speed advantage is obvious.

The experimental data set comes from the company's NGSOC and TIP platforms. The company's globally controlled probes receive tens of thousands of new HTML files every day and scan them using 60 Web threat scanners. The experimental data set is the data pushed by the platform in the first nine months of 2018, as shown in figure 3.

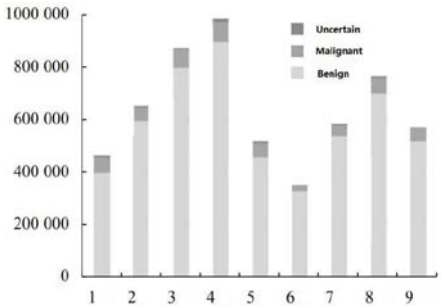


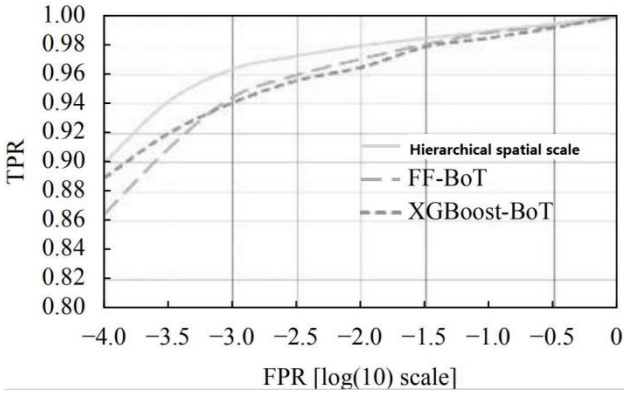
Fig 3 Experimental data set

HTML files are uniquely identified based on SHA256. The split of the training / test set is calculated according to the first push time of the file. This ensures that: first, the training and test sets are different (because the same HTML file submitted later will be interpreted as re-pushing the file and will choose to ignore it); second, the training and test experiments are closer to the actual deployment scenario.

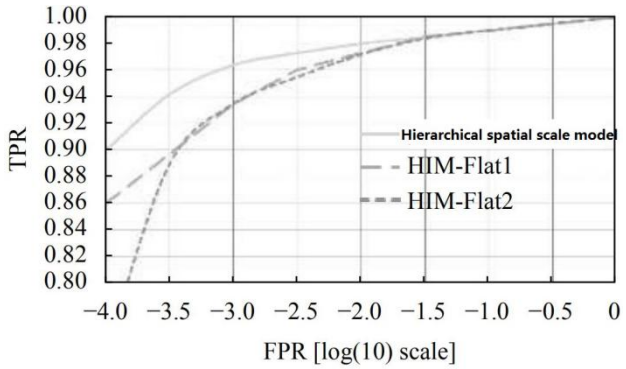
4. Experimental results

Figure 4 (a) and figure 4 (b) show the ROC curve of the experimental results. The y axis represents the true positive rate of TPR and the x axis represents the false positive rate of FPR.

Figure 4 (a) compares FF-BoT and XGBoost-BoT with this model, and figure 4 (b) compares the modified model with the complete model.



(a)Comparison between hierarchical Spatial scale Model and word bag baseline Model



(b)Comparison between hierarchical spatial scale model and modified model

Fig 4 Comparison diagram of experimental ROC curve

The ROC curve of figure 4 (a) shows that this model is superior to other baseline models. Comparing the relative performance of these models with 0.1% FPR, the detection rates of this model, FF-BoT and XGBoost-BoT are 96.4%, 94.5% and 94.1%, respectively. According to FPR, the false positive rate of this model is 3.6%.While FF-BoT and XGBoost-BoT are about 5.5% and 5.9%. Overall, the overall ROC curve of the hierarchical scale space model is significantly better than that of FF-BoT and XGBoost-BoT.

In addition, the FF-BoT parameter (about 20 million) far exceeds that of the proposed model (about 4 million), which shows that the proposed model captures a more effective feature representation of malicious HTML documents, thanks to the fact that the inspector uses the same parameters in each spatial context checked in the hierarchical representation.

Fig. 4 the ROC curve of (b) shows that under the premise of 0.1% FPR, the detection rate of HIM-Flat1 and HIM-Flat2 variants is 93.5%.

The ROC curve of the variant model is obviously worse than that of the model in this paper. and HIM.

The comparative experiments of Flat1 show that checking content on multiple spatial scales is very important to obtain good accuracy. Similarly, the comparison with HIM-Flat2 shows that it is important for inspectors to use the same parameters when checking each spatial context and scale, because HIM-Flat2 uses separate weights for each spatial context to get worse results.

In order to better understand what the hierarchical scale space model has learned, this paper uses malware family tags to subdivide malicious samples. Based on the global false alarm rate threshold of 0.1%, the overall detection rate of the hierarchical scale space model is 96.4%, and the checked files are classified according to the preset malware family. The experimental results show that the model achieves ideal detection results in code injection XSS, browser vulnerability exploitation and iFrame hijacking, but the detection rate on phishing sites is not high. Because a malicious file may belong to multiple families at the same time, the sum of the percentages in the table should be more than 100%.

In addition, this paper also examines the inconsistency between the model and the company label, and proves that the model has actually detected unknown malicious Web content missed by 360. to carry out this analysis, this paper examines the top 20 scoring test examples marked as benign verification sets, and finds that 11 of the 20 are actually malicious or potentially threatening.

Nine are false positives. Three of the malicious files are alert pages from the web content blocker, which are not malicious but indicate malicious warnings. Three are malicious camouflage JQuery libraries, one is a Javascript, that releases svchost.exe Trojans to disk, one is a page click hijacking, one is a page containing downloader execution code, and two are spam emails. the analysis verifies that the model can surpass the tag noise and identify unknown malicious content.

5. Conclusion

Unlike most deep learning-based document classification (such as emotion classification), this paper tries to capture malicious behaviors that try to evade detection, while the authors of emotion classification sentences do not hide their emotions. in addition, this paper avoids using the original character sequence as the input of the model. Because the length of HTML documents makes convolution neural networks involving original strings difficult to deal with on endpoints and firewalls, this model adopts a purely token-based static detection method, which avoids the need for complex parsing or simulation systems, and can effectively deal with the detection of Web pages, regardless of the size of Web pages. 96.4% detection performance is achieved with a false alarm rate of 0.1%. It can even identify malicious Web content that has not been captured by security vendors before. the good speed and accuracy of the model make it suitable for deployment to endpoints, firewalls and Web agents.

References

[1] The Security report of the China-China Internet Network in the first half of 2018. Http://zt.360.cn/1101061855.php?dtid=1101062360&did=491357630. [2018-07 Mutual 30].

[2] Sha Hongzhou, Liu Qingyun, Liu Tingwen, et al. A review of research on malicious web page identification. Journal of computer Science, 2016,39 (3): 529Mel 542. [doi: 10.11897/SP.J.1016.2016.00529]

[3] Akiyama M, Yagi T, Itoh M. Searching structuralneighborhood of malicious urls t o improve blacklisting.Proceeding of the 2011 IEEE/IPSJ International Symposiumon Applicatio ns and the Internet. Munich, Bavaria, Germany,2011: 1 – 10.

[4] Ma J , Saul L K , Savage S , et al. Learning to detect malicious URLs[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011.

[5] Choi H, Zhu BB, Lee H. Detecting malicious web links andidentifying their attack types. WebApps’ 11 Proceedings ofthe 2nd USENIX Conference on Web ApplicationDevelopment. Berkeley, CA, USA. 2011. 11.

[6] Li Yang, Liu Biao, Feng Huamin. Malicious code detection method for web pages based on machine learning. Journal of Beijing Institute of Electronic Science and Technology, 2012,20 (4): 36mur40, 12.

[7] Ma Hongliang, Wang Wei, Han Zhen. Anomaly detection of lightweight malicious web pages based on JavaScript. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2014

[8] Seifert C, Welch I, Komisarczuk P. Identification ofmalicious web pages with static heuristics. 2008 AustralasianTelecommunication Networks and Applications Conference.Adelaide, SA, Australia. 2008. 91–96.

[9] Kim Y. Convolutional neural networks for sentenceclassification. arXiv preprint arXiv: 1408: 5882, 2014.

[10] Zhang Y, Wallace B. A sensitivity analysis of (andpractitioners’ guide to) convolutional neural networks forsentence classification. arXiv: 1510: 03820, 2015.

[11] Zhang X, Zhao JB, LeCun Y. Character-level convolutionalnetworks for text classification. arXiv: 1509.01626, 2015.

[12] Ba JL, Kiros JR, Hinton GE. Layer normalization. arXivpreprint arXiv: 1607.06450, 2016.

[13] Ma Hongliang, Wang Wei, Han Zhen. Anomaly detection of lightweight malicious web pages based on JavaScript. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2014