An Interpretable Multi-Modal Transformer-Based Intrusion Detection System Utilizing Log Messages and PCAP Files

Nadia Niknami*, Vahid Mahzoon†, Rajorshi Biswas‡, Slobodan Vucetic†, and Jie Wu§
* *Villanova University*, Department of Computing Sciences, Villanova, USA
† *Temple University*, Center for Hybrid Intelligence, Philadelphia, USA
‡ *Penn State University(Berks)*, Information Sciences and Technology, Reading, USA
§ *Temple University*, Center for Networked Computing, Philadelphia, USA

***Abstract*—Intrusion detection systems (IDS) primarily rely on signature-based approaches, which can fail to detect novel or sophisticated attacks. This paper addresses the underutilized potential of leveraging a multi-modal approach that combines packet capture (PCAP) and log data for anomaly detection. To enhance detection capabilities, we propose an interpretable hybrid neural network architecture, TransIDS, that integrates a packet-based transformer with an efficient transformer-based language model for log messages. The proposed framework extracts semantic vectors from raw log messages and concatenates them with packet embeddings. An attention-based classification model then detects anomalies by determining the importance of each log message and packet for the neural network's decision. By fusing spatial features from PCAP data with temporal features from log data, TransIDS utilizes this multi-modal data fusion to identify anomalies that might be missed by conventional systems. This approach not only leverages the strengths of two distinct transformer-based architectures but also provides a more comprehensive analysis of network traffic, leading to more effective detection of previously undetected attacks and strengthening overall network security. We use a real testbed for our experiments to validate the effectiveness of our proposed approach.**

***Index Terms*—Intrusion Detection System(IDS), Natural Language Processing, Multi-Modality, PCAP, Transformer-based IDS**

I. Introduction

Intrusion Detection Systems (IDS) play a crucial role in maintaining the security and integrity of networks [1]. These systems are designed to monitor network traffic for suspicious activity and potential threats, such as unauthorized access attempts, malware, and other forms of cyberattacks. By analyzing data packets, an IDS can detect patterns that suggest malicious activity, allowing network administrators to take prompt action to mitigate risks [2], [3]. One of the key outputs of IDS is the generation of logs, which are detailed records of events and activities observed within the network [4]. In service and system management, logs play an essential role in serving as a vital tool to capture significant events and runtime information [5].

Traditional log anomaly detection methods typically depend on techniques like keyword matching or regular expressions to identify abnormal log entries. However, these approaches can struggle to detect attacks or anomalies involving logs from various systems or logs with differing formats and structures. Effective anomaly detection from system logs faces several significant challenges including understanding the intricate attributes present in event logs, extracting complex contextual relationships among events, and providing clear explanations to human analysts regarding detected anomalies. As a result, there is a critical need for efficient and automated anomaly detection systems capable of rapidly identifying and resolving potential issues in network infrastructures through log analysis.

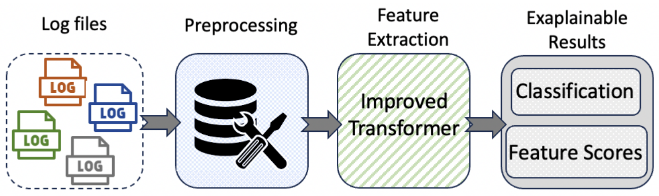


Fig. 1: Log-based intrusion detection.

Log-based intrusion detection [6] involves a systematic process to identify malicious activities within a network by analyzing log files. As depicted in Fig. 1, this process begins with the collection of various log files from various sources. These log files undergo pre-processing to standardize and organize the data, making them ready for further analysis. Following preprocessing, feature extraction is conducted using an improved transformer model, which identifies relevant features and patterns indicative of potential security threats. The extracted features are then used for classification, enabling the detection of anomalies and intrusions. In addition, feature scores are generated to improve the interpretability of the detection outcomes.

Transformers [7], initially designed for natural language processing (NLP) tasks, are highly effective at capturing long-term relationships in data and efficiently processing long sequences without extensive pre-processing. This ability to model extended sequences with greater efficiency has made transformers a powerful tool not only in NLP but also in time-series anomaly detection. Typically, transformers are pretrained on extensive text corpora and subsequently fine-tuned on smaller, task-specific datasets. Beyond NLP, transformers have proven computationally efficient and scalable, extending their utility to computer vision, particularly in image classification. By leveraging the capabilities of the Transformer encoder [8], [9], we can extract both time series and global features from the data, resulting in significant improvements in classification performance. In light of the above, this paper introduces a log-based IDS that combines multiple transformers for effective attack detection. By leveraging both log and PCAP data, the system implements a multi-modal [10] approach, utilizing the strengths of each data type. Our multi-modal Transformer-based framework employs an attention mechanism to determine the significance of each log message and packet in the decision-making process, ensuring greater transparency in detection outcomes. To further enhance interpretability, feature scores are generated, offering deeper insights into the detection process.

The main contributions of this work are as follows:

- We propose a multi-modal Transformer-based framework that uses the computational efficiency and scalability of different types of Transformers to detect attacks.
- We develop a log-based IDS by utilizing a packet-based transformer that outperforms the traditional CNN-Transformer combination.
- To enhance the system's capability to detect unknown attacks and minimize false alarms, we employ two distinct types of input: log and PCAP data, leveraging the strengths of a multi-modal approach.
- Our approach provides interpretability by using an attention-based mechanism to determine the importance of each log message and packet for the system's decision-making.

II. Related Work and Background

In this section, we delve into various approaches and methodologies employed in the field of log-based attack detection, deep learning-based attack detection, and log anomaly detection based on Transformer models.

A. Log-based Attack Detection

IDSs generate output in the form of logs, which are essential for documenting various events and activities within a network. These logs provide detailed records of network traffic, including alerts about potential security incidents. By analyzing log entries, anomalies can be identified, and alerts can be triggered for early notification of potential issues. Logs serve as valuable resources for the timely detection of system anomalies, with output logs offering a potential means to identify abnormal system states [11]–[14].

Traditional log anomaly detection relies on domain expertise, manual checks, and rule-based systems, which are costly, inflexible, and limited by expert knowledge. Machine learning models also struggle, as they focus on simple features and fail to capture complex log patterns, limiting their effectiveness. To overcome these limitations, more advanced techniques such as deep learning and NLP are being explored. For instance, LogRobust employs an attention-based Bi-LSTM network to detect anomalies in unstable log data [15]. Anomaly detection methods based on logs are classified into graph model-based, probability analysis-based, and machine learning-based approaches. Graph model-based methods capture sequence relationships, association relationships, and log text content. Probability analysis-based methods utilize correlation analysis and comparison to calculate the correlation probability between logs and anomalies. Machine learning-based methods often employ LSTM models to infer the abnormality of logs by examining sequential relationships [16].

B. Deep Learning-based Attack Detection

In sequence-level anomaly detection, employing RNN-based deep learning models like LSTM and GRU is a common approach [17]–[20]. The Transformer model was initially created to handle sequence-to-sequence tasks, such as language translation. It operates by taking in a sequence of any length, applying a self-attention mechanism to understand the context, and generating an output sequence based on this learned context. The process begins with encoding, where the model grasps the significance of the input sequence. Then, it decodes this learned context into the output sequence. This study focuses solely on utilizing the transformer encoder as a substitute for LSTM in the context of anomaly detection [7].

Transformers have gained traction for attack detection, with studies combining them with CNNs to detect DDoS attacks [21] or leveraging their self-attention mechanism for robust intrusion detection solutions [22]. The original Transformer model comprises both an encoder and a decoder. However, for intrusion detection tasks, which predominantly involve classification, this module exclusively utilizes the encoder component. Transformers, known for their ability to model contextual dependencies, have also been applied to log anomaly detection tasks. HitAnomaly, for example, employs a hierarchical Transformer structure to encode log sequences and parameter values [23].

Approaches like DeepLog focus on achieving high accuracy in streaming fashion for log anomaly detection, using LSTM to capture latent patterns from normal log sequences [16], [24]. Similarly, LogAnomaly utilizes LSTM to capture normal patterns while introducing a novel template2Vec method to represent log templates [12]. TRANSLOG, a unified Transformer-based framework, comprises a pretraining stage and an adapter-based tuning stage for log anomaly detection [25]. In contrast, Loader utilizes the Transformer encoder for semi-supervised log anomaly detection, exclusively relying on normal log sequences for training .Additionally, Loader introduces a flexible and robust top-p algorithm for candidate set determination during detection [26].

In contrast, our proposed multi-modal Transformer-based framework addresses these gaps by combining both log and PCAP data, thus providing a more comprehensive analysis of network anomalies. Furthermore, our framework introduces an attention-based mechanism that improves interpretability by

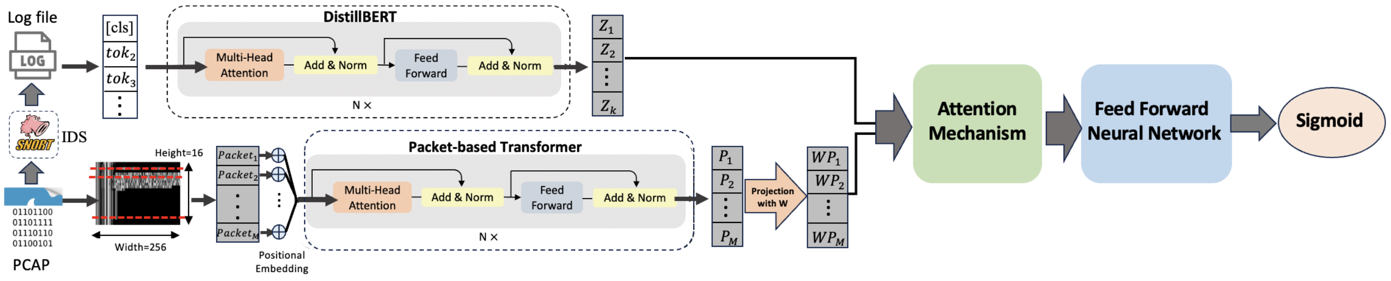


Fig. 2: TransIDS framework with DistillBERT and packet-based Transformer.

determining the importance of each log message and packet for the decision-making process, a feature often missing in prior methods.

III. Methodology

PCAP file contains network traffic information and is used to assess the underlying data exchange between servers. However, such data is noisy and does not carry meaningful semantics for the detection system. We can integrate some auxiliary information to preserve actual semantics. To integrate network traffic with the alert messages of the IDS, we need to merge the two types of input data: the PCAP file and the log file. The IDS monitors all network traffic and generates logs including alerts. The IDS analyzes the behavior of each traffic flow. Simultaneously, traffic data values are fed to the framework to aid in traffic behavior analysis. Consequently, our framework can detect attacks based on both the attack behavior and its impact. The selected traffic data should be within the same time window as the duration of the attack detection, which is typically a very short period.

We utilized transformer-based models for processing both log contents and packets within a PCAP file, as shown in Fig. 2. Transformers consist of sequential blocks, each applying a transformation to the input sequences. The attention mechanism within Transformers captures long-range dependencies, semantic (contextual) dependencies in logs and spatial in packets which leads to more accurate classifier. By computing the relevance between different parts of a sequence, attention allows the model to focus on critical components. Attention computes the relevance between different parts of a sequence. The multi-head self-attention applies multiple self-attention functions to its input in parallel, each of which is called a head, then concatenates all heads' outputs and projects them to the same d-dimensional space as the input. The results obtained by each head are concatenated together to construct the final result. This mechanism enables the model to focus on different positions and provides multiple representation subspaces for the attention layer.

Compared to previous works that primarily use CNNs to capture patterns from packets, we opted for a packet-based transformer inspired by vision transformer. This approach allows us to obtain an embedding for each packet and assess its importance through an attention mechanism. In contrast, CNNs tend to lose local information about each packet as they focus on extracting global patterns.

We combined two transformer-based models, DistilBERT and the Packet-based Transformer, to create a hybrid and interpretable model named TransIDS, designed to achieve superior performance compared to traditional deep learning models and conventional hybrid approaches. TransIDS automatically learns spatio-temporal features of network traffic using deep neural networks, making it well-suited for interpretable anomaly detection. The pre-processed traffic sequence serves as the input for Transformer NIDS. A structural diagram of the proposed model is shown in Fig. 2.

- **Module 1: (Data Preprocessing)** The goal of this module is ensuring that both data modalities are prepared for passing through the deep learning models to obtain meaningful embeddings.
- **Module 2: (BERT-based embedding for log sequence)** The goal of this module is to provide a vector representation for log content with the help of the BERT model.
- **Module 3: (Transformer-based embedding for PCAP sequence)** The goal of using transformer is to provide the temporal and structural patterns features in the packet data and derive their embeddings.
- **Module 4: (Decoder)** Attention mechanism integrates the representations of log events and PCAP data. The goal is to focus on the most relevant parts of the sequence and determine the importance of each modality (PCAP or log sequence) and specific segments within the log sequence for the classification task.

A. Data Preprocessing

Raw log messages are unstructured, which contain many different format texts. The original transformer model does not consider a special feature of the raw log template but only considers the self-attention relationship matrix of the raw log sequence itself making a lot of important information ignored, which may lead to false alarm.

Data preprocessing includes *Log Parsing* and *PCAP Parsing*. We applied regex to extract various entities from the raw log messages, creating a more structured and organized format. In order to tackle log parsing errors or information loss problems, this paper preprocesses the raw log message data without log parsing to delete non-character parameters and obtain the log content ℓ. We will then tokenize the log message and feed it into a BERT-based model, which will be explained in the next section. Additionally, we must preprocess the associated PCAP format for each log message.

Algorithm 1 TransIDS

```
1: Input $(X, Y)$: Dataset where each $X$ contains a PCAP and a Log message
2: Output Trained classifier with parameters $\Theta$
3: Initialize trainable parameters
       $\Theta = \{W_1, W_2, L, V, \text{Packet-Transformer parameters}\}$
4: Freeze pretrained DistilBERT parameters
5: $L_{\text{total}} \leftarrow 0$                          ▷ Initialize total loss
6: while Accuracy is improving do
7:     for each $(\text{PCAP}, \text{Log}, Y)$ in dataset $(X, Y)$ do
8:         $L \leftarrow \text{DistilBERT(Log)}$
9:         $P \leftarrow \text{Packet-Transformer(PCAP)}$
10:        $P' \leftarrow P \cdot W$
11:        $H \leftarrow [P'; Z]$                    ▷ Concatenate embeddings
12:        $B \leftarrow \tanh(H \cdot V)$
13:        $A \leftarrow \text{softmax}(B \cdot W)$          ▷ Find importance weights
14:        $D \leftarrow A^T \cdot H$
15:        $P_i \leftarrow \text{Sigmoid}(\langle L, D \rangle)$
16:        $L_{\text{total}} \leftarrow L_{\text{total}} + \text{CrossEntropy}(P_i, Y)$
17: Update $\Theta$ using Adam optimizer to minimize $L_{\text{total}}$
```

During this step, we format the packets to make them suitable for the model and apply normalization to the packet data. This step ensures that both data modalities are prepared for passing through the deep learning models to obtain meaningful embeddings (vector representations).

B. BERT-based Embedding

We apply *DistilBERT* [27], a lightweight version of BERT model, on preprocessed log sequence to obtain a vector representation for log sequence. We leverage the capabilities of Bidirectional Encoder Representations from Transformers (BERT), a powerful language model that excels in understanding text semantics. BERT is utilized to analyze the semantic similarity between different alert messages. This process involves converting the alert messages into rich embeddings through several steps: tokenization, initial embedding via the embedding layer, positional embedding, multi-head attention, and feed-forward layers to capture the sequential nature of the data.

1) Tokenization and Vector Representation: BERT processes an input sequence by tokenizing it using WordPiece [28] and adding special tokens, such as [CLS] at the beginning of each sequence. Each token is then converted into a dense vector representation through WordPiece embeddings , using a vocabulary of 30,000 tokens.

2) Positional Embedding: The BERT model utilizes the transformer architecture [7], which is permutation invariant, meaning the order of input tokens does not affect the model's output. To provide the model with information about the position of each token, BERT employs positional encoding. This technique incorporates the position of tokens using sinusoidal functions, defined as:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{1}$$

$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right), \tag{2}$$

where pos is the position of the token, i is the dimension index and d is the dimension of the embeddings. Obtained embeddings from the first two steps are fed into BERT's architecture, enabling it to capture rich contextual relationships within the text.

3) Multi-Head Attention: The multi-head attention mechanism allows the model to focus on different parts of the alert message simultaneously. We denote the obtained embedding of one alert message from the first two steps by $X \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the embedding dimension. BERT consists of multiple layers of multi-head attention, in which a new embedding for each token is obtained. We denote the final embedding for layer l as $E^{(l)}$. In each head of the first multi-head attention layer, initially three different representations of X are obtained, called Q (Query), K (Key), and V (Value), through linear projection using three trainable matrices W_i^Q, W_i^K, W_i^V with dimensions $\mathbb{R}^{d \times d_k}$ as follows:

$$Q = XW_i^Q, \quad K = XW_i^K, \quad V = XW_i^V. \tag{3}$$

Then, for each head:

$$\text{head}_i = \text{softmax}\left((QW_i^Q)(KW_i^K)^T/\sqrt{d_k}\right) VW_i^V. \tag{4}$$

Next, the heads are concatenated and multiplied by another trainable matrix W^O:

$$\text{MultiHead} = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O. \tag{5}$$

In order to obtain the final embedding, residual connection [29], layer normalization (LN) [30] and Feed Forward Neural Network (FNN) have been utilized as follows:

$$E^{(1)} = \text{LN}(\text{FNN(out)} + \text{out}), \tag{6}$$

where $E^{(1)}$ is the final embedding for the first multi-head attention layer and out $= \text{LN}(\text{MultiHead} + X)$. This new embedding will then serve as the input for the second multi-head attention layer, and so on. Now we need to elaborate on how DistilBERT is distilled from BERT. Knowledge distillation is a compression technique in which a compact model—the student—is trained to reproduce the behavior of a larger model—the teacher. The student is trained with a distillation loss over the soft target probabilities of the teacher. The final training objective is a linear combination of the distillation loss L_{ce} with the supervised training loss, in our case the masked language modeling loss.

C. Transformer-based Embedding for PCAP

The packet-based Transformer component for analyzing log files in the TransIDS framework starts with the PCAP file, which contains network packet data in binary form. Network traffic flow is defined as the volume of data transmitted between two communication nodes over a specific period.

Each network flow comprises sequential packets. Each packet, regardless of the network protocol it follows, includes one or more headers and payloads of certain bytes. Since a network flow consists of several bytes, it is analogous to a pixel in a grayscale image. This insight led us to transform network flow data into grayscale images, where each byte of the network flow represents a single pixel in the resulting image-like data.

We propose a novel encoding method called Grayscale Flow to convert network flow data into a grayscale image-like representation. Since the encoded data resembles grayscale images, we use a 1D byte array to represent a single packet and append each 1D array, corresponding to packets belonging to the same flow, in chronological order to construct a 2D representation. In other words, we arrange the content of each packet as a row in the output matrix.

The PCAP data is transformed into an image-like representation, where each packet is depicted as a matrix with a specific height (16) and width (256). Each row in this matrix corresponds to a feature vector extracted from the packet. The packet images are fed into a transformer model inspired by vision transformer model, which consists of multiple layers of processing. After the transformer layers, the packets ($Packet_1$, $Packet_2$, ..., $Packet_M$) are combined with positional embeddings, which encode the position of each packet in the sequence. The obtained embedding for the packets are then projected using a matrix (W) to transform the features into a new space, similar to the embedding size in BERT's embedding space, resulting in the vectors WP_1, WP_2, ..., WP_M.

D. Attention Mechanism and Classification

The attention mechanism is designed to focus on the most relevant parts of the input data when making a prediction. It assigns different levels of importance (weights) to various features, packet and log token embeddings, allowing the model to prioritize critical information.

The attention mechanism takes as input two sets of features. The first set of features comes from the DistilBERT model, which processes log file data. The second set comes from the Packet-based Transformer, which processes the packet data from PCAP files. The attention mechanism processes the features from the DistilBERT output (log file features) and the Packet-based Transformer (packet features) by computing relevance scores for each feature in relation to the entire sequence of features. The attention mechanism then creates a weighted combination of these features, where the weights are determined by the computed relevance scores. Features with higher relevance scores will contribute more to the final combined feature set. The output is a single, combined feature set that integrates information from both the log files and the packet data. This combined feature set captures the most critical patterns and interactions between the log and packet data. We provide a detailed mathematical explanation of attention mechanism in Section III-E.

Subsequently, the unified embeddings will be passed through a Feedforward Neural Network (FNN) to obtain a probability distribution over all log templates, indicating the likelihood of each template appearing after the input log sequence. The FNN is a type of neural network where the data moves in one direction, from input to output, through multiple layers of processing. The FNN consists of several layers, where each layer applies a series of transformations to the input data, such as linear combinations and nonlinear activation functions. These transformations help the network learn complex patterns and relationships within the data. The final layer of the FNN is connected to a Sigmoid activation function. The sigmoid function is used because it squashes the output into a range between 0 and 1, making it suitable for binary classification tasks (e.g., distinguishing between normal and malicious traffic). The sigmoid output can be interpreted as a probability score. For example, a score close to 1 might indicate a high likelihood of an attack, while a score close to 0 suggests normal behavior. The model's final output is the probability that the input data (both log files and packet data) belongs to a certain class (e.g., an attack or normal traffic). This probability is then used to make a classification decision.

E. Model Formulation

In this section, we present a precise mathematical formulation of our model, detailing the various modules that constitute our methodology. This formulation will clarify the structure and interactions between the components, providing a comprehensive understanding of how each part contributes to the overall approach. For each data point, we have an associated log message and a PCAP file. The log, after tokenization, consists of N tokens with each token having a dimensionality of h. We denote the tokenized log as follows:

$$L = \left[\text{tok}_{CLS}, \text{tok}_2, \ldots, \text{tok}_N\right] \in \mathbb{R}^{N \times h} \tag{7}$$

The PCAP file comprises M packets, each of size 256 bytes, which we represent as:

$$\text{PCAP} = \left[p_1, p_2, \ldots, p_M\right] \in \mathbb{R}^{M \times 256} \tag{8}$$

The tokenized log L is then fed into the DistilBERT model, while the PCAP file is input into a Packet-based Transformer. The resulting embeddings for the log tokens and packets are computed as follows:

$$Z = \text{DistilBERT}(L) \in \mathbb{R}^{N \times E} \tag{9}$$

$$P = \text{Packet-Transformer}(\text{PCAP}) \in \mathbb{R}^{M \times E'}, \tag{10}$$

where E and E' denote the embedding dimensions produced by the DistilBERT and Packet-based Transformer models, respectively. To align the embedding space of the packets and log tokens and ensure they share the same dimensionality, we project the packet embeddings using a trainable matrix W_1:

$$P' = PW_1 \in \mathbb{R}^{M \times E} \tag{11}$$

We then concatenate the projected packet embeddings P' and the log token embeddings Z:

$$H = \left[P'; Z\right] \in \mathbb{R}^{(M+N) \times E} \tag{12}$$

Next, the concatenated embeddings H are passed through an attention mechanism. The attention mechanism computes the importance weights A as follows:

$$B = \tanh(HV) \in \mathbb{R}^{(M+N)\times E} \tag{13}$$

$$A = \text{softmax}(BW_2) \in \mathbb{R}^{(M+N)\times 1} \tag{14}$$

Here, V and W_2 are trainable matrices. The i-th element of A represents the importance of each log token and packet, and the softmax operation normalizes these importance scores to form a distribution across all tokens and packets. The weighted sum of the matrix H, based on the importance weights A, yields a specific representation $D = A^T H \in \mathbb{R}^E$.

Finally, the representation D is used to predict the probability P_i of the flow being classified as an attack:

$$P_i = \text{sigmoid}(\langle L, D\rangle), \tag{15}$$

where L is a trainable vector and $\langle\cdot,\cdot\rangle$ denotes the inner product. Based on a predefined threshold θ, we can assign a label to the flow. This label indicates whether the flow is classified as normal or as an attack.

IV. Evaluation

In this section, we evaluate our model's performance in binary and multi-class attack classification, as well as in zero-day attack scenarios, comparing it against several baselines. Additionally, we assess the interpretability of our model by demonstrating how it focuses on different parts of the input modalities for some representative flows. We conducted our experiments on a system equipped with an Intel(R) Xeon(R) W-2225 CPU @ 4.10GHz, featuring an $x86\_64$ architecture with 8 CPUs, each with 2 threads per core and 4 cores per socket, reaching a maximum frequency of 4.6 GHz. The system is also equipped with a NVIDIA GeForce RTX 2080 SUPER graphics card. We conduct each experiment multiple times and then calculate the average to present the results. All experiments were conducted using PyTorch. Seed values were fixed to ensure reproducibility of the results. Training was primarily performed on the GPU to leverage its computational efficiency. We conducted a grid search to tune the learning rate, batch size, and number of attention heads. The optimal configuration was selected based on the highest validation accuracy. The model was optimized using binary cross-entropy loss with the Adam optimizer, with a learning rate of 5×10^{-5}. Regarding the packet-based transformer, the hidden dimension was set to $E' = 256$, and we used 2 layers with 4 attention heads per layer. The ReLU activation function was employed throughout the model, which was trained in batches of 32 samples. To prevent overfitting, early stopping was applied. Training was halted after 400 epochs if the validation error failed to improve.

Fig. 3 illustrates the process of converting network traffic data from a PCAP file into an image format for analysis. It starts with a PCAP file, which contains packets of network data. Each packet includes a header with metadata and a payload with the actual data. These packets are extracted and

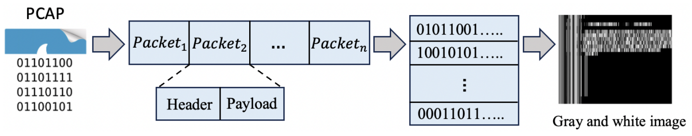


Fig. 3: Converting PCAP data to image.

converted into binary sequences of 0s and 1s, representing the entire content of each packet. Next, these binary sequences are organized into a grid to create an image. The patterns formed by the sequences of 0s and 1s generate a visual representation of the network traffic, which can be analyzed for patterns, anomalies, or other insights.

Log messages were tokenized using the DistilBERT tokenizer provided by the transformers library. This tokenizer is specifically designed to convert raw text into a sequence of tokens that can be effectively processed by the DistilBERT model, ensuring that the semantic meaning of the log messages is accurately captured. The tokenization process involved splitting the log messages into smaller units, such as words and subwords, and converting them into corresponding numerical representations that align with the pretrained vocabulary of DistilBERT. On the other hand, PCAP files, containing raw network packet data, were preprocessed to extract individual packets. The raw byte data was transformed into a format suitable for the packet-based transformer.

A. Dataset and Customized Rule Sets

We evaluate TransIDS on CICIDS 2018 dataset [31]. This dataset captures a wide array of network traffic over a five-day period, including a diverse set of malicious behaviors, with key attack vectors such as BruteForce, DoS, DDoS, and web-based attacks. We employed Snort 2.9, configured with both the Community rule set and the Emerging Threat rule set to ensure comprehensive detection coverage across a wide range of known threats, including both standard and emerging attack vectors. Additionally, we implemented customized rules to minimize false positive rates as much as possible. Some of the customized rules are as follows:

- **DoS-Slowloris:**
```
alert tcp any any -> any any
(msg:"Possible Slowloris
attack detected"; flags:S;
detection_filter:track by_src, count
5, seconds 60; sid:1000010; rev:2;)
```
- **BruteForce:**
```
alert tcp any any -> any HTTP_PORTS
(msg:" [+]Login Attempt
Detected On Web Login Page !";
content:"login.php"; sid:1000003;
rev:1;)
```
- **DDoS-LOIC-HTTP:**
```
alert tcp any any -> any 80 (msg:"LOIC
```

TABLE I: Performance metrics of different methods

Method	Individual Classes Accuracy				Overall		
	DDoS	**BruteForce**	**DoS**	**Bot**	**Accuracy**	**Recall**	**F1-score**
DistilBERT	0.7484	0.8119	0.7555	0.6975	0.7534	0.5073	0.6731
CNN-Packet	0.8662	0.9000	0.8950	0.8307	0.8784	0.8458	0.8745
Packet-based Transformer	0.9047	0.9250	0.9175	0.8537	0.9122	0.8776	0.8986
TransIDS	**0.9365**	**0.9700**	**0.9642**	**0.8975**	**0.9424**	**0.8948**	**0.9389**

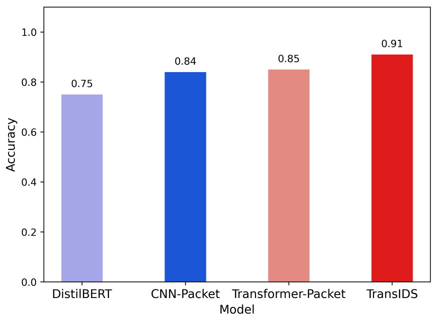


Fig. 4: Multi-class classification.

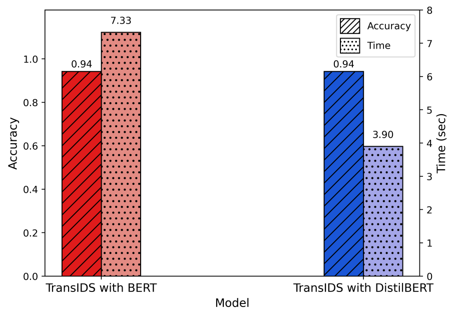


Fig. 5: BERT vs DistilBERT TransIDS.

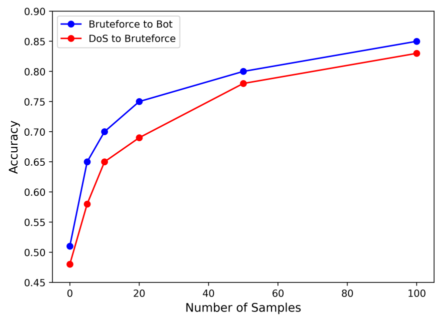


Fig. 6: Few-shot learning.

```
HTTP DDoS Attack Detected";
flow:to_server, established;
content:"User-Agent|3a| LOIC";
http_header; classtype:attempted-dos;
sid:100001; rev:1;)
```

- **DDoS-LOIC-UDP:**

```
alert udp any any -> any
any (msg:"LOIC UDP DDoS Attack
Detected"; dsize:<=32; threshold:type
both, track by_src, count 20,
seconds 2; classtype:attempted-dos;
sid:100002; rev:1;)
```

Snort was meticulously configured to generate a complete version of the log data, capturing detailed information about each detected event. Snort was configured to provide detailed log data corresponding to the detected attacks, including timestamps, attack types, and relevant network information.

B. Evaluation of TransIDS vs. Baselines

Table I presents a comprehensive comparison between our proposed methodology and the baseline models: 1) DistillBERT (processes only log messages without considering the packet data), 2) CNN-based model, and 3) Packet-based Transformer(operates exclusively on packet data without utilizing log messages). Results show that DistillBERT performs relatively poorly across all classes, with the lowest accuracy, recall, and F1-score. CNN-Packet shows moderate performance, particularly excelling in BruteForce and DoS attacks but struggling with BruteForce and Bot attacks. The Packet-based Transformer outperforms both, but the highest performance is achieved by TransIDS, which integrates multiple data types. TransIDS achieves superior effectiveness in handling various attack types by leveraging its multi-modal approach.

C. Multi-class Attack Classification

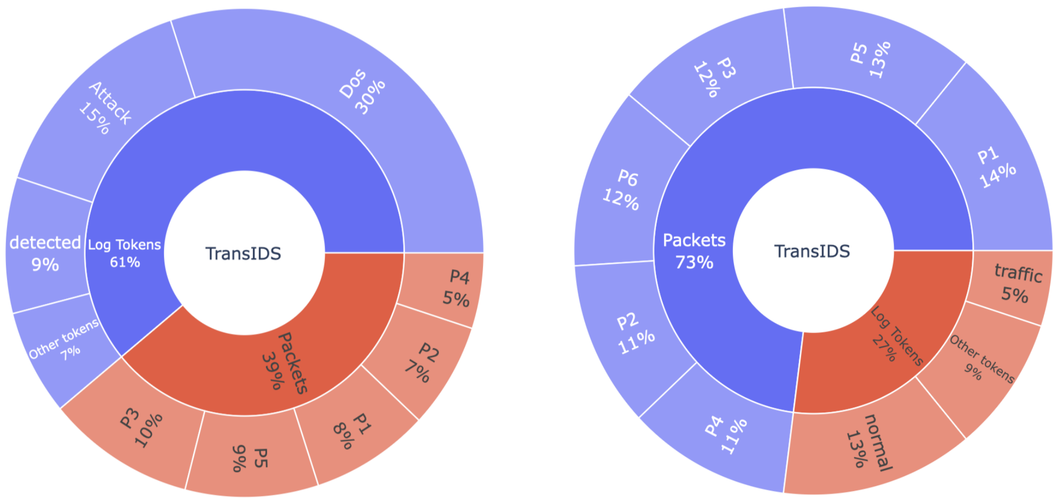
We aimed to develop a deep learning-based IDS capable of not only detecting attacks but also classifying the specific type of attack. To evaluate the effectiveness of TransIDS, we compared it against several baseline models: DistilBERT, CNN-Packet, and Transformer-Packet. The accuracy results for each model are presented in Fig. 4. Results show that TransIDS model significantly outperforms the baselines, achieving the highest accuracy of 0.91. This demonstrates the superiority of our methodology in handling the multi-class classification task, where distinguishing between various types of attacks is crucial for effective threat mitigation.

D. Efficiency of BERT vs. DistilBERT

This section presents a comparison of the performance of TransIDS using BERT and TransIDS using DistilBERT on 10,000 testing flows with a batch size of 512, focusing on accuracy and processing time, highlighting the trade-off between the two models. The results in Fig. 5 indicate that both versions of TransIDS achieve an identical accuracy of 0.94. However, DistilBERT significantly outperforms BERT in terms of processing time, requiring only 3.90 seconds compared to BERT's 7.33 seconds, making it a more efficient choice in time-sensitive scenarios. This demonstrates that by incorporating DistilBERT, we can maintain the same level of accuracy while significantly reducing inference time.

E. Model Interpretability

We employed an attention mechanism in our model to assess the relative importance of each modality (PCAP and the log sequence). We assess the importance of individual segments within each modality for the classification task. In this section, we focus on specific examples to analyze the decisions made by TransIDS and to examine the importance assigned to each



(a) Flow 1 (b) Flow 2

Fig. 7: Model Explainability for Flow 1 and Flow 2 related to DoS Attacks. (a) Flow 1 consists of 5 packets. It is detected by the log-based (signature-based IDS) as "Possible SYN DoS, Possible DoS attack detected". (b)Flow 2 contains 6 packets. It was not detected by the log-based system and identified as "It is a normal traffic".

segment of the log sequence and packet data. Fig. 7 presents the results. The inner circle in this figure illustrates the model's attention distribution between log tokens and packets, while the outer circle highlights the specific importance of individual log tokens and packets (P1, P2, etc.) in the decision-making process. We analyzed two flows, both of which represent DoS attacks. Flow 1 was successfully detected by our signature-based IDS. Flow 2 resulted in a false negative, meaning it went undetected by the traditional signature-based system. However, our TransIDS model was able to identify both flows as attacks.

As depicted in Fig. 7, for Flow 1, our model predominantly focuses on the log message, particularly on specific tokens such as "DoS" and "Attack," which are critical for identifying the nature of the threat. Additionally, the model assigns varying levels of importance to different packets within the flow, demonstrating its ability to weigh multiple factors in its decision-making process. In the case of Flow 2, which was a false negative for the signature-based IDS, our model adapts by primarily focusing on the packet data rather than the log tokens. This shift in focus highlights the model's capacity to learn from instances where traditional methods fail, utilizing packet-level information to correctly classify the flow as an attack. The attention distribution in this scenario underscores the importance of integrating both log and packet data and interpretability of our model.

F. Zero-day Attacks

In this section, we evaluate the knowledge transferability of our model in detecting zero-day attacks by comparing its performance across various attack types. Fig. 8 visualizes this comparison, illustrating that the model performs exceptionally well when tested on the same attack type it was trained on, as shown by the high values in diagonal cells, often near or equal to 1.00. This indicates a strong performance in detecting familiar attack types. However, when the model is tested on different attack types from those it was trained on, its generalization ability varies significantly. For instance, a model trained on BruteForce attacks achieves moderate accuracy in detecting DDoS attacks (0.64), but its performance drops when tested on more distinct attack types, such as DDoS evaluated on Bot (0.54), highlighting weaker generalization. We also observe high accuracy when transferring between DoS and DDoS attacks, likely due to their similarities, resulting in less distributional shift.

Fig. 8 also includes rows labeled "All" and "Leave-One-Out", representing the model's performance when trained on a combination of all attack types or using a leave-one-out strategy, where the model is trained on all tasks except the one being tested. These strategies generally lead to improved performance across different attack types, suggesting they enhance the model's ability to detect a broader range of zero-day attacks. Thus far, we have primarily focused on zero-shot learning. Next, we examined two scenarios with the lowest accuracy in Fig. 8 which are "BruteForce to Bot" and "DoS to BruteForce". We applied few-shot learning by further training the pretrained model on a limited number of samples from the target task, as shown in Fig. 6. The results demonstrate that even with a minimal number of samples, high accuracy can be achieved in both scenarios.

V. Conclusion

Efficient log anomaly detection is vital for effective service management and maintenance. However, existing methods often focus on a single type of anomaly, and fully utilizing log messages remains challenging. In this paper, we introduce TransIDS, a hybrid attention-based transformer framework for

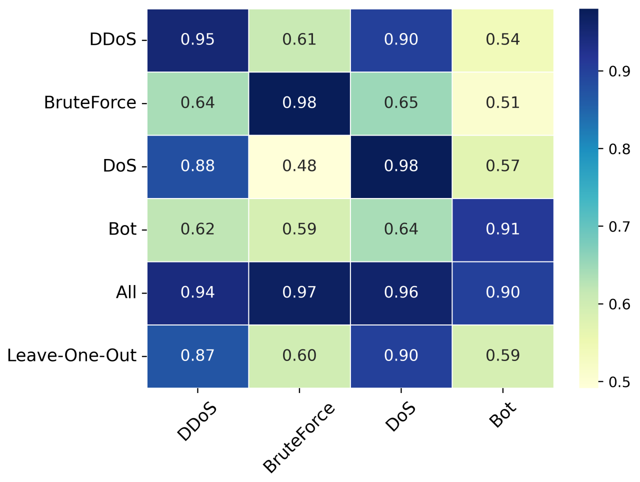


Fig. 8: Results on Zero-day Attacks

log anomaly detection, which incorporates a preprocessing stage to handle two different input types. Our extensive experiments show that TransIDS is stable across various hyperparameters and achieves excellent performance on diverse datasets and attack types. Additionally, using the Transformer enhances anomaly detection speed without sacrificing accuracy, meeting the demands of timely diagnosis in log streams. Future work could involve adding more stages to enhance anomaly extraction and evaluating the model on a wider range of datasets with varying traffic sizes and novel attacks.

References

[1] N. Niknami and J. Wu, "Advanced ml/dl-based intrusion detection systems for software-defined networks," in *Network Security Empowered by Artificial Intelligence*. Springer, 2024, pp. 121–146.

[2] N. Niknami and J. Wu, "Deepidps: An adaptive drl-based intrusion detection and prevention system for sdn," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2024.

[3] N. Niknami, V. Mahzoon, and J. Wu, "Meta-ids: A multi-stage deep intrusion detection system with optimal cpu usage," in *International Conference on Meta Computing (ICMC)*. IEEE, 2024, pp. 74–83.

[4] N. Niknami, V. Mahzoon, and J. Wu, "Crossalert: Enhancing multi-stage attack detection through semantic embedding of alerts across targeted domains," in *Conference on Communications and Network Security (CNS)*. IEEE, 2024, pp. 1–9.

[5] M. Cinque, R. Della Corte, and A. Pecchia, "Microservices monitoring with event logs and black box execution tracing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 294–307, 2019.

[6] S. Huang, Y. Liu, C. Fung, H. Wang, H. Yang, and Z. Luan, "Improving log-based anomaly detection by pre-training hierarchical transformers," *IEEE Transactions on Computers*, vol. 72, no. 9, pp. 2656–2667, 2023.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[8] R. F. Bikmukhamedov and A. F. Naplov, "Generative transformer framework for network traffic generation and classification," *T-Comm-*, vol. 14, no. 11, pp. 64–71, 2020.

[9] H. Y. He, Z. G. Yang, and X. N. Chen, "Pert: Payload encoding representation from transformer for encrypted traffic classification," in *ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*. IEEE, 2020, pp. 1–8.

[10] A. Barua, M. U. Ahmed, and S. Begum, "A systematic literature review on multimodal machine learning: Applications, challenges, gaps and future directions," *IEEE Access*, vol. 11, pp. 14 804–14 831, 2023.

[11] P. Gao, X. Xiao, Z. Li, F. Xu, S. R. Kulkarni, and P. Mittal, "Aiql: Enabling efficient attack investigation from system monitoring data," in *USENIX Annual Technical Conference (USENIX ATC)*, 2018, pp. 113–126.

[12] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *IJCAI*, vol. 19, no. 7, 2019, pp. 4739–4745.

[13] B. Xia, Y. Bai, J. Yin, Y. Li, and J. Xu, "Loggan: a log-level generative adversarial network for anomaly detection using permutation event modeling," *Information Systems Frontiers*, vol. 23, pp. 285–298, 2021.

[14] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2672–2681.

[15] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 807–817.

[16] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1285–1298.

[17] J. Liu, M. Simsek, M. Nogueira, and B. Kantarci, "Multidomain transformer-based deep learning for early detection of network intrusion," *arXiv preprint arXiv:2309.01070*, 2023.

[18] N. Xing, S. Zhao, Y. Wang, K. Ning, and X. Liu, "A dynamic intrusion detection system capable of detecting unknown attacks," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 7, 2023.

[19] S. Luo, Z. Zhao, Q. Hu, and Y. Liu, "A hierarchical cnn-transformer model for network intrusion detection," in *Proceedings of the 2nd International Conference on Applied Mathematics, Modelling, and Intelligent Computing (CAMMIC)*, vol. 12259, 2022, pp. 853–860.

[20] Y. Li, X. Yuan, and W. Li, "An extreme semi-supervised framework based on transformer for network intrusion detection," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 4204–4208.

[21] H. Wang and W. Li, "Ddostc: A transformer-based network attack detection hybrid mechanism in sdn," *Sensors*, vol. 21, no. 15, p. 5047, 2021.

[22] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64 375–64 387, 2022.

[23] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[24] S. R. Wibisono and A. I. Kistijantoro, "Log anomaly detection using adaptive universal transformer," in *Proceedings of the IEEE International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 2019, pp. 1–6.

[25] H. Guo, X. Lin, J. Yang, Y. Zhuang, J. Bai, T. Zheng, B. Zhang, and Z. Li, "Translog: A unified transformer-based framework for log anomaly detection," *arXiv preprint arXiv:2201.00016*, 2021.

[26] T. Xiao, Z. Quan, Z.-J. Wang, Y. Le, Y. Du, X. Liao, K. Li, and K. Li, "Loader: A log anomaly detector based on transformer," *IEEE Transactions on Services Computing*, 2023.

[27] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceeding of the IEEE Conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[31] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.