Payload Level Graph Attention Network for Web Attack Traffic Detection

Huaifeng Bao^{1,2}, Wenhao Li^{1,2}, Xingyu Wang^{1,2}, Zixian Tang^{1,2}, Qiang Wang^{1,2} Wen Wang^{1,2}, and Feng Liu^{1,2}

^{1} State Key Laboratory of Information Security
Institute of Information Engineering, CAS, Beijing, China
^{2} School of Cyber Security, University of CAS, Beijing, China
{baohuaifeng, liwenhao, wangxingyu, tangzixian, wangqiang3113, wangwen, liufeng}@iie.ac.cn

Abstract. With the popularity of web applications, web attacks have become one of the major threats to cyberspace security. Many studies have focused on applying machine learning techniques for web attack traffic detection. However, past approaches suffer from two shortcomings: firstly, handcrafted feature extraction-based approaches are prone to introduce biases in existing perceptions, and secondly, current end-to-end deep learning approaches treat traffic payloads as non-structured string sequences ignoring their inherent structural characteristics. Therefore, we propose a graph-based web attack traffic detection model to identify the payloads in the traffic requests. Each pre-processed payload is transformed as an independent graph in which the node representations are shared through a global feature matrix. Finally, graph-level classification models are trained with graph attention networks combining global information. Experimental results on four publicly available datasets show that our approach successfully exploits local structural characteristics and global information to achieve state-of-the-art performance.

Keywords: web attack, traffic classification, graph representation learning, graph attention networks

1 Introduction

With the rapid iteration of Internet technology and the continuous improvement of computing power, the more efficient B/S architecture is becoming increasingly popular on the Internet. More and more content providers deploy their services on web pages to replace their original applications. In addition to traditional websites, various application programming interfaces (APIs) and applets have become new sources of traffic. More traffic entries and invocation methods have increased the efficiency of web application development while also bringing more complex security issues. Some malicious attackers have proposed various attack methods by studying Web technologies, such as cross-site scripting (XSS) attacks, Structured Query Language (SQL) injection, command execution, directory traversal, etc. With these attack methods and the vulnerabilities in web services, web attackers can maliciously attack and infiltrate web applications to steal user data or disrupt the normal operation of the system. According to the [8], only for financial institutions, more than 736 million web attacks were recorded in 2020. It can be seen that web attacks have become one of the most mainstream cyber attacks, with a wide range of attacks and most of them posing serious privacy risks or financial losses.

In order to circumvent the above malicious attacks, researchers in related fields have conducted a lot of research. Many scholars have proposed techniques from the perspective of strengthening the security of web application , such as reverse proxy and SSL verification. However, such defenses rely heavily on web security team building, which is costly and not flexible enough to cope with changing attack methods. Therefore, more researchers focus on detecting web attacks from the application layer traffic payload. Traditional web application firewalls (WAFs) apply regular expressions to match sensitive fields from traffic requests for misuse detection. Considering that the expressive power of a single regular expression is insufficient to match the increasingly complex forms of web attack payloads, researchers have started to detect known attacks by building databases of attack signatures [3]. Such approaches require constant updating of the attack signature databases to adapt to new attack techniques and vulnerability types. The lag caused by such periodic updates poses a potential problem for system security.

With the popularity of artificial intelligence techniques, applying related techniques in web attack traffic detection has become a hot research topic. These methods can be broadly classified into two categories: handcrafted-features-based and end-to-end methods. Handcrafted-features-based methods extract features from suspicious traffic requests guided by expert experience and feed them numerically into traditional machine learning models to obtain determination results. Typical methods include extracting statistical features from the request header fields, extracting n-gram statistical features from the request payload, etc. These methods transform traffic into numerical features based on a priori knowledge of web security. So they face problems such as the inability to obtain comprehensive and in-depth information in the traffic, poor robustness against traffic pattern variations, and time-consuming pre-processing stages. In addition, the detection performance of such methods is limited by the classification capability of traditional machine learning algorithms. Recently, as deep learning has been widely used in web attack detection research, end-to-end methods have shown more robust detection capability and generalization performance with no need to design new features for novel attack methods. Researchers have started treating key fields such as uniform resource identifiers (URIs) in requests as string text and introducing natural language processing (NLP) techniques to mine the semantic information within them. Models such as TextCNN [19], BiLSTM [16], and Attention [9] have been used successively for such problems. However, such approaches treat the URI or other fields in the request as unstructured text for feature extraction and ignore the non-sequential key-value pairs dependencies inherent in web attack traffic requests.

Graph neural networks are designed to work with data embedding in non-Euclidean spaces, aggregating relevant node content information through the topology of the graph, which can achieve effective feature updates and avoid the interference of disjoint fields. In this paper, we propose Payload-Level-GAT to transform structured traffic payloads into heterogeneous graphs and update node semantic features using graph attention network (GAT) [14]. And then, node representations are aggregated by introducing global information to compute graph-level embedding and implement classification tasks.

The main contributions of this paper are as follows:

– We propose a payload-level graph representation for feature learning that abstracts traffic requests into graph structures carrying structural relationships.
– To achieve the graph-level classification task, we designed Payload-Level-GAT, combining global information and local structured contextual information to enhance the classification effect.
– We have conducted sufficient experiments on four publicly available datasets. The experimental results show that our proposed method is more efficient and generalized than baselines.

2 Related Work

2.1 Malicious Traffic Analysis based on Machine Learning

Application layer protocol payloads play a crucial role in web attacks. Hence numerous works have focused on them. As one of the earliest studies in payload analysis, wang et al. [17] proposed PAYL to extract 1-gram frequency distribution from all bytes of the payload as features. This work achieves high detection rates and low false alarm rates but is vulnerable to mimicry attacks. After that, there are many works focused on payload analysis. Ariu et al. [1] proposed HMMPAYL, which was developed based on PAYL and used Hidden Markov Models to detect attack traffic. Oza et al. [6] extracted n-grams of HTTP traffic with various n-values. Bortolameotti et al. [2] generated features from HTTP request URIs and HTTP headers. Then, the attack traffic is detected based on the fingerprints generated by these features. In addition, end-to-end deep learning paradigms are becoming increasingly popular. Qin et al. [9] proposed an attention-based deep learning model, ATPAD, which was the first work to apply attention mechanisms in payload anomaly detection. Wang et al. [16] proposed a model to detect malicious traffic by combining CNN and LSTM.

2.2 Graph Neural Network

In recent years, there has been a growing interest in extending deep learning methods to structured data such as a graph. Researchers have proposed graph neural networks (GNN) to process graph data with neural networks.

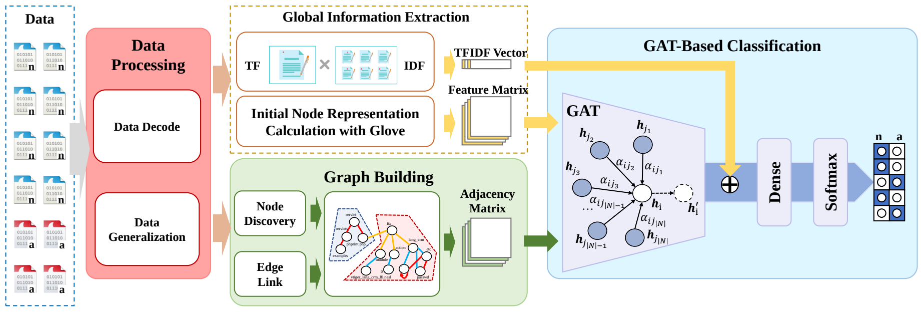


Fig. 1: Architectural view of Payload-Level-GAT.

GNN has been used in cyber security for the past few years. In [20], GNN is described to disassemble x86 instructions quickly and accurately. The key idea is to capture and propagate instruction relationships with GNN models. Another application of GNN is processing control flow graphs (CFGs) for binary code similarity detection [15]. Zhuang et al. [21] extracted control flow and data flow semantics from source code and detected vulnerabilities in smart contracts based on GNN. Recently, GNNs have also been applied to web attack detection. Liu et al. [4] proposed GraphXSS to construct graphs for the global corpus with payloads and the preprocessed items as nodes. TextGCN [18] is used to classify the payload nodes. However, this approach has two major problems: first, it uses co-occurrence relations of items in the global corpus for graph construction, ignoring local structural relations, and second, it has a large memory footprint and cannot be applied to inductive learning to determine the maliciousness of emerging traffic payloads. In contrast, our approach focuses on local structural relationships of the payloads and uses global information for their graph-level independent representation. So the method could be used for inductive learning. In addition, the memory footprint can be significantly reduced by combining mini-batch training.

3 Approach

3.1 Overall Architecture

The general architecture of our proposed method is shown in Fig. 1. First, all the data are decoded and generalized in the data preprocessing module. This is to avoid excessive dimensionality of the embedding space caused by encoding or random nonsense values and to improve the classification efficiency of the graph neural network. In the graph construction module, regular expressions are constructed to segment the payloads and discover the graph nodes from the segmented subsegments. Next, heterogeneous edges are linked, relying on contextual and structural relationships of graph nodes, forming a graph for each request. The parameters of the graph structure are learned and optimized using the GAT algorithm in the GAT-Based Classification module. The node embedding is updated starting from the initial Global Vectors for Word Representation (GloVe) [7] vector. The Tf-Idf [10] algorithm is applied to calculate the node weights with which the node embedding is aggregated to obtain the embedding of the whole graph. Finally, the classification probability distribution of the graph is calculated by SoftMax.

3.2 Data processing

To construct a graph representation of the traffic payload, it is first necessary to extract the payload portion of the application layer traffic and perform decoding and generalization operations. Some special characters that may appear in the traffic payload could have an impact on the normal parsing process of the server, such as '?', '&' in the URI, or '⟨' in the request body of HTML format. Therefore, the server will first decode the request message after receiving it. This allows attackers to bypass defenses such as firewalls by encoding malicious fields. We attempt to decode each traffic payload to avoid the semantics of the maliciously encoded fields being segmented during subsequent graph construction. This consists of constant URI decoding, HTML entity decoding and optionally Base64 decoding, Unicode decoding, and decoding with the String.fromCharCode() method for variable or function statements in JavaScript (JS) code.

Next, we further generalize the decoded payload to prevent the graph neural network from learning particular fields and even overfitting and improve the feature learning efficiency. The above decoding step substantially reduces the vocabulary size in the whole corpus of data by restoring potentially readable fields. However, there are still some fields whose values are meaningless for determining maliciousness, although they are not encoded. We use regular expressions to identify the IP, domain, and port in the payload and generalize the fields with abstract patterns. After replacing these values, we also match the remaining numeric values in the payload and replace them with 0.

3.3 Graph Building

After data preprocessing, we propose a framework to construct a graph for each traffic payload as the input to the following graph neural network. In this subsection, we first segment the payloads into independent tokens as nodes of the graph and then link the edges of the graph from the contextual and structurally logical relationships of the tokens.

The abnormal areas of web attacks are usually found in the request path and request body of the payload (the request parameters of GET requests are treated as request bodies in this paper). For example, directory traversal attacks usually contain excessive directory hierarchies in the URI, cross-site scripting attacks, and SQL injection usually contain suspicious content in the request body, such as incomplete HTML entities and JS-sensitive functions and methods. Since these suspicious contents are mainly shown as continuous readable strings separated by some special characters after decoding, we construct a set of regular expressions to split the request URI and the request bodies into several tokens, respectively. We denote payload data containing $m + n + l$ tokens as $D = \{U_1, ..., U_i, ..., U_m, K_1, ..., K_j, ..., K_n, V_1, ..., V_k, ..., V_l\}$, U_i denotes a token of one of the URIs, and K_j and V_k denote the token separated from the keys and values in the request body, respectively. To construct a graph of the traffic payload, we treat each token appearing in it as a node of the graph.

Three classes of heterogeneous edges form the edge set of the graph. The first type of edge takes its nodes from the key-value pairs in the request body and is constructed from the logical structural relationships in the request body, i.e., such an edge starts at a key in the request body and ends at a token corresponding to the value for that key. The second type of edge is constructed with contextual adjacency relations. Each edge of this type starts at a token of the URI or request value and ends at its neighboring token. In addition, to guarantee graph connectivity, we set a 'public' node B_0 to abstractly represent the request body content. The last type of edge connects this node to the last token of the URI and to each key of the request body.

Formally, a graph of a load is defined as $G = (N, E)$, where N denotes its node set and E denotes its edge set. Concretely, the node set and the edge set can be represented as

$$\begin{aligned} N = &\{U_i, i \in [1, m]\} + \{K_j, j \in [1, n]\} \\ &+ \{V_k, k \in [1, l]\} + B_0, \end{aligned} \tag{1}$$

$$\begin{aligned} E = &\{e_{jk}, j \in [1, n], k \in [1, l]\} \\ &+ \{e_{ii'}, i \in [1, m], i^{'} \in [i - p, i + p]\} \\ &+ \{e_{kk'}, k \in [1, l], k^{'} \in [k - p, k + p]\} \\ &+ \{e_{m0}\} + \{e_{0j}, j \in [1, n]\}, \end{aligned} \tag{2}$$

where p denotes the number of neighboring tokens considered in the second class of the edge construction process. During graph building, not only the proximity dependencies of contextual relations are considered, but also the potential long-distance dependencies between keys could be iterated in short steps through the abstract node B_0.

3.4 GAT-Based Classification

In this subsection, we describe how to initialize the node representations and update them with graph attention networks to obtain a representation of the graph for classification.

The GloVe model is used to construct a globally shared initialized node representation. GloVe is an unsupervised word representation tool based on global term frequency statistics. It combines term co-occurrence information on the global corpus while capturing contextual semantic links. We pre-train the GloVe model on the global traffic payload database and obtain the initialized global shared feature matrix of graph nodes. The initial set of node embedding for graph G is defined as $N_{Emd} = \{\boldsymbol{h}_n, n \in N\}$.

There are two mainstream approaches to updating node feature vectors in graph neural networks: the spectral method and the spatial method. The spectral method maps the graph onto the spectral domain, such as the space obtained from the Laplacian matrix after feature decomposition, one of the representative methods is GCN. In this work, we use GAT [14] to update node features, a spatial method that operates directly on the graph. The model extends the information of the first-order neighbor nodes into the feature representation of the current node by masked attention, which is defined as

$$a_{ij} = a([\boldsymbol{W}\boldsymbol{h}_i || \boldsymbol{W}\boldsymbol{h}_j]), \tag{3}$$

$$\alpha_{ij} = \frac{exp(LeakyReLU(a_{ij}))}{\sum_{k \in \mathcal{N}_i} exp(LeakyReLU(a_{ik}))}, \tag{4}$$

$$\boldsymbol{h}_i^{'} = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{W}\boldsymbol{h}_j). \tag{5}$$

In (3), a_{ij} denotes the importance of node j to node i, which is computed by first a linear mapping with shared parameters W to dimensionalize the features of nodes. The dimensionalized features of nodes i,j are concatenated and mapped to the real number space by $a(\cdot)$. The softmax is used to normalize the correlation coefficients to get the attention coefficients in (4). Finally, based on the computed attention coefficients, the weighted sum of the features of neighboring nodes is used as the output features of the nodes. In addition, to stabilize the learning process of self-attention, Veličković et al. found that the extension of attention with Multi-head Attention is an enhancement to the model. The computational formula of the K-head attention mechanism is defined as follows

$$\boldsymbol{h}_i^{'}(K) = \|_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{k} \boldsymbol{W}^k \boldsymbol{h}_j). \tag{6}$$

GAT aggregates the features of neighboring nodes to the central node with attention coefficients, which means that the updated node representation contains semantic correlations between neighboring nodes. Thus, even though tokens with large semantic gaps may appear in different payloads, their exact semantics in a particular traffic payload can be obtained by weighting the information of their neighboring nodes. Moreover, since the attention parameters a and W are shared globally during the update process, it implies that the final representation of the nodes also contains global information similar to that of other graph neural network models.

Finally, the representations of all nodes in the graph are used to predict the labels of the payloads,

$$y = softmax(ReLU(W_2 \sum_{n \in N} \beta_n \boldsymbol{h}_n^{'}(K) + b)), \tag{7}$$

where β_n serves as the contribution weights of the nodes determined by the TF-IDF algorithm, $W_2 \in \mathbb{R}^{d \times c}$ is a matrix that maps the node features to the graph representation space, $b \in \mathbb{R}^c$ is the bias vector. The training objective is to minimize the cross-entropy of the ground truth label g_i and the predicted label y_i

$$l = -g_i log y_i. \tag{8}$$

4 Experimental Evaluation

4.1 Dataset and Setup Description

To verify the detection effect of the proposed model, experiments were conducted in CSIC2010 [5], FWAF [3], TBWIDD [12], and BDCI2022 [4]. CSIC2010 contains HTTP traffic data generated for e-commerce web applications, including 36000 normal requests and 25065 abnormal requests. The anomalous requests include popular attacks such as sql injection, buffer overflow, cross-site scripting, etc. FWAF is a publicly available large-scale malicious request dataset published by Fsecurify, a company aiming to develop intelligent web firewalls. They combine expert knowledge with heuristics for labeling malicious traffic. TBWIDD is published by Stevanović et al. They develop and deploy web honeypots to capture in-the-wild web attack traffic by filtering normal behavior through predefined whitelists. It contains 13048 normal requests and 9249 abnormal requests. The BDCI2022 dataset comes from the public data of the Web Attack Detection and Classification Identification track of the CCF Big Data & Computing Intelligence Contest 2022. The traffic data is divided into six labels, including normal requests, SQL injection, XSS, directory traversal, command execution, and remote code execution, totaling about 35,000 pieces. We divide the above four datasets into training, validation, and test sets according to the 6:2:2 ratio to conduct experiments, respectively. It is worth noting that except for the setting of multiclassification on the BDCI2022 dataset, the objectives of the remaining three datasets are binary classification.

We set the dimensionality of the node representation to 300 and initialize the vector with GloVe as described in Section 3.4. Adam is used as the optimizer. The initial learning rate is set to 0.001, the batch size to 64, the number of training epochs to 100, and the number of early stop epochs to 10. We compare the performance of our method by three widely used metrics, including accuracy (Acc), missed alarm rate (MA), and false alarm rate (FA). These metrics can be calculated as follows.

$$Acc = \frac{TP + TN}{TP + FN + FP + TN} \tag{9}$$

$$MA = \frac{FN}{TP + FN} \tag{10}$$

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

[4] https://www.datafountain.cn/competitions/596

$$FA = \frac{FP}{FP + TN} \tag{11}$$

where TP represents the correctly classified positive samples, FN represents the misclassified negative samples, and FP represents the misclassified negative samples.

4.2 Classification Results and Discussion

RQ1. How is the detection performance of Payload-Level-GAT?

We compare Payload-Level-GAT with some handcrafted feature-based methods and end-to-end methods. Handcrafted feature-based baselines include HMM-PAYL [1], logistic regression (LR) [11], support vector machine (SVM) [11], RandomForest (RF) [13]. The end-to-end methods include TextCNN [19], LSTM [16], ATPAD [9], and GraphXSS [4] based on TextGCN. For baselines, we use the parameters described in their original paper to reimplement on the above datasets (if no relevant experimental results are available). We also take the 300-dim GloVe model for the models requiring initial embedding. Table 1 shows that our method performs competitively in detection performance on all four datasets. Our method shows the optimal performance on most metrics, with only the Fa on CSIC2010 and the Ma on BDCI2022 being suboptimal and marginally less than the optimal values. It can be seen that the end-to-end methods generally outperform the handcrafted feature-based methods. And the gap is more noticeable in challenging classification scenarios, such as multi-classification on BDCI2022, and less in more straightforward scenarios, such as TBWIDD dataset with similar detection performance across models. Remarkably, our method demonstrates perfect detection capability on the TBWIDD dataset, the potential reason for which may be that the structural information in the payload ignored by the baseline is exploited by Payload-Level-GAT. In addition to our method, another graph-based method, GraphXSS, shows superior performance in baselines, with nearly half of the metrics being suboptimal. This indicates that graph neural network-based methods hold advantages over traditional deep learning-based approaches.

To verify the detection efficiency of Payload-Level-GAT, we record the total training time and the detection time for a single test instance compared to other end-to-end baseline methods. As shown in Table 2, for the total training time and detection efficiency, the difference between Payload-Level-GAT and the baseline methods is not significant. The training time on the FWAF dataset is significantly lower than the other baselines, which can be explained by the simpler payload structure of the instances in this dataset that makes graph node updates faster, which coincides with the smaller number of edge nodes of the graphs in this dataset in Table 4. Overall, Payload-Level-GAT pervasively shows stronger detection performance on four different datasets while being comparable in efficiency to the end-to-end baseline approach.

RQ2. What factors affect the detection performance of Payload-Level-GAT?

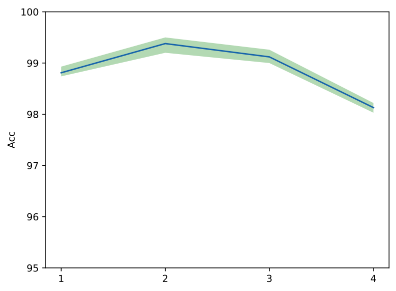
Table 1: Comparison of Payload-Level-GAT Accuracy (Acc), Missed Alarm rate (Ma), and False Alarm rate (Fa) with baselines. Highlighted values are: (**Bold Font**) overall best classifier and (†) second best, for each metric.

Model	**CSIC2010**			**FWAF**			**TBWIDD**			**BDCI2022**		
	Acc	**MA**	**FA**	**Acc**	**MA**	**FA**	**Acc**	**MA**	**FA**	**Acc**	**MA**	**FA**
Handcrafted-features-based Methods												
HMMPAYL	92.63	8.27	7.97	91.26	7.51	8.32	96.84	3.48	3.76	85.32	23.81	28.93
LR	94.68	7.64	4.02	93.57	4.32	3.98	97.23	2.54	1.83	79.85	31.28	35.24
SVM	95.12	5.53	7.04	96.38	4.82	5.21	98.30	1.43	1.02	93.26	17.58	23.49
RF	95.06	3.63	3.27	95.72	3.03	3.56	98.68	1.62	0.55†	94.76	28.30	32.14
End-to-end Methods												
TextCNN	98.48†	0.38†	2.74	96.67	4.66	3.79	97.79	2.05	2.13	94.57	6.80	7.16
LSTM	98.90	1.20	1.10	94.28	2.94	3.57	98.22	1.27	1.38	94.55	6.57	9.35
ATPAD	95.86	5.60	**0.30**	97.93†	3.23	3.76	97.53	2.51	3.07	93.90	8.37	7.08†
GraphXSS	97.21	3.31	2.93	97.44	2.61†	2.59†	99.01†	1.06†	1.06	97.26†	**5.56**	9.52
Ours	**99.38**	**0.32**	0.61†	**98.69**	**1.17**	**1.31**	**100.00**	**0.00**	**0.00**	**97.47**	6.01†	**2.53**

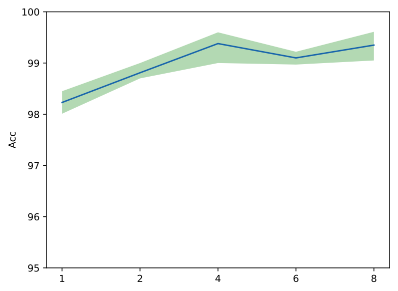
Table 2: Comparison of Payload-Level-GAT training time consumption and detection efficiency with end-to-end methods. The **Train** column is the total training time and the **Test** column is the average time to classify a single instance in the test set.

Model	**CSIC2010**		**FWAF**		**TBWIDD**		**BDCI2022**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
TextCNN	36m47s	0.61ms	38m26s	0.59ms	3s	0.39ms	21m46s	0.54ms
LSTM	1h7m55s	0.78ms	58m19s	0.70ms	4s	0.52ms	31m30s	0.79ms
ATPAD	52m49s	0.81ms	46m25s	0.79ms	4s	0.75ms	27m35s	0.83ms
GraphXSS	38m57s	0.74ms	41m18s	0.73ms	3s	0.62ms	15m26s	0.56ms
Ours	49m5s	0.76ms	15m12s	0.63ms	5s	0.58ms	21m42s	0.45ms

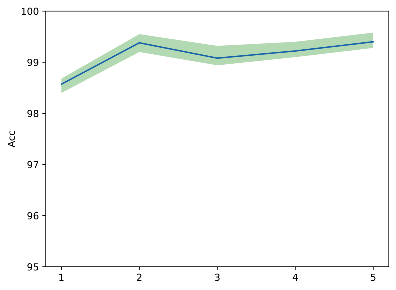
We performed ablation experiments to verify the effect of local structure information as well as global information on detection performance. In this work, local structure information is captured in the construction of the graph for the second and third types of heterogeneous edges. We set the ablation experiments to eliminate such edges and construct the graph from token contextual relations exclusively. In addition to the shared parameters of the graph neural network, global information is introduced in the initial GloVe embedding for graph nodes and the computation of the graph embedding weights by the Tf-Idf algorithm. In the ablation experiment, the node representations are initialized in random embedding and the graph embedding is calculated by isometric summation. Table 3 shows the detection performance for the model without global information (GI) and without local structure information (LSI) on the four datasets and the per-



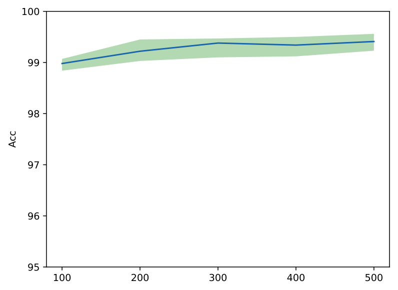
(a) GAT Layers



(b) Attention Heads



(c) p Value



(d) GloVe Embedding Dim

Fig. 2: Comparison results of different hyperparameters on CSIC2010.

formance loss compared to the standard method. From the data in the table, it can be concluded that both local structural information and global information bring about an improvement in detection performance.

In addition, we conducted comparative experiments on the CSIC2010 dataset for the key hyperparameters of Payload-Level-GAT to investigate the impact of hyperparameters on model detection performance. These hyperparameters include the number of GAT layers, the number of Attention headers in a single-layer GAT, the context range p considered in the construction of the first type of heterogeneous edges, and the dimensionality of GloVe encoding. Fig. 2 shows the effect of different hyperparameters on the detection performance. Overall, the accuracy of the model performs stably with different parameters taken into account.

RQ3. How is the memory consumption of Payload-Level-GAT? GraphXSS, which also uses graph neural networks for payload classification, builds a graph based on the global corpus with two types of heterogeneous nodes, payload, and token in payload, achieving payload detection through node classification. As the corpus grows, the size of the global graph becomes larger, and

Table 3: Comparison Results of Ablation Experiments for global information (GI) and local structural information (LSI).

Datasets		w/o GI	w/o LSI
CSIC2010	Acc %	98.96 (0.42 ↓)	98.13 (1.25 ↓)
	MA %	0.78 (0.16 ↑)	0.93 (0.31 ↑)
	FA %	0.89 (0.28 ↑)	1.36 (0.72 ↑)
FWAF	Acc %	97.82 (0.87 ↓)	97.59 (1.10 ↓)
	MA %	1.78 (0.61 ↑)	2.10 (0.93 ↑)
	FA %	1.99 (0.68 ↑)	2.03 (0.72 ↑)
TBWIDD	Acc %	99.51 (0.49 ↓)	99.22 (0.78 ↓)
	MA %	0.92 (0.92 ↑)	0.84 (0.84 ↑)
	FA %	0.77 (1.22 ↑)	1.05 (1.05 ↑)
BDCI2022	Acc %	95.37 (2.10 ↓)	96.82 (0.65 ↓)
	MA %	8.35 (2.34 ↑)	8.41 (8.40 ↑)
	FA %	6.25 (3.72 ↑)	5.42 (2.89 ↑)

the dependencies between nodes become more complex. The computational cost and memory requirements will become unbearable. The memory footprint and the number of edges in the graph for GraphXSS and Payload-level-GAT (mean value in our method) are recorded in Table 4. It can be seen that our proposed method has a low memory footprint and is easier to deploy in practice combining with batch training strategies.

Table 4: Comparison of memory consumption and number of edge set for GraphXSS and Payload-Level-GAT.

Datasets	**GraphXSS**	**Payload-Level-GAT**
CSIC2010	84370MB(2305449)	285MB(92)
FWAF	76905MB(1619610)	119MB(68)
TBWIDD	4660MB(69551)	28MB(55)
BDCI2022	23964MB(2140072)	291MB(213)

5 Conclusion and Feature Work

In this paper, we propose Payload-Level-GAT, a graph attention network-based approach to detect web attack traffic. The core idea is to regard the payload of web attack traffic as semi-structured text and construct payload-level graphs through local contextual and structured relationships. Furthermore, a GAT-based detection framework incorporating global information to classify the payload-level graphs is proposed. The results show that Payload-Level-GAT has good detection performance and generalization ability.

In future work, there are two directions to improve Payload-Level-GAT. First, although our scheme has the potential to be deployed in decrypted traffic environments at enterprise gateways with the widespread adoption of encrypted traffic orchestration, we plan to conduct graph structure modeling of encrypted attack traffic with side-channel information to more efficiently address the trend of traffic encryption. Secondly, we plan to explore the interpretability of graph neural networks from the perspective of result analysis to support its on-the-ground application in cybersecurity.

References

1. Ariu, D., Tronci, R., Giacinto, G.: Hmmpayl: An intrusion detection system based on hidden markov models. computers & security **30**(4), 221–241 (2011)
2. Bortolameotti, R., van Ede, T., Caselli, M., Everts, M.H., Hartel, P., Hofstede, R., Jonker, W., Peter, A.: Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. In: Proceedings of the 33rd Annual computer security applications Conference. pp. 373–386 (2017)
3. Clincy, V., Shahriar, H.: Web application firewall: Network security models and configuration. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). vol. 1, pp. 835–836. IEEE (2018)
4. 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)
5. 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, pp. 25–32. Springer (2011)
6. Oza, A., Ross, K., Low, R.M., Stamp, M.: Http attack detection using n-gram analysis. Computers & Security **45**, 242–254 (2014)
7. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
8. Pitchkites, M.: Top cyber security statistics, facts & trends in 2022 (2022), https://www.cloudwards.net/cyber-security-statistics/
9. Qin, Z.Q., Ma, X.K., Wang, Y.J.: Attentional payload anomaly detector for web applications. In: International Conference on Neural Information Processing. pp. 588–599. Springer (2018)
10. Ramos, J., et al.: Using tf-idf to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning. vol. 242, pp. 29–48. New Jersey, USA (2003)
11. Smitha, R., Hareesha, K., Kundapur, P.P.: A machine learning approach for web intrusion detection: Mamls perspective. In: Soft Computing and Signal Processing, pp. 119–133. Springer (2019)
12. Stevanović, N., Todorović, B., Todorović, V.: Web attack detection based on traps. Applied Intelligence pp. 1–25 (2022)
13. 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)

14. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations
15. Wang, H., Qu, W., Katz, G., Zhu, W., Gao, Z., Qiu, H., Zhuge, J., Zhang, C.: jtrans: jump-aware transformer for binary code similarity detection. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 1–13 (2022)
16. 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)
17. 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)
18. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 7370–7377 (2019)
19. 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)
20. Yu, S., Qu, Y., Hu, X., Yin, H.: Deepdi: Learning a relational graph convolutional network model on instructions for fast and accurate disassembly. In: Proc. of the USENIX Security Symposium (2022)
21. Zhuang, Y., Liu, Z., Qian, P., Liu, Q., Wang, X., He, Q.: Smart contract vulnerability detection using graph neural network. In: IJCAI. pp. 3283–3290 (2020)