Rule Generation for Network Intrusion Detection Systems Based on Packets-To-Video Transformation

Sheng-Tzong Cheng(✉), Yu-Ling Cheng, and Ka-Chun Cheung

National Cheng Kung University, Tainan, Taiwan
stevecheng1688@gmail.com

Abstract. Modern intrusion detection systems utilize machine learning to identify network anomalies. Traditional static rules may not be sufficient to combat emerging attacks, making it critical to adopt a dynamic approach for keeping intrusion detection rules up-to-date. This study introduces an intelligent rule generator with a packet encoding method to represent packets into images, a vision model to encode the images, and a video captioning model, mapping image features to textual descriptions, thereby generating rules suitable for network intrusion detection systems. The results of our simulated data experiments show that our classification model has a higher accuracy than others and is capable of generating rules.

Keywords: Rule-based Network Intrusion Detection System · Rule Generator · Video Captioning · Network Security

1 Introduction

With the emergence of generative artificial intelligence, there has been a shift from traditional classification problems to generating new content, such as articles and images. This trend has also extended to network security, where past studies focused primarily on classifying packet flow, identifying whether it is an attack or what kind of attack it is. However, recent studies have incorporated generative artificial intelligence into this domain.

Intrusion Detection Systems (IDS) have traditionally relied on pre-defined rules. However, with the rapid changes in network environments and evolving intrusion techniques, pre-defined rules often need to respond more quickly to novel intrusion behaviors. Furthermore, updating the rule set of IDS is not only time-consuming but also labor-intensive. Therefore, integrating generative artificial intelligence into IDS can offer a more efficient solution to adapt to the ever-changing network traffic.

By leveraging generative artificial intelligence, IDS can evolve from reactive to proactive systems. IDS can learn from its environment, adapt to new threats, and generate

This research is supported by the project of National Science and Technology Council, Taiwan, R.O.C under project no. NSTC 112-2218-E-006-012.

D.-J. Deng and J.-C. Chen (Eds.): SGIoT 2023, LNICST 557, pp. 51–65, 2024.
https://doi.org/10.1007/978-3-031-55976-1_6

new rules autonomously. This approach can significantly reduce the time and resources required to update the rule set of IDS.

This paper proposes an alternative approach to traditional generations. Firstly, we collect network traffic datasets containing common attacks. We convert packets into images to further process this data and build our architecture for attack classification and feature extraction from traffic data. Subsequently, we employ these features to train the captioning model, LSTM, for generating new rules for IDS.

The remaining sections of this paper are structured as follows: Sect. 2 discusses related work, including prior research and methods relevant to our study. Section 3 describes our approach regarding traffic data preprocessing, feature extractors, and captioning models. Section 4 elaborates on the specific implementation details of our experiments, including dataset collection, training processes, and evaluation metrics. Finally, we conclude our research findings and discuss future work and potential areas for improvement in Sect. 5.

2 Related Work

2.1 Problem for Problem Formulation

Our objective is to generate rules for a Network Intrusion Detection System (NIDS) based on video recognition of packet flow. Given a training dataset D, where $D = \{A_i | i$ represents the index of attack$\}$. Each attack A_i in the dataset consists of the corresponding rule R_i for the NIDS and several traffic data P_j^k, where j refers to the traffic data instance index and k represents the packet index of the traffic data P_j. The formula (1) is the structure of our data.

$$D = \{A_i | A_i = \{P_1, P_2, \ldots, P_N\}, where\, P_j = \{P_j^1, P_j^2, \ldots, P_j^m\}\} \tag{1}$$

We convert each packet P_j^k into an image representation, denoted as F_j^k. Furthermore, collecting packet images F_j as a Flow-Video can be interpreted as a traffic data flow. Our task aims to develop a function f that maps the Flow-Video F_j to the corresponding rule R_i, i.e., $R_i = f(F_j)$. This function f takes the frame representations as input and generates the NIDS rule associated with the traffic data flow.

Our task aims to develop a function f that maps the Flow-Video F_j to the corresponding rule R_i, i.e., $R_i = f(F_j)$. This function f takes the frame representations as input and generates the NIDS rule associated with the traffic data flow.

By solving this problem, we aim to provide an approach for generating NIDS rules based on packet video recognition, enabling more effective and adaptable intrusion detection in network security.

2.2 Rule Generator

T. Vollmer et al. [1] implement a Genetic Algorithm (GA) to create rules autonomously. N. Fallahi et al. [2] generated eight new features for flow-based intrusion detection. A. Sagala et al. [3] created and activated the Snort rule automatically based on the data sent by the honeypot server. Laryea et al. [4] train a neural network to generate partially effective malware rules for the Snort IPS.

2.3 Network Traffic Classification

Detecting abnormal traffic is crucial to Network Security, particularly when utilizing machine learning methods. Traffic Classification involves various techniques, but feature-based and image-based methods are two common ways to preprocess flow data. The following section will discuss the prior work on traffic classification.

Feature-Based. A flow consists of a group of packets within a time window, and the feature extractor extracts the features of the flow. Most studies used CIC-IDS2017 and CIC-IDS2018 [5] datasets with extracted features to classify whether the packets are benign or abnormal.

R. Li et al. [6] design a system that can classify a single packet without assembling IP packets into flows. M. Verkerken et al. [7] propose a novel multi-stage approach for hierarchical intrusion detection performing binary and multi-class detection, including known and zero-day attacks.

Image-Based. S. Potluri et al. [8] convert the features of the NSL-KDD dataset to a binary vector space and then represent the binaries into an 8×8 image. Identify different network attacks on ICS that mainly impact the security parameters' availability and confidentiality using CNN. H. -K. Lim et al. [9] treat the payload of packets as image data and use the convolutional neural network (CNN) and residual network (ResNet) to classify network traffic. K. Millar et al. [10] introduce flow-image representing network traffic and create a CNN model to exploit the known properties of TCP/IP packets.

2.4 Video Captioning

Video captioning generates textual descriptions for videos, conveying critical information without the need to watch. It involves analyzing video frames to extract visual features and identify objects, actions, and scenes. 3D CNN models such as C3D and R3D are commonly used for this task. These visual features are combined with language models, employing deep learning techniques like LSTM or transformers to generate accurate captions. Figure 1 illustrates the procedure of video captioning.

Visual Model. In Fig. 2 (a) C3D, D. Tran et al. [11] develop deep 3-dimensional convolutional networks (3D ConvNets) called C3D trained on a large-scale supervised video dataset. (b) R3D, Residual learning introduced skip-connected that directly connects earlier layers to the last layers. D. Tran et al. [12] proposed a 3D-convnets combined with residual learning called R3D. (c) R(2 + 1)D, in [12], the authors also proposed an architecture based on the theory that it may be more conveniently approximated by a 2D convolution followed by a 1D convolution called R(2 + 1)D.

Language Model. H. Sak et al. [13] present a recurrent neural network (RNN) type designed to capture long-term dependencies in sequential data and address the vanishing gradient problem using memory cells and gating mechanisms. LSTM can generate or predict the next word or sequence of words in a given text. The input to the LSTM language model is a sequence of words or characters, typically encoded as numerical representations.

Our problem can be seen as a video captioning task using Flow-Video as the input and the NIDS rule as the output. R. Pasunuru et al. [14] proposed a method for enhancing video captioning through multimodal, multi-task learning. They introduced temporally and logically directed knowledge into the process by incorporating video prediction and entailment generation tasks. Bairui Wang et al. [15] propose RecNet, an encoder-decoder-reconstructor architecture for video captioning. It utilizes forward and backward flows with specialized global and local structure restoration reconstructions. Wenjie Pei et al. [16] use ResNeXt-101 with 3D convolutions pretrained on the Kinetics dataset to extract 3D features and propose MARN, a Memory-Attended Recurrent Network for video captioning that utilizes a memory structure to enhance word understanding and improve captioning quality.

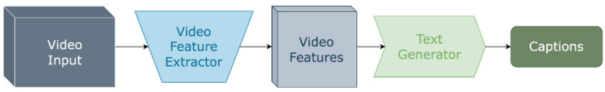


Fig. 1. Video captioning procedure

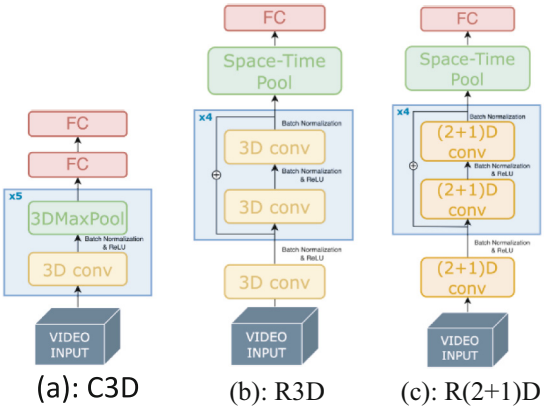


Fig. 2. Three type of 3D CNN model architecture

3 Methods

3.1 Encoding Network Traffic

In [10], the authors investigated the representation of network traffic as square matrix pixel images, specifically analyzing byte values in network packets. They successfully classified the UNSW-NB15 dataset using CNNs, like image classification techniques. Building upon their work, the authors further improved the process [17] by employing a One-to-Multi Mapping method to encode packet byte values. Inspired by this encoding scheme, we adopted it to treat traffic data as a video. We generated 512 images based on the packets, referred to as Flow-Video.

The following steps outline the process by which the scheme encodes network traffic, as depicted in Fig. 3.

Step 1: We apply a zero mask to the digits representing destination and source IP addresses.

Step 2: We extract each packet's hexadecimal digits from the packet byte values. For instance, referring to Fig. 3, the first-byte value of the network traffic is $0 \times 5c$. We split it into "5" and "c." Utilizing base-16 hexadecimal representation, "5" can be represented by the range 0×50 to $0 \times 5F$ (i.e., 80_{10} to 95_{10}), and "c" can be represented by the range $0 \times C0$ to $0 \times CF$ (i.e., 192_{10} to 207_{10}). We then map the midpoints of the two sub-ranges, 0×58 and $0 \times C8$, to two 2×2 sub-matrices.

Step 3: All sub-matrices are combined to form a matrix representing the entire packet.

To formulate the process, let $A = a_n \ldots a_1 a_0$ represent the byte value for a packet after masking the IP address, where n is the length of the packet byte value (limited to less than 1024). For the i^{th} digit a_i, , we apply the conversion (2) to obtain b_i.

$$b_i = f_c(a_i) = (a_i \times 16) + (16/2) \tag{2}$$

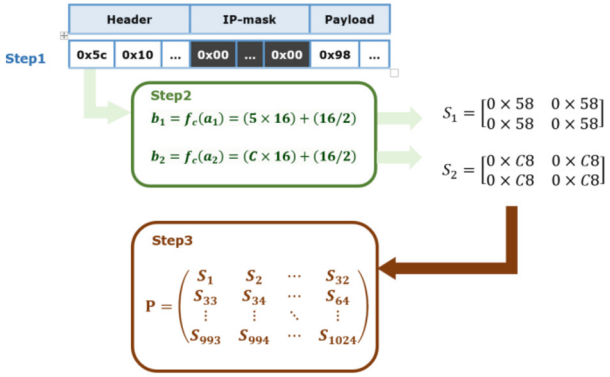


Fig. 3. Packet encoding procedure

After the conversion using f_c, A is transformed into B. We then repeat b_i to create a 2×2 sub-matrix:

$$S_i = [[b_i, b_i], [b_i, b_i]] \tag{3}$$

The matrix P is constructed as a 64×64 matrix comprising the sub-matrices S_i in (3):

$$P[s+2 : s+4, t : t+2] = S_i \tag{4}$$

where $s = i/64$ and $t = i \, mod \, 64$

For a network traffic flow $P_j = P_j^k$, we convert each packet P_j^k in P_j into a matrix using the abovementioned encoding. Consequently, we perceive a network traffic flow F_j, as a Flow-Video composed of several packet images, which will serve as the input for our subsequent model.

3.2 Architecture

We aim to generate one rule $R = \{w_1, w_2, \ldots, w_n\}$, where w_i represent the word of the rule to describe the given malicious packet flow F. To address the problem, we employ an encoder-decoder mechanism, where the encoder extracts the features of flow-video frames, and the decoder generates the IDS rule with frame features as input. The distribution of the output rule word $\{w_1, w_2, \ldots, w_{|R|}\}$ corresponding to the input:

$$P(R|F) = P\big(w_1, w_2, \ldots, w_{|R|}|F\big) = \prod_i P(w_i|w_{<i}, F; \theta) \tag{5}$$

where θ denotes the parameters of the encoder-decoder model, $|R|$ denotes the length of the words in the rule, $w_{(<i)}$ means the previous generated words (i.e., $\{w_1, w_2, \ldots, w_{(i-1)}\}$).

Equation (5) quantifies the likelihood of generating a specific word given the context of the previously generated words and the Flow-Video features. This probabilistic formulation ensures that the generated IDS rule aligns with the semantic and syntactic patterns observed in the training data.

Most image-text matching studies rely on pretrained models, like CLIP [18]. Unlike the studies, we transfer our packet byte values into pixel images as input. As a result, we can't just apply a pretrained model to solve the problem. Figure 4 shows the architecture we proposed. The following section will propose our customized encoder-decoder mechanism for the input.

Encoder: Feature Extractor. To caption well, we need to find out the important features of our flow-video. Unlike the studies using pretrained models, since we transfer our packet byte values into pixel images as input, we need to customize and train the encoder

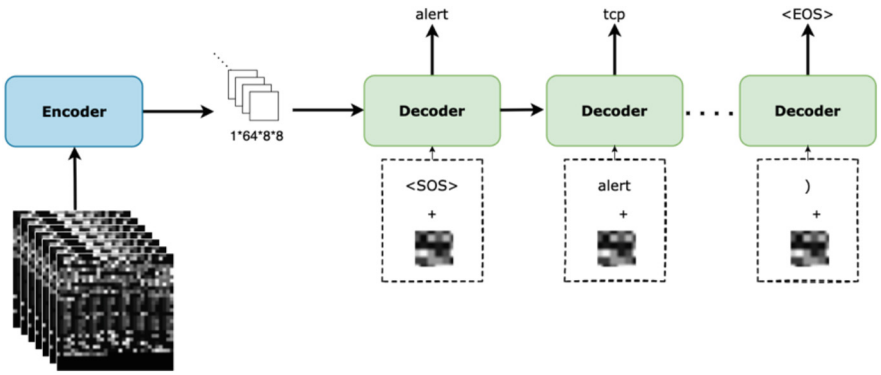


Fig. 4. Proposed **architecture**

ourselves. Consider the flow-video is a 3D-data, and use 3D-CNNs models, C3D, R3D, and R(2 + 1)D [11, 12], as our baseline model. Further, we reconstructed the 3D-CNNs model, called "PAC3D", to be more compatible with our pixel images input.

(2 + 1)D conv Block. To capture every packet information's temporal feature, we first employ spatial convolution to capture every packet's spatial feature:

$$Conv_{spatial} = y[i, j, k] = \sum_{n} \sum_{l} x[i, j - n, k - l] \cdot h[1, n, l] \tag{6}$$

where x represents input, y represents output, h represents kernel, i, j, k represent the position of the output element, and n, l represent offset within the kernel.

After obtaining spatial features, we do the temporal convolution to get the temporal correlation of these spatial features:

$$Conv_{temporal} = y[i, j, k] = \sum_{m} x[i - m, j, k] \cdot h[m, 1, 1] \tag{7}$$

where x represents input, y represents output, h represents kernel, i, j, k represent the position of the output element, and m represents offset within the kernel.

The (2 + 1)D conv block is the orange one shown in Fig. 5, and the formulation equation is following:

$$Conv_{2Plus1D} = Conv_{temporal}\big(Conv_{spatial}(x)\big) \tag{8}$$

Residual Block. The blue block in Fig. 5 represents the residual block, which introduces the residual network concept:

$$y = Conv(Conv(x)) + x \tag{9}$$

where x is the input of the residual block, and y is the output of the residual block.

3D Conv Block.

$$Conv_{3D} = y[i, j, k] = \sum_{m} \sum_{n} \sum_{l} x[i - m, j - n, k - l] \cdot h[m, n, l] \tag{10}$$

where x represents input, y represents output, h represents kernel, i, j, k represent the position of the output element, and m, n, l represent offset within the kernel.

After the convolutions, we performed global average pooling over the whole volume. And the output of our model is the classification result of the Flow-Video. Remove the fully connected layer; the result is the input of the language model introduced in the following paragraph.

Decoder: Language Generator. With the features of flow-video, we combine it with the rule word sequence as our decoder output. We use the LSTM network as our language model to generate the captions with its ability to maintain long-term dependencies. (11)–(16) show the LSTM operation.

$$f_s = \sigma\big(W_f[h_{s-1}, x_s] + b_f\big) \tag{11}$$

$$i_s = \sigma\left(W_i\left[h_{s-1}, x_s\right] + b_i\right) \tag{12}$$

$$o_s = \sigma\left(W_o\left[h_{s-1}, x_s\right] + b_o\right) \tag{13}$$

$$\widetilde{c_s} = tanh\left(W_c\left[h_{s-1}, x_s\right] + b_c\right) \tag{14}$$

$$c_s = f_s * c_{s-1} + i_s * \widetilde{c_s} \tag{15}$$

$$h_s = o_t * tanh(c_s) \tag{16}$$

where f_s represents forget gate, i_s represents input gate, o_s represents output gate, h_s represents the hidden state, h_{s-1} represents the output of previous cell, c_t represents cell gate, $\widetilde{c_s}$ represents the candidate for cell gate, W_g, b_g represent the weight and bias of respective gate g, x_s refers to the input of current step, and σ refers to sigmoid function. Figure 6 shows the LSTM cell.

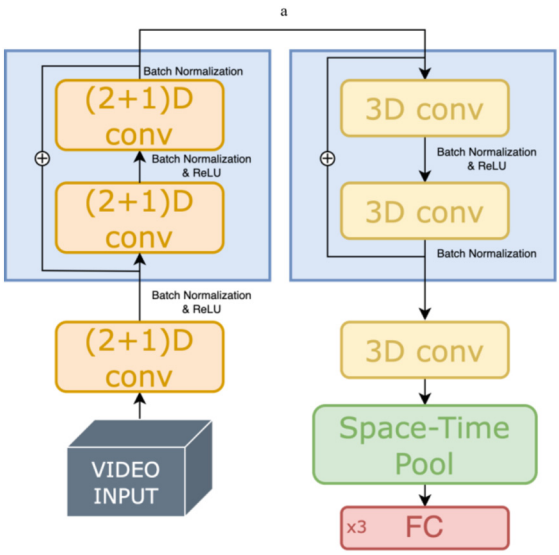


Fig. 5. PAC3D architecture

4 Implementation and Experiments

4.1 Dataset

Despite several existing datasets for IDS, such as CIC-IDS2017, CIC-IDS2018 [5], UNSW-15 [19], and NSL-KDD [20], these datasets lack sufficient information for conducting a comprehensive analysis of the task. One limitation is the absence of original

packet capture (PCAP) data. In contrast, another limitation is the lack of specific attack time information, making identifying the range of attack packets challenging. Consequently, we collected data, which involved capturing real-world PCAPs containing four different attacks and benign data. We used Wireshark, a packet capture tool, to collect the dataset to capture the network's packet flow. We set two VM, one with Kali Linux as the attacker OS and another with BeeBox (a custom Linux VM pre-installed with bWAPP) as the victim OS. Figure 7 illustrates the network structure of our environment. The data collection period lasts four days, starting from 17:00 on Wednesday, April 26, 2023, and concluding at 23:40 on Saturday, April 29, 2023. The implemented attacks included Brute Force SSH using Patator, DoS Goldeneye, DDoS LOIC, Heartbleed, and PortScan. For each attack category, we conducted the attack almost 200 times. Table 1 shows the records of times and numbers of attacks. Before each simulated attack, we captured normal traffic data for 1 min and continued capturing for an additional 1 min after the attack concluded. To streamline the data collection process, we developed scripts to automate the execution of the attacks and configured Wireshark to save the captured data at the designated times.

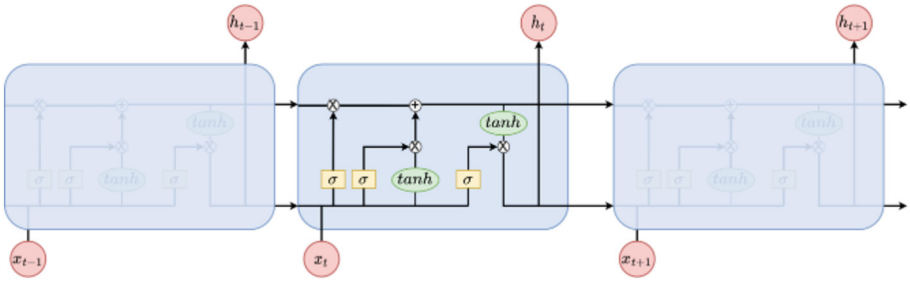


Fig. 6. LSTM cell

4.2 Implementation Details

Training Environment. We use a computer with NVIDIA GeForce RTX 3090 to train our models. Table 2 provides detailed information about our experimental environment.

Training Procedure and Evaluate Metrics. We train our model in two stages since we can't use a pretrained model and must train the feature extractor by ourselves.

Stage 1. Feature Extractor. We train our feature extractor as an attack type classifier using our dataset with the Flow-Video as input and the attack type as output. We employ Adam [21] gradient descent optimization to optimize our loss function Cross Entropy Loss (17) and perform training for 30 epochs with a learning rate decayed by 0.1 every ten epochs.

$$H(p, q) = -\Sigma_x(p(x) \log q(x) \tag{17}$$

where p(x) is our predicted attack type and q(x) is the actual attack type when x is input.

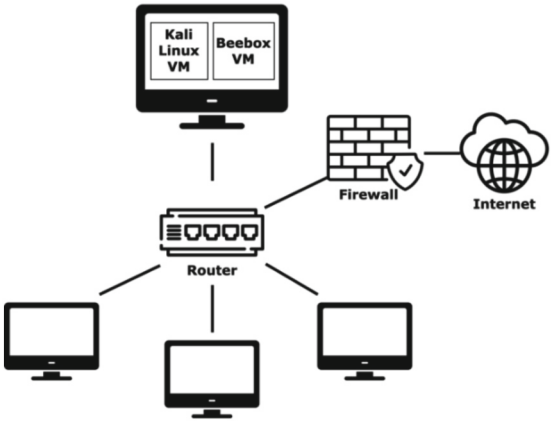


Fig. 7. The network structure of our environment.

Table 1. Number of each attack category sample

Attack Category	Time Range	Train	Validation	Test	**Total**
SSH Patator	4/26 17:00 - 4/27 08:00	128	32	40	200
DDoS GoldenEye	4/27 14:38 - 4/28 07:38	142	36	45	223
Heartbleed	4/28 22:48 - 4/29 15:00	116	29	37	182
LOIC	4/28 10:10 - 4/28 20:10	127	32	40	199
Total		513	129	162	804

Table 2. Experimental environment settings

Item	Setting
OS	Ubuntu 20.04.4 LTS
Program language	Python 3.7.16
CUDA/ cuDNN	11.6 / 8.2.1
TensorFlow	2.8.0
CPU	AMD Ryzen 9 5950X 16-Core Processor
GPU	NVIDIA GeForce RTX 3090
RAM	32GB

To evaluate our model, we use accuracy, precision, recall, and F1 with confusion matrixes to calculate and analyze.

$$Accuracy = \frac{|CorrectPredictions|}{|TotalPrediction|} \tag{18}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{19}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{20}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{21}$$

where all the metrics are as higher as possible.

Stage 2. Language Model. We use the pretrained model in Stage 1 and train our language model with Cross Entropy Loss as the loss function and Adam gradient descent as the optimizer. Perform training for 30 epochs with LSTM hidden size set as 256 and the word embedding size set as 256. After each LSTM layer in our language model, we applied dropout regularization with a probability of 0.5.

We employ two standard evaluation metrics in a captioning task to evaluate our language model. The following y denotes the actual word and y′ denotes the predicted word:

$$ROUGE_L = F_{LCS} = \frac{(1+\beta^2)R_{LCS}P_{LCS}}{R_{LCS}+\beta^2 P_{LCS}} \tag{22}$$

where $P_{LCS} = \frac{\mathrm{LCS}(y,y')}{\mathrm{len}(y)}$, $R_{LCS} = \frac{\mathrm{LCS}(y,y')}{\mathrm{len}(y')}$, LCS represents least common multiple.

$$BLEU = BP \times exp\left(\sum_{n=1}^{N} W_n \times \log P_n\right) \tag{23}$$

where $\mathrm{BP} = \min\left(1, \exp\left(1 - \frac{actuallength}{gerneatedlength}\right)\right)$, $\mathrm{W_n}$ represent the weight and $\mathrm{P_n} = \frac{\{\text{number of n}-\text{gram matches}\}}{\{\text{number of total n}_{\text{grams}} \text{in gerneated output}\}}$.

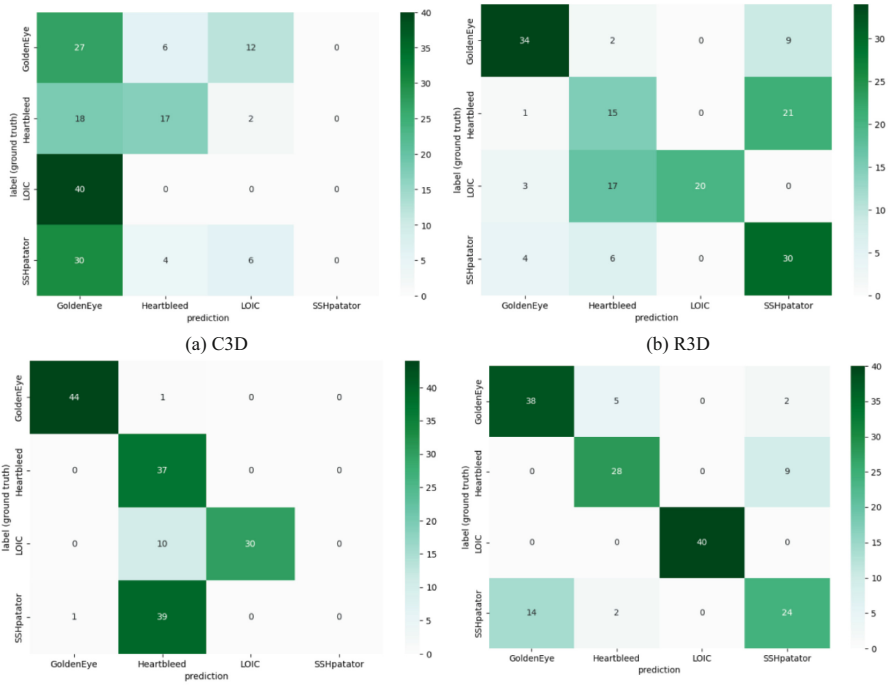
4.3 Experiments and Discussion

During our experiment, we examined input data sizes as 512 packets to generate captions for the resulting rules. Initially, we compared the performance of our feature extractor "PAC3D" as a classification model to baseline models C3D, R3D, and R(2 + 1)D. Additionally, we evaluated their performance when paired with the same language model. The accuracy results presented are an average performance from ten training runs. And the other metrics show the best model performance.

Classification Model. Table 3 shows the results of testing the baseline model: C3D, R3D, and R(2 + 1)D on our dataset. From the result, we can observe that our model "PAC3D" have higher accuracy than the other three models. However, accuracy may not be the only evaluation we should consider. We also value the trade-off of precision and recall, where are average scores of each class. Our model also has the most balance between Precision and Recall. Figure 8 indicates the confusion matrixes of the four models with the highest accuracy in each ever tested. A comparison of the four results reveals that our model can averagely classify the attack categories well.

Table 3. The results of attack classification model (encoder)

Model Name	Accuracy	Precision	Recall	F1
C3D	0.24	0.21	0.26	0.23
R3D	0.58	0.63	0.60	0.62
R(2 + 1)D	0.64	0.60	0.68	0.64
PAC3D(proposed)	**0.76**	**0.80**	**0.80**	**0.8**



(a) C3D (b) R3D

(c) R(2+1)D (d) PAC3D

Fig. 8. The confusion matrixes (a) C3D (b) R3D (c) R(2 + 1)D (d) PAC3D

Captioning. In Table 4, we can observe that our model demonstrates superior captioning performance based on both evaluation metrics. However, it is important to note that the BLEU@1 scores appear to be relatively low for all models. This is primarily because the BLEU@1 metric is designed to measure similarity between short and simple sentences, whereas our objective is to generate complex and machine-readable network intrusion detection system rules. Therefore, the low BLEU@1 scores do not necessarily indicate poor performance in our specific task. Instead, they reflect the inherent challenges of evaluating rule-based captions, which tend to be longer and less human-readable.

Table 4. The results of captioning model with different classifiers

Model Name	BLEU@1	ROUGE
C3D + LSTM	\	\
R3D + LSTM	0.13	0.61
R(2 + 1)D + LSTM	0.13	0.77
PAC3D + LSTM (proposed)	0.11	0.89

Furthermore, in Table 5, we present a comparison between the target rules and the generated rules. The results indicate that the generated rules align perfectly with the target rules, providing evidence that our model captures the desired correlations effectively. However, this also suggests that our dataset might be too small for the captioning task, as the generator can easily capture the word correlations within the limited data.

Table 5. The comparison of the target rules and the generated rules

Target rules	Generated rules
alert tcp $EXTERNAL_NET any - > $HOME_NET any (msg:"ET DOS Inbound Low Orbit Ion Cannon LOIC DDOS Tool desu string"; flow:to_server,established; content:"desudesudesu"; nocase; threshold: type limit,track by_src,seconds 180,count 1; reference:url,www.isc.sans.org/diary.html?storyid=10051; classtype:trojan-activity; sid:2012049; rev:5; metadata:created_at 2010_12_13, updated_at 2010_12_13;)	alert tcp $EXTERNAL_NET any - > $HOME_NET any (msg:"ET DOS Inbound Low Orbit Ion Cannon LOIC DDOS Tool desu string"; flow:to_server,established; content:"desudesudesu"; nocase; threshold: type limit,track by_src,seconds 180,count 1; reference:url,www.isc.sans.org/diary.html?storyid=10051; classtype:trojan-activity; sid:2012049; rev:5; metadata:created_at 2010_12_13, updated_at 2010_12_13;)
alert tcp $EXTERNAL_NET any - > $HOME_NET 22 (msg:"ET SCAN Potential SSH Scan"; flow:to_server; flags:S,12; threshold: type both, track by_src, count 5, seconds 120; reference:url,en.wikipedia.org/wiki/Brute_force_attack; reference:url,doc.emergingthreats.net/2001219; classtype:attempted-recon; sid:2001219; rev:20; metadata:created_at 2010_07_30, updated_at 2010_07_30;)	alert tcp $EXTERNAL_NET any - > $HOME_NET 22 (msg:"ET SCAN Potential SSH Scan"; flow:to_server; flags:S,12; threshold: type both, track by_src, count 5, seconds 120; reference:url,en.wikipedia.org/wiki/Brute_force_attack; reference:url,doc.emergingthreats.net/2001219; classtype:attempted-recon; sid:2001219; rev:20;metadata:created_at 2010_07_30, updated_at 2010_07_30;)

5 Conclusions and Future Work

We employed a method to transform packet information into a video format and devised an architecture for generating rule captions for the network intrusion detection system using video-captioning techniques. Our proposed model incorporates spatial-temporal and comprehensive information in the encoder component, resulting in superior performance for our specific use case. Furthermore, we conducted a comparative analysis with a commonly used video captioning model to highlight the advantages of our approach.

To evaluate our model, we created a dataset by simulating various attack scenarios and performed experiments using this dataset. The results presented in Sect. 4.3 demonstrate

that our model outperforms C3D, R3D, and R(2 + 1)D models in terms of performance on our dataset.

Unlike other captioning models that benefit from large pretraining datasets and can handle multiple tasks such as QA, translation, and captioning for unseen classes, our architecture has limitations due to using a smaller dataset. However, we believe that with the collection of a larger dataset, our model can excel in identifying zero-day attacks and generating network intrusion detection system rules specifically tailored to those attacks. By expanding the dataset, our model will have the potential to overcome its cur-rent limitations and achieve improved performance in capturing and addressing new and emerging security threats.

References

1. Vollmer, T., et al.: Autonomous rule creation for intrusion detection. Proc. IEEE Symposium on Computational Intelligence in Cyber Security (CICS) **2011**, 1–8 (2011)
2. Fallahi, N., et al.: Automated flow-based rule generation for network intrusion detection systems. In: Proc. 2016 24th Iranian Conference on Electrical Engineering (ICEE), pp. 1948–1953 (2016)
3. Sagala, A.: Automatic SNORT IDS rule generation based on honeypot log. In: Proc. 2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 576–580 (2015)
4. Laryea, E.N.A.: Snort Rule Generation for Malware Detection using the GPT2 Transformer. Université d'Ottawa/University of Ottawa (2022)
5. Sharafaldin, I., et al.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization (2018)
6. Li, R., et al.: Byte Segment Neural Network for Network Traffic Classification. In: Proc. 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), pp. 1–10 (2018)
7. Verkerken, M., et al.: A novel multi-stage approach for hierarchical intrusion detection. IEEE Trans. Netw. Serv. Manage. **20**(3), 3915–3929 (2023). https://doi.org/10.1109/tnsm.2023.3259474
8. Potluri, S., et al.: Convolutional Neural Networks for Multi-class Intrusion Detection System, pp. 225–238. Springer International Publishing (2018)
9. Lim, H.-K., et al.: Packet-based network traffic classification using deep learning. In: Proc. 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), IEEE, pp. 046–051 (2019)
10. Millar, K., et al.: Using convolutional neural networks for classifying malicious network traffic. In: Alazab, M., Tang, M. (eds.) Deep Learning Applications for Cyber Security, pp. 103–126. Springer International Publishing (2019)
11. Tran, D., et al.: Learning Spatiotemporal Features with 3D Convolutional Networks. Proc. IEEE International Conference on Computer Vision (ICCV) **2015**, 4489–4497 (2015)
12. Tran, D., et al.: A closer look at spatiotemporal convolutions for action recognition. Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition **2018**, 6450–6459 (2018)
13. Sak, H., et al.: Long short-term memory recurrent neural network architectures for large scale acoustic modeling (2014)
14. Pasunuru, R., Bansal, M.: Multi-task video captioning with video and entailment generation. arXiv preprint arXiv:1704.07489 (2017). https://doi.org/10.48550/arXiv.1704.07489
15. Wang, B., et al.: Reconstruction Network for Video Captioning. Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition **2018**, 7622–7631 (2018)

16. Pei, W., et al.: Memory-Attended Recurrent Network for Video Captioning. Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) **2019**, 8339–8348 (2019)
17. Cheng, A.: PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks. In: Proc. 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 0728-0734 (2019)
18. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: Proc. International conference on machine learning, pp. 8748–8763. PMLR (2021)
19. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). Proc. Military Communications and Information Systems Conference (MilCIS) **2015**, 1–6 (2015)
20. Tavallaee, M., et al.: A detailed analysis of the KDD CUP 99 data set. Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications **2009**, 1–6 (2009)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv: 1412.6980 (2014). https://doi.org/10.48550/arXiv.1412.6980