A Framework for Intelligent Generation of Intrusion Detection Rules Based on Grad-CAM

Xingyu Wang[1,2], Huaifeng Bao[1,2], Wenhao Li[1,2], Haoning Chen[1,2], Wen Wang[1,2](✉), and Feng Liu[1,2]

[1] Institute of Information Engineering, CAS, Beijing, China
[2] School of Cyber Security, University of CAS, Beijing, China
{wangxingyu, baohuaifeng, liwenhao, chenhaoning, wangwen, liufeng}@iie.ac.cn

Abstract. Intrusion detection systems (IDS) play a critical role in protecting networks from cyber threats. Currently, intrusion detection methods based on artificial intelligen(AI) stand as the mainstream, yet they grapple with the challenges of interpretability and high computational costs. Conversely, rule-based approaches offer ease of comprehension and lower computational overhead, but their development demands extensive expertise. This paper proposes an intelligent framework for generating intrusion detection rules, which integrates the strong representational capabilities of AI detectors while retaining the advantages of rule-based detection. Initially, the framework involves training a TextCNN model for traffic payload classification. The parameters of this model, along with the Gradient-weighted Class Activation Mapping (Grad-CAM) algorithm, are employed to analyze critical fields in captured traffic payloads. Subsequently, a comprehensive list of keywords is obtained through a sensitive words aggregation algorithm, and regular expressions are generated to describe the detection content. These regular expressions undergo fine-tuning to reduce their false positive rate. Furthermore, adhering to the syntax of Suricata rules, they are formulated into intrusion detection rules. The proposed method is evaluated on two publicly available datasets, with experimental results demonstrating commendable detection efficacy for the generated intrusion detection rules.

Keywords: Intrusion detection · grad-cam · rule generation · traffic classification

1 Introduction

The development of computer networks has revolutionized the way people communicate, collaborate, and access information. Concurrently, security issues associated with computer networks have increasingly become a focal point of concern [8]. By scrutinizing network traffic, intrusion detection systems can prevent potential intrusions and ensure the confidentiality, integrity, and availability of the network [1]. In recent years, AI-based intrusion detection methods have proliferated [9]. However, traffic features often fluctuate with changes in the network environment, potentially leading to a high number of false positives [4]. Additionally, the lack of interpretability in artificial intelligence(AI) poses another challenge to the widespread application [10]. This is due to the fact that their output is limited to positive or negative discriminations, and the accuracy of which is difficult to assess effectively. Moreover, neural network models typically incur substantial computational costs. By scrutinizing network traffic, Intrusion detection systems can prevent potential intrusions and ensure the confidentiality, integrity, and availability of the network [1]. In comparison, traditional rule-based methods possess the following advantages:

– Low False Positive Rate: Rule-based intrusion detection systems generally demonstrate lower false positive rates due to their focus on detecting the fundamental payloads of attack behaviors.
– Interpretability: Rule-based intrusion detection systems enable explanations and a better understanding of intrusive behaviors. These rules can be reviewed and validated, making the system's behavior more predictable [6].
– Low Computational Resource Requirement: Rule-based intrusion detection systems demand less computational resources and boast easier deployment when contrasted with machine learning-based counterparts.

These advantages effectively mitigate the limitations of AI. While rule-based intrusion detection possesses the aforementioned merits, it still faces some constraints. Firstly the existing production process of malicious traffic detection rules is inefficient and human resource-consuming. This is because the majority of them are manually extracted by security experts from malicious samples, exploit codes or malicious traffic based on their expertise and experience. Secondly, current manual rule generation methods struggle to promptly generate rules for newly emerging malicious traffic, leading to delayed updates that create opportunities for 1-day attacks. There are existing studies focusing on intelligent intrusion detection rule generation, such as [13,15]. However, existing studies have primarily focused on fields within threat intelligence such as IP and domain. Additionally, challenges arise due to the difficulty of deployment, attributed to the constraints imposed by the syntax of the Suricata rules.

This paper proposes an intelligent intrusion detection rule generation framework. The method begins by training a TextCNN model for traffic payload classification. The Gradient Weighted Class Activation Mapping (Grad-CAM) [18] algorithm is then used to analyze the keywords in the malicious traffic payload from the sample inputs and the well-trained model parameters. A sensitive words aggregation algorithm is devised to obtain a more comprehensive list of keywords. Subsequently, regular expressions are formulated for detection, and through fine-tuning operations, they are further refined to reduce the false positives. Finally, intrusion detection rules are written based on the syntax of Suricata rules [2]. This method formulates rules based on the payload of the attack, specifically targeting the core of the attack behavior. It not only reduces manual dependence but also enhances the interpretability of intrusion detection by focusing on the key aspects of attack payloads.

The main contributions of this paper are as follows:

- We propose an intelligent framework for generating intrusion detection rules based on Grad-CAM. This framework leverages a pre-trained TextCNN model to identify keywords within malicious traffic payloads.
- To achieve more efficient intrusion detection, we introduce a Sensitive Words Aggregation Algorithm aimed at obtaining a more comprehensive list of keywords.
- We have conducted thorough experiments on two publicly available datasets. The results indicate that our method consistently outperforms baselines, demonstrating superior effectiveness in intrusion detection.

The rest of this paper is organized as follows. In Section 2, we provide a brief overview of existing research on intrusion detection systems and intelligent rule generation. Section 3 presents a detailed description of the implementation details of the intelligent rule generation method proposed in this paper. In Section 4, comprehensive experiments are conducted to validate the effectiveness of the proposed method. Finally, we conclude our work in the last section.

2 Related Work

2.1 Intrusion Detection System

In the realm of network attack detection, intrusion detection plays a pivotal role, and this domain has witnessed substantial research endeavors. Suricata [2], for instance, employs a predefined set of rules to conduct traffic auditing, discerning the presence of attacks therein. With the advancement of machine learning, a plethora of intrusion detection methods empowered by artificial intelligence have emerged. PAYL [22] utilizes the payload field of network traffic as the detection target, employing natural language processing principles to extract features from the payload through 1-gram analysis. This approach achieves a high detection rate with low false positives. ATPAD [17], an end-to-end method, employs an attention mechanism to effectively defend against various payload-based attacks attempting to circumvent detection. Bao et al. [5] introduced a web attack traffic detection method based on graph networks, wherein the payload is represented using graphs, demonstrating superior representation capabilities compared to word vectors. Li et al. [11] utilized a graph matching algorithm to enhance the robustness of traffic identification. The AI-based methods encounter challenges of high computational overhead and low interpretability, while rule-based methods can overcome these drawbacks. However, the generation of rules itself is a complex task. The proposed framework in this study aims to intelligently generate rules, effectively addressing the limitations of rule-based methods.

2.2 Intelligent Intrusion Detection Rule Generation

Due to the efficiency and interpretability advantages of rule-based intrusion detection methods, as well as the inherent difficulty in their manual formulation, there have been research efforts aimed at automating the generation of intrusion detection rules.

Liu et al. [13] proposed a rule generation method based on threat intelligence text, utilizing ResLCNN to extract IOC (Indicators of Compromise) information such as malicious domains from threat intelligence text, subsequently facilitating the creation of detection rules. Basim Mahboob et al. [15] introduced an intrusion detection rule generation method based on a decision tree model, which analyzes the importance of various features and extracts interpretable detection rules from the decision tree. However, the rules generated by this method cannot be directly deployed on common rule-based intrusion detection engines. Li et al. [10] presented a rule extraction method based on an unsupervised detection model. They designed a novel decision tree called CART to approximate the decision boundaries of black-box models, but this method does not inspect malicious traffic payloads. The method proposed in this paper utilizes malicious traffic payloads to generate intrusion detection rules, providing an intuitive depiction of attack behaviors and effectively enhancing the detection accuracy.

3 Methodology

The overall architecture of the rule generation method proposed in this work is illustrated in Figure 1. Firstly, preprocessing is applied to the payload of malicious traffic. Secondly, the text features of preprocessed characters are vectorized, and a malicious payload classifier is trained based on the TextCNN model. Thirdly, in the Grad-CAM analysis module, traffic undergoes the aforementioned steps, yielding discriminative results. In the case of a positive output, the designed Grad-CAM algorithm is employed to examine the role of each word in the discriminative result, forming a heatmap to describe the contribution of each word to the outcome. Subsequently, through a sensitive words aggregation algorithm, words with lower importance scores that still play a crucial role are included in the list of key words. Finally, referencing this list, regular expressions for detection are formulated, and they are refined into intrusion detection rules, following the syntax of Suricata rules. Subsequent subsections (3.1–3.5) will provide detailed explanations of the proposed rule generation method in this paper.

3.1 Data Processing

The data preprocessing steps encompass three stages: decoding, generation, and segmentation. To address potential inaccuracies introduced by certain special characters in network traffic payload, such as "?" and "&" in URIs, we decode each payload, including URI, HTML entities, and optional Base64, Unicode, and JavaScript code. This decoding is employed to mitigate the risk of attackers encoding malicious fields. Subsequently, we generalize the decoded payload. Contents like usernames and IP addresses in the payload do not actively contribute to detection; instead, they may increase computational overhead and impact detection effectiveness. The generalization operation enhances feature

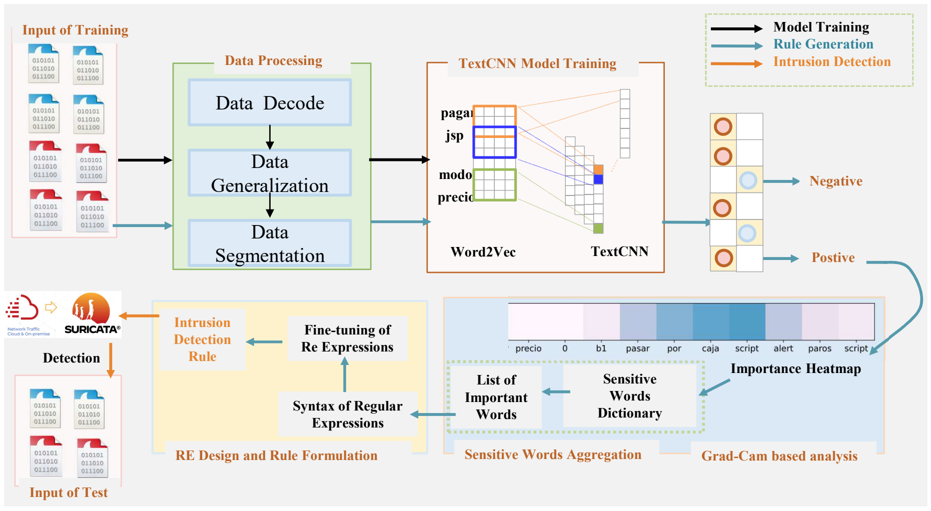


Fig. 1. The system architecture of the proposed method

learning efficiency, preventing excessive false positives in subsequent rule generation. Segmentation involves segmenting the payload into a sequence of words using symbols like "?", "&", and spaces. These three operations prepare the decoded and generalized payload for text feature vectorization.

3.2 TextCNN Model Training

The issue of malicious traffic payload classification fundamentally constitutes a textual classification task. This study employs the TextCNN model [24] for text classification, where TextCNN is composed of convolutional layers and pooling layers, culminating in a fully connected layer for output.

A malicious traffic payload serves as the input, and the text is transformed into word vectors through the application of Word2Vec [7]. Word2Vec is a word embedding technique that involves training a neural network on extensive textual data to learn vector representations for each vocabulary word, capturing the semantic relationships between words.The input text undergoes forward propagation through TextCNN to obtain the model's output:

$$Z = f(\mathrm{Conv}(X) + \mathrm{max\_pooling}(\mathrm{Conv}(X))) \tag{1}$$

where X is the word vector representation of the input text, Conv denotes the convolution operation, f represents the activation function, and Z is the output after convolution and pooling. In the context of TextCNN-based text classification, the loss computation is expressed as:

$$Loss = CrossEntropy(Z, TrueLabels) \tag{2}$$

During the subsequent backpropagation step, the loss is propagated backward through the model to calculate gradients with respect to the input.

3.3 Grad-Cam Based Analysis

In textual classification tasks, the direct application of Grad-CAM may lack the intuitive visual representation present in pixel-based image data, effective integration of Grad-CAM [18] into text data can be achieved through appropriate transformations and adaptations.

For a specific layer of a CNN, it is assumed that for a specific layer, the output of the k-th channel for the j-th token in the CNN is denoted as A^k_{ij}, and the logit score for category c before the softmax operation is y_c, so

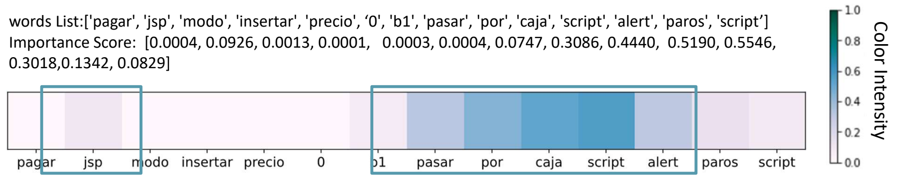
$$a^c_k = \frac{1}{T}\sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{3}$$

represents the importance weights for each channel, where $\frac{1}{T}\sum_i \sum_j$ is the average across the height and width dimensions. Subsequently, a weighted average can be applied to the output of each channel from the CNN.

$$\text{Grad-CAM Map} = \mathit{activation}(\sum_k a^c_k A^k) \tag{4}$$

Finally, based on the equation mentioned above, a heatmap is generated to represent the importance of words towards the classification results by appropriately processing and normalizing the Grad-CAM Map.

Figure 2 serves as an illustrative example of a heat map generated using the proposed methodology outlined above. It is discernible that certain words, highly improbable for usage by regular users, exhibit elevated scores. The heatmap visually represents these words with darker colors, indicative of heightened scores.



```
words List:['pagar', 'jsp', 'modo', 'insertar', 'precio', ‘0', 'b1', 'pasar', 'por', 'caja', 'script', 'alert', 'paros', 'script’]
Importance Score:  [0.0004, 0.0926, 0.0013, 0.0001,   0.0003, 0.0004, 0.0747, 0.3086, 0.4440,  0.5190, 0.5546,
0.3018,0.1342, 0.0829]
```

Fig. 2. An example of a heat map generated using the proposed method

3.4 Sensitive Words Aggregation

Observing the payload traffic, it is noticed that the occurrence of certain words, such as "cat", may not be abnormal, as regular users might use this term to refer to the animal. Consequently, its importance score in the aforementioned classification task might not be very high. However, it is also a Linux command used to view file contents. When it appears concurrently with critical system

Algorithm 1: Sensitive Words Aggregation Algorithm

```
Data: $words$, $L_{heap}$, $indexes$, $dict$
Result: $Res$
$Res \leftarrow indexes$;
for $i \leftarrow 1$ to $length(indexes)$ do
    $left = i - 1$, $right = i + 1$;
    while $left > 0$ do
        if $words[left]$ in dict and left not in indexes then
            $Res.add(left)$;
            $left = left - 1$;
        end
        if $words[left]$ in dict and left in indexes then
            $left = left - 1$;
        end
        if $words[left]$ not in dict and left not in indexes then
            break;
        end
    end
    while $right < length(words)$ do
        if $words[right]$ in dict and right not in indexes then
            $Res.add(right)$;
            $right = right + 1$;
        end
        if $words[right]$ in dict and right in indexes then
            $right = right + 1$;
        end
        if $words[right]$ not in dict and right not in indexes then
            break;
        end
    end
end
Return Res;
```

files like "/etc/passwd", it can be considered as a potentially malicious command attempting to illicitly access system password information.

In such cases, neglecting its importance may result in overlooking potential attacks. Therefore, after obtaining the heatmap representing the importance of word vectors, this paper employs an attention aggregation algorithm. This algorithm aggregates certain words with lower scores, yet possessing significance, along with closely related words that have higher scores. This aggregated information is utilized for subsequent rule generation. We propose an sensitive words aggregation mechanism. The algorithm's procedure is illustrated as Algorithm 1. The input includes the original payload sequence *words*, arrays L_{heap} containing importance scores for each term generated in the preceding subsection, and an array *indexes* containing the indices of selected crucial terms. Furthermore, a predefined sensitive word dictionary *dict* is supplied. The sensitive word dictionary encompasses various Linux system commands and other words unlikely to be used by ordinary users.

For each index already included in the *indexes* array, maintain an original sliding window of size 0, expanding towards both left and right sides with a step size of 1. If an element is encountered that appears in the sensitive word dictionary but is not present in the *indexes* array, add it to the result array, and continue expanding the sliding window. If an element is encountered that is already in the *indexes* array, increase the window size by 1 without taking any further action. If the window has reached its maximum size or encounters a condition other than the ones mentioned above, the search concludes. The final array of word indices, which is used to generate rules, is the sum of the original *indexes* array and the indices from the aforementioned *Res* array.

3.5 Regular Expression Design and Rule Formulation

In the preceding section, we obtained a sequence of key terms $w1, w2, ...wn$, representing a series of significant words that appear sequentially in a malicious network traffic. In the process of rule creation, this method utilizes the aforementioned key term sequence to describe a scenario based on regular expressions [12], wherein this sequence of key terms appears simultaneously. When all these key terms are present in a test traffic, it is considered indicative of a network attack. The specific regular expression employed is as follows: $/w1. * w2. * .... * wn/i$, utilizing a case-insensitive matching to capture the simultaneous occurrence of the specified key terms.

After crafting regular expressions for detection, it is necessary to fine-tune them. The approach involves using these regular expressions to search through negative payloads and observing whether the number of matches is acceptable. If the regex designed for malicious payloads also produces numerous matches, additional terms should be incorporated into the regular expression to ensure it doesn't match negative traffic.

The intrusion detection rules developed in this study must adhere to the syntax constraints of the Suricata engine. Suricata is an open source intrusion detection tool known for its high efficiency and ease of deployment. The intrusion detection rules in Suricata primarily include Rule ID, detection content, and alert content. The detection content is a crucial factor influencing the effectiveness of intrusion detection, as it directly describes the conditions under which an alert will be triggered. Some other crucial keywords for Suricata rules are listed in Table 1.

Table 1. Partial Catalog of Suricata Rule Keywords

Field Name	Meaning	Example
Action	Action executed when rule matching	alert/drop
Protocol	Network protocol	TCP/HTTP
Source IP	IP address of the source	192.168.1.10
Source Port	Port used by the source	54789
Dest. IP	IP address of the destination	any
Dest. Port	Port used by the destination	any
msg	Content of alert triggered	ET SCAN Suspicious Traffic
Signature ID	ID associated with the signature	2019812
Content	Content to be detected	index.php
Pcre	Regularized representation of the content	/[0-9]{6}/

4 Experiments Evaluation

4.1 Datasets

To validate the detection performance of the proposed model, experiments were conducted using the CSIC2010 [16] and FWAF [1] datasets. CSIC2010 comprises HTTP data traffic generated for e-commerce web applications, consisting of 36,000 normal requests and over 25,000 abnormal requests. The dataset includes various attacks such as SQL injection, buffer overflow, information gathering, file disclosure, CRLF injection, cross-site scripting, server-side includes, parameter tampering, etc. FWAF is a publicly available large-scale malicious request dataset released by Fsecurify, a company dedicated to developing intelligent web firewalls. They combine professional knowledge with heuristic methods to label malicious traffic.The two datasets were divided into training and testing sets in a 8:2 ratio for experimentation. The training set is utilized to train the TextCNN model and generate intrusion detection rules, while the testing set aims to evaluate the effectiveness of the generated rules. It is noteworthy that the classification target of the proposed method in this paper is binary classification.

Additionally, this study captured 100GB of benign traffic on the laboratory gateway for the purpose of validating whether the generated intrusion detection rules exhibit an acceptable false positive rate and demonstrate efficient detection performance.

4.2 Experimental Environment and Other Configurations

The experiments in this paper were conducted on a 64-bit Windows 10 operating system, with the exception of the suricata-related detection experiments, which

[1] https://github.com/faizann24/Fwaf-Machine-Learning-driven-Web-Application-Firewall

were performed on Ubuntu 20.04. All functionality modules were developed and executed in Python 3.9.1. For the vectorization of traffic payloads, a word2vec model was retrained based on a traffic payload vocabulary, and the embedding dimension of word vectors was set to 300. During the training of the TextCNN model, the Adam optimizer was employed with a batch size of 32, a learning rate of 0.001, and 50 training epochs.

4.3 Evaluation Metrics

The result of each classification may either be positive or negative in this binary decision problem. For evaluating the method we proposed, we adopt three widely used metrics in experiments, defined as follows:

$$\text{Accuracy(ACC)} = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

$$\text{False Positive Rate(FPR)} = \frac{FP}{TN + FP} \tag{6}$$

$$\text{False Negative Rate(FNR)} = \frac{FN}{TP + FN} \tag{7}$$

where TP (True Positive) refers to the number of classified as a specific class correctly; FP (False Positive) refers to the number of the misclassified as that class; FN (False Negative) refers to the number of cases that should be classified as that class but misclassified as other classes; TN (True Negative) is the number of cases that predicted as the non-corresponding classes.

4.4 Evaluation Results and Discussion

The experimental evaluation primarily involves the following aspects:

- Accuracy: We evaluate the detection accuracy of the proposed method, particularly examining whether the detection rate diminishes and whether it indeed achieves lower false positive rates.
- Ablation Study: We identify what factors influence the proposed method's effectiveness.
- Computational Consumption: We evaluate whether the rule-based detection method introduced in this paper incurs lower computational overhead compared to AI-based methods.
- Detection Efficiency: Assess whether the proposed method demonstrates higher detection speed during actual deployment testing.

Accuracy In this study, we compare our proposed method with some feature-based methods and end-to-end methods. Feature-based baselines include HMM-PAYL [3], Logistic Regression (LR) [19], Support Vector Machine (SVM) [19], and Random Forest (RF) [20]. End-to-end methods include TextCNN [23], AT-PAD [21], and GraphXSS [14]. For the baselines, we reimplement them with the

Table 2. Comparison of the method in this paper Accuracy (Acc), False Positive Rate(FPR), and False Negative Rate(FNR) with baselines.

Model	CSIC2010			FWAF		
	Acc	**FNR**	**FPR**	**Acc**	**FNR**	**FPR**
Handcrafted-features-based Methods						
HMMPAYL	92.63	8.27	7.97	91.26	7.51	8.32
LR	94.68	7.64	4.02	93.57	4.32	3.98
SVM	95.12	5.53	7.04	96.38	4.82	5.21
RF	95.06	3.63	3.27	95.72	3.03	3.56
End-to-end Methods						
GraphXSS	97.21	3.31	2.93	97.44	2.61	2.59
ATPAD	95.86	5.60	0.30	97.93	3.23	3.76
TextCNN	98.48	0.38	2.74	96.67	4.66	3.79
Ours	98.51	2.90	0.08	98.94	3.12	0.57

parameters described in the original papers on the datasets mentioned above (if no relevant experimental results are available). Table 2 demonstrates that our method exhibits relatively good detection performance on both datasets. Generally, our method demonstrates the best performance across most metrics. From the performance metric FNR on CSIC2010, it can be observed that the detection rate of our method has not declined. There is even a slight improvement compared to TextCNN. This is attributed to the precise identification of keywords in the Grad-CAM analysis stage and the synergistic effects of subsequent steps. The FNR slightly decreases compared to TextCNN on the FWAF dataset, due to the greater diversity of data in this dataset. However, this decrease is within an acceptable range. Additionally, it is worth noting that our method exhibits outstanding performance in the FPR performance metric, significantly lower than the baseline methods, showcasing the advantages of rule-based intrusion detection. This suggests that our approach integrates the powerful representational capabilities of an AI detector while retaining the advantages of rule-based detection.

Meanwhile, rules generated based on two datasets were deployed, and false positive tests were conducted using the benign traffic captured as described in Section 3.1. The experimental results indicate that, with 100GB of traffic, only 0 and 3 false positives were respectively generated for the two rule sets. This suggests that when these rules are formally deployed in the future, the likelihood of encountering numerous false positives is low, rendering them practically usable.

Ablation Experiments We conducted ablation experiments to assess the impact of the sensitive words aggregation algorithm and the fine-tuning of regular expressions on the detection results. In this work, the sensitive words aggregation algorithm aims to obtain a more comprehensive list of key terms, while the fine-tuning of regular expressions is intended to reduce potential false positives.

In the ablation experiments, we compared the original experimental results with those where the Sensitive Words Aggregation step or the Regular Expressions Fine-tuning step was omitted. The results are illustrated in Figure 3. As anticipated, the detection performance deteriorates when either step is omitted. Specifically, the fine-tuning of regular expressions has a more significant impact on FPR, while its effect on FNR is relatively minor, indicating its role in reducing false positives. On the other hand, the Sensitive Words Aggregation Algorithm has a substantial impact on both FNR and FPR, as a more comprehensive list of key terms enables more precise filtering of malicious traffic. The experimental results align with expectations, confirming the beneficial impact of both modules on the detection outcomes.

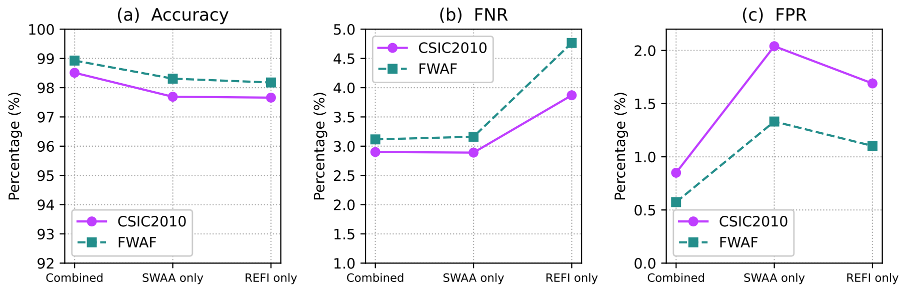


Fig. 3. The comparative results of the ablation experiments for the Sensitive Words Aggregation(SWA) Algorithm and Regular Expressions Fine-tuning(REFI). Each subplot presents the contrasts in ACC, FNR, and FPR between the proposed method and the scenarios where only SWA or only REFI is conducted.

Computational Consumption To validate whether the computational overhead required by the proposed method is acceptable in practical applications, we recorded the total training durations in comparison to end-to-end baseline methods. As shown in Table 3, the training duration of our method is slightly less than ATPAD but more than TextCNN. This discrepancy is attributed to the additional rule generation process following the training of a TextCNN model. However, the time difference indicates that the rule generation step is relatively time-efficient. Thus, the computational overhead of our method falls within reasonable limits, providing favorable conditions for its practical deployment. Additionally, we recorded the maximum memory consumption during training on the CSIC2010 and FWAf datasets, which were 1581 MB and 932 MB, respectively. Considering the powerful computational capabilities commonly available in contemporary computers, such memory consumption is deemed acceptable.

Table 3. Comparison of the method in this paper Acc training time consumption and detection efficiency with end-to-end methods. The **Train** column is the total training time and the **Test** column is the average time to classify a single instance in the test set.

Model	**CSIC2010**		**FWAF**	
	Train	**Test**	**Train**	**Test**
TextCNN	36m47s	0.61ms	38m26s	0.59ms
ATPAD	52m49s	0.81ms	46m25s	0.79ms
Ours	44m12s	0.51ms	43m58s	0.47ms

Detection Efficiency As shown in Table 3, we also recorded the time required for end-to-end baseline methods and our proposed method to process each traffic flow during the testing phase. It is observed that the rule-based method proposed in this paper has the lowest processing time on both datasets. This is expected, as intrusion detection methods based on Suricata are generally faster than other model-based methods, often implemented in Python. Additionally, it is worth noting that, compared to AI models, rule-based methods offer more straightforward deployment and practical advantages when applied to high-speed traffic systems.

5 Conclusion and Future Work

This paper introduces an intelligent generation framework for intrusion detection rules based on Grad-CAM. The core idea is to assess the importance of each word in attack payloads based on the Grad-CAM algorithm and the parameters of a pre-trained TextCNN model. Furthermore, through operations such as sensitive words aggregation, this keywords list is further refined, ultimately generating intrusion detection rules. Experimental results demonstrate that the proposed method exhibits a commendable generation rate, low computational overhead, and relatively superior detection performance.

In future work, optimization of the proposed method could be considered in the following aspects. Firstly, the method designed in this paper primarily targets traffic using the HTTP protocol. We should explore how to intelligently generate rules in scenarios involving encrypted traffic. Secondly, we plan to explore methods to enhance the generalization capabilities of intrusion detection rules, facilitating their practical application in cybersecurity.

6 Acknowledgement

This work was supported by the National Key R&D Program of China with No. 2021YFB3101402.

References

1. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F.: Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies **32**(1), e4150 (2021)
2. Albin, E., Rowe, N.C.: A realistic experimental comparison of the suricata and snort intrusion-detection systems. In: 2012 26th International Conference on Advanced Information Networking and Applications Workshops. pp. 122–127. IEEE (2012)
3. Ariu, D., Tronci, R., Giacinto, G.: Hmmpayl: An intrusion detection system based on hidden markov models. computers & security **30**(4), 221–241 (2011)
4. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 3971–3988 (2022)
5. Bao, H., Li, W., Wang, X., Tang, Z., Wang, Q., Wang, W., Liu, F.: Payload level graph attention network for web attack traffic detection. In: International Conference on Computational Science. pp. 394–407. Springer (2023)
6. Caswell, B., Beale, J., Baker, A.: Snort intrusion detection and prevention toolkit. Syngress (2007)
7. Di Gennaro, G., Buonanno, A., Palmieri, F.A.: Considerations about learning word2vec. The Journal of Supercomputing pp. 1–16 (2021)
8. Jacobs, A.S., Beltiukov, R., Willinger, W., Ferreira, R.A., Gupta, A., Granville, L.Z.: Ai/ml for network security: The emperor has no clothes. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 1537–1551 (2022)
9. Li, J., Pan, Z.: Network traffic classification based on deep learning. KSII Transactions on Internet & Information Systems **14**(11) (2020)
10. Li, R., Li, Q., Zhang, Y., Zhao, D., Jiang, Y., Yang, Y.: Interpreting unsupervised anomaly detection in security via rule extraction. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)
11. Li, W., Zhang, X.Y., Bao, H., Wang, Q., Li, Z.: Robust network traffic identification with graph matching. Computer Networks **218**, 109368 (2022)
12. Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Jagadish, H.: Regular expression learning for information extraction. In: Proceedings of the 2008 conference on empirical methods in natural language processing. pp. 21–30 (2008)
13. Liu, L., Zhao, Q., Zheng, R., Tian, Z., Sun, S.: An automatically generated intrusion detection rule method based on threat intelligence. Computer Engineering and Design **43**(1), 1–8 (2022)
14. Liu, Z., Fang, Y., Huang, C., Han, J.: Graphxss: an efficient xss payload detection approach based on graph convolutional network. Computers & Security **114**, 102597 (2022)
15. Mahbooba, B., Timilsina, M., Sahal, R., Serrano, M.: Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. Complexity **2021**, 1–11 (2021)
16. Nguyen, H.T., Torrano-Gimenez, C., Alvarez, G., Petrović, S., Franke, K.: Application of the generic feature selection measure in detection of web attacks. In: Computational Intelligence in Security for Information Systems: 4th International Conference, CISIS 2011, Held at IWANN 2011, Torremolinos-Málaga, Spain, June 8-10, 2011. Proceedings. pp. 25–32. Springer (2011)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63783-4_12

17. Qin, Z.Q., Ma, X.K., Wang, Y.J.: Attentional payload anomaly detector for web applications. In: Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part IV 25. pp. 588–599. Springer (2018)
18. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
19. Smitha, R., Hareesha, K., Kundapur, P.P.: A machine learning approach for web intrusion detection: Mamls perspective. In: Soft Computing and Signal Processing: Proceedings of ICSCSP 2018, Volume 1. pp. 119–133. Springer (2019)
20. Tama, B.A., Nkenyereye, L., Islam, S.R., Kwak, K.S.: An enhanced anomaly detection in web traffic using a stack of classifier ensemble. IEEE Access **8**, 24120–24134 (2020)
21. Wang, J., Zhou, Z., Chen, J.: Evaluating cnn and lstm for web attack detection. In: Proceedings of the 2018 10th International Conference on Machine Learning and Computing. pp. 283–287 (2018)
22. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: International workshop on recent advances in intrusion detection. pp. 203–222. Springer (2004)
23. Yu, L., Chen, L., Dong, J., Li, M., Liu, L., Zhao, B., Zhang, C.: Detecting malicious web requests using an enhanced textcnn. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). pp. 768–777. IEEE (2020)
24. Zhang, T., You, F.: Research on short text classification based on textcnn. In: Journal of Physics: Conference Series. vol. 1757, p. 012092. IOP Publishing (2021)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63783-4_12