Towards Real-time Network Intrusion Detection with Image-based Sequential Packets Representation

Jalal Ghadermazi [1], Ankit Shah [2], and Nathaniel Bastian [1]

[1]Affiliation not available
[2]University of South Florida

October 31, 2023

Abstract

This study proposes a novel artificial intelligence-enabled methodological framework for packet-based network intrusion detection system that effectively analyzes header and payload data and considers temporal connections among packets. The AI framework transforms sequential packets into a two-dimensional image, which is then passed through a convolutional neural network-based intrusion detector model. Experimental results using publicly available data sets demonstrate that the methodology can detect network attacks earlier than flow-based approaches. It also exhibits high transferability and shows promising resilience against adversarial examples.

Towards Real-time Network Intrusion Detection with Image-based Sequential Packets Representation

Jalal Ghadermazi, Ankit Shah*, *Member, IEEE*, Nathaniel D. Bastian, *Member, IEEE*

Abstract—Machine learning (ML) and deep learning (DL) advancements have greatly enhanced anomaly detection of network intrusion detection systems (NIDS) by empowering them to analyze big data and extract patterns. ML/DL-based NIDS are trained using either flow-based or packet-based features. Flow-based NIDS are suitable for offline traffic analysis, while packet-based NIDS can analyze traffic and detect attacks in real-time. However, current packet-based approaches rely on assumptions that generate bias in the models, resulting in an increase in false negatives and positives. Additionally, most literature-proposed packet-based NIDS capture only payload data, leaving out significant information from packet headers. To address these limitations, we propose a novel artificial intelligence-enabled methodological framework for packet-based NIDS that effectively analyzes header and payload data and considers temporal connections among packets. Our framework transforms sequential packets into a two-dimensional image, which is then passed through a convolutional neural network-based intrusion detector model. Our framework excels in detecting network attacks earlier than flow-based approaches, achieving detection rates of 97.7% to 99% across different attack types using publicly available big data sets. It also exhibits high transferability, with an average attack detection rate of 95% on a new target data set, and displays promising resilience against adversarial examples.

Index Terms—Network intrusion detection system, packet-based NIDS, early attack detection, sequential packets image representation

1 Introduction

Intrusion detection systems (IDS) are designed to monitor and identify attacks on organizations' computer and network systems. They can be classified into host-based IDS (HIDS) and network-based IDS (NIDS). NIDS are a popular option for detecting attacks in large organizations, since they analyze the network traffic of critical nodes to identify attack behavior, as opposed to monitoring a single node in HIDS. The detection strategies used in NIDS include signature-based and anomaly-based methods. While signature-based methods rely on creating domain-specific rules, anomaly-based methods employ machine learning (ML) and deep learning (DL) algorithms and train on big data to identify malicious behavior. ML/DL-enabled NIDS are mainly trained using either of the two types of features extracted from the network traffic data, flow-based or packet-based.

Flow-based features aggregate information from the packet headers in network communications, while packet-based features are extracted directly from the packet data. There are many limitations to the flow-based NIDS. (i) They analyze traffic once the flow between the sender and receiver is completed to identify any malicious activity, making them suitable for offline network traffic analysis [1]. (ii) They mainly extract features from lower levels of the transmission control protocol (TCP)/ internet protocol (IP) model, making it challenging to detect higher-level attacks that target the application layer [2]. For example, a *distributed denial-of-service (DDoS)* attack such as *SYN* flooding targets network packet header data, whereas a *SQL injection* attack injects anomalous code into the SQL queries (i.e., at a packet payload level) [3]. (iii) They identify attacks based on extracted flow features, which do not capture the functional behavior of network traffic in the packets. (iv) With different ways of extracting flow-based features, including the use of CIC Flowmeter [4] and Zeek (Bro) [5], the feature set used to train the intrusion detection models also varies among different network environments, making it difficult to benchmark the trained model's performance in a new target environment (domain adaptability).

Packet-based NIDS, on the other hand, are more suitable for real-time detection as features are extracted directly from the packet data. However, there are challenges with the packet-based approach. (i) Categorizing packets as benign or malicious is non-trivial. Not all packets have a malicious intent in an attack. For e.g., packets such as TCP three-way handshakes represent normal network characteristics in both benign as well as malicious traffic. (ii) Most packet-based NIDS do not consider the sequential functioning of packets in a flow and instead treat them as independent packets. As a result, the temporal correlations among the packets belonging to the flows are not captured [6], which may result in an incorrect classification by the NIDS (iii) They do not consider the direction of packets due to independence assumptions. However, the direction of a packet in a flow (forward or backward) can provide significant information in identifying attacks. For instance, network attacks like *Distributed Denial-of-Service (DDoS)* and *Port*

- *J. Ghadermazi and A. Shah are with the University of South Florida, Tampa, FL 33620, USA. Email: {jghadermazi,ankitshah}@usf.edu.*
- *N. Bastian is with the United States Military Academy, West Point, NY, 10996, USA. Email: nathaniel.bastian@westpoint.edu*
- ** corresponding author*

Scan often exhibit substantial differences in forward and backward packet patterns compared to normal traffic [7]. Our study addresses these limitations in flow-based and packet-based approaches for a timely detection of network attacks. We propose a novel methodology that combines both these approaches to preserve the temporal-spatial association between packets and their features for prompt detection of different attacks on packet header and payload data. Our method extracts features from high-level and low-level packet information and utilizes them sequentially.

The contributions of this study are as follows. The primary contribution is the development of a novel methodological framework that leverages AI techniques for (near-)real-time detection of network attacks. This AI-enabled framework overcomes the limitations of traditional packet-based NIDS by considering both header and payload data and analyzing temporal connections among packets. Another novel aspect of our method is the unique representation of the network traffic data. The sequential packets in a communication flow are transformed into a two-dimensional image, enabling the application of convolutional neural network (CNN) for intrusion detection. This representation allows the development of an optimized CNN-based network intrusion detection model that captures the underlying patterns and features associated with the network attacks. Other relevant contributions to the cybersecurity research community include the insights from the conducted experiments and their subsequent analyses. The study found that malicious intent can be detected early in a network communication during an attack. The transmission of the fourth to the ninth packet in a two-way communication was sufficient to detect malicious activity with high accuracy. This early detection capability is a paradigm shift in reducing response time to network attacks compared to flow-based approaches that typically require analysis of a large number of packets before making a detection. Our methodology has shown promising results of being deployable in diverse environments without requiring complete retraining, enhancing flexibility and efficiency for cybersecurity teams. Our sequential packets image-based network intrusion detection system (SPIN-IDS) framework also demonstrated robustness against adversarial examples, accurately detecting network attacks even with carefully crafted packet perturbations, unlike other ML/DL-based NIDS with high false negative rates.

The remainder of this paper is structured as follows. Section 2 reviews the literature on flow-based and packet-based NIDS that utilize ML/DL algorithms and identifies research gaps. Section 3 offers a detailed description of the proposed methodology, covering packet-based feature extraction, image representation of ongoing traffic, and development of the network intrusion detection model. In Section 4, the numerical experiments are discussed, while the results and analysis of these experiments conducted using our framework are presented in Section 5. Finally, Section 6 provides the conclusions of this study and potential directions for future research.

2 Literature Review

Over the past decade, significant research has been conducted on integrating ML/DL algorithms into the development of anomaly-based NIDS. ML/DL-enabled NIDS can be developed using features extracted from network flows as well as from the information obtained directly from the packets. Below we provide a summary of literature pertaining to both these approaches.

2.1 Flow-based NIDS

The predominant approach used in building NIDS is through a flow-based feature extraction process, which entails analyzing a network communication (or a flow) and aggregating information from its packets. Packet header data is typically used to obtain flow-based features [1], using publicly available tools like CIC FlowMeter [4]. ML/DL models are then trained using these features to identify anomalies in network traffic.

Deep neural network (DNN), a fundamental DL structure with multiple layers, including input, hidden, and output layers, is used to model complex nonlinear functions [8], such as the ones that map network traffic data to benign or malicious categorical labels. Authors in [9] proposed a NIDS using DNN with four hidden layers. Their experiments showed a superior performance of their DL model against other ML classifiers, while noting a lower detection accuracy for the *user to root* (U2R) attack class. The performance was benchmarked using older intrusion detection evaluation data sets such as KDD Cup'99 [10] and NSL-KDD [11]. Recurrent neural networks (RNNs), as an extension of traditional feed-forward neural networks, have also been proposed by cybersecurity researchers for the development of NIDS [12]–[14]. Yin et al. [12] presented an RNN-based NIDS for binary and multiclass classification of samples in the NSL-KDD data set. Their NIDS was shown to outperform the ML classifiers. Higher computational processing was required to train their model and the experiment results using their trained model indicated a low detection rate for some attack types such as *remote to local* (R2L) and U2R attack classes. Xu et al. [13] proposed an RNN-based NIDS using gated recurrent units (GRU). Their study used older data sets for validation and their model had lower accuracy in detecting the minority attack class samples. Naseer et al. [14] compared different DL and ML algorithms for detecting intrusions and found that a coupled model of long short-term memory (LSTM) and CNN achieved better accuracy compared to other approaches. The validation in this study was performed using the older and almost obsolete data set, NSL-KDD.

Autoencoder (AE) is a DL technique that uses an unsupervised learning approach, in which the input data samples do not have any labels associated with them [15]. Shone et al. [16] proposed a NIDS based on deep AE and random forest (RF) algorithm, which showed improved efficiency compared to the deep belief network (DBN) approach used in [17] . Yan et al. [18] proposed a NIDS using stacked sparse autoencoder (SSAE) and support vector machine (SVM) to achieve superior performance. Yang et al. [19] proposed a NIDS model based on supervised variational autoencoder (VAE) with regularization and DNN (SAVAER-DNN) that

was effective in detecting low frequency and new attacks. However, all of these AE-based models were evaluated on older data sets, KDD Cup '99 and NSL-KDD, and showed lower accuracy in detecting samples belonging to minority attack classes.

CNN is commonly used for image classification and object detection tasks due to its ability to capture spatial information through convolutional filters. Flow-based features do not contain spatial relations. Hence, CNN is often used in hybrid systems with RNN, in which the CNN model acts as a feature extractor and the output is classified with RNN [20]–[22]. Yu et al. [23] proposed a NIDS model based on few-shot learning, which uses DNN and CNN as embedding functions for feature extraction and dimension reduction. The aforementioned models were evaluated using KDD Cup'99, NSL-KDD, and UNSW-NB15 data sets, and their detection accuracies were found to be lower for the minority attack classes.

2.2 Packet-based NIDS

An alternative method of developing ML/DL-enabled NIDS is by extracting features directly from each packet in a flow and then training the model on these packet-based features. Recent literature studies examining the use of packet-based features for NIDS are as follows. An RNN method with an attention mechanism (ATPAD), proposed in [24], utilizes word embedding and RNN to extract features that capture the correlation between detection results and potential bytes of the payload. The proposed method uses binary classification and is trained using the CICIDS-2017 data [25]. Another study, [26], also uses the same data set to construct a block sequence using the packet payload data. This approach captures short-term and long-term dependency relationships among the malicious bytes in the payload data. A tool named payload-byte is proposed in [27] to extract the packet payload bytes from publicly available data sets containing raw packet capture (pcap) files. Using the extracted packets, DNN-based models are then trained to detect different attack types. With the results from their experimental study, the authors concluded that packet-based NIDS are as effective as flow-based NIDS. However, these studies [24], [26], [27] are limited to only capturing payload features. However, the maliciousness of some attacks lies in the packet header data [2]. Additionally, these studies only consider packets with payload data and do not develop a structure for sequentially capturing packets belonging to the same flow, making it difficult to identify malicious attacks in real-time.

The authors in [28] proposed a unified packet representation that employs raw packet information to fingerprint host operating systems and devices. To evaluate its efficacy, they tested the proposed method on ten distinct data sets. Although the proposed approach was found to be suitable for fingerprint host operating systems, it is not applicable to network traffic data classification. This approach incorporates all bytes from the raw packet file, including headers that contain information about IP addresses and ports. Such an all-inclusive approach to packet analysis may result in the model being trained to recognize specific IP addresses that generate malicious traffic or frequently attacked ports. Zhang et al. [29] proposed a method for creating grey-scale images of packets without performing feature extraction. In their study, each packet was assumed to have 1518 bytes of hexadecimal payload data padded with 0s. The p-zigzag encoding scheme was used to create a grey-scale image, which was then transformed using inverse discrete cosine transform to create a pattern texture image for classification. However, the spatial-temporal relationship of packets belonging to the same flow was ignored, as the images only represented individual packets. In addition, similar to the approach in [28], their method is all-inclusive and thus training can be biased towards certain header information, such as IP addresses and ports. Yu et al. [30] proposed a CNN-based IDS named PBCNN. Their method involves capturing the raw packet data and creating a single image that comprises all packets belonging to a flow/session. This approach demonstrates high efficiency when analyzing modern data sets, such as CICIDS-2017 and CICIDS-2018. However, their method is unsuitable for real-time attack detection since it captures a fixed number of packets within a session/flow (20 packets) before providing a prediction (similar to flow-based methods). By using a fixed number of packets to generate images from network flows will result in missing out on network flows which contain only a few number of packets, such as DDoS attacks. Also, they create greyscale images from the packets in a flow which ignores how forward and backward packets are transmitted in the network. As a result, the patterns in forward packets sent by the attacker is not captured. Additionally, keeping network-specific features from the packet data, such as source ports, destination ports, and protocol will add bias towards this information in the model.

2.3 Research gaps in literature

In summary, packet-based NIDS are well suited for detecting network attacks in (near-)real-time as they reflect the network's current state and can detect anomalies as they emerge. However, the current packet-based approaches for creating ML/DL-based NIDS make two strong assumptions. First, all packets that are a part of an attack (malicious flow) are categorized as malicious during the training of these models. Clearly, not all packets are malicious in the entire network communication, in which an attack is orchestrated. Second, each packet is evaluated independently for its maliciousness without considering its dependency on other packets in the respective flow. In a real-world implementation, this approach will result in false positives as the trained model will classify many harmless packets as malicious, resulting in a high volume of alerts for security analysts to investigate in a resource-constrained environment. At the same time, such an approach will also miss out on detecting malicious attacks, resulting in false negatives. In addition to the aforementioned impacts of these assumptions, tracing back a packet to the actual flow for investigation will add more complexity to the intrusion detection and mitigation process, thereby requiring more processing time. It is also to be noted that most of the packet-based NIDS proposed in literature only capture payload data, thereby leaving out significant information from the packet headers. Although some studies [28], [29] have explored the use of both header

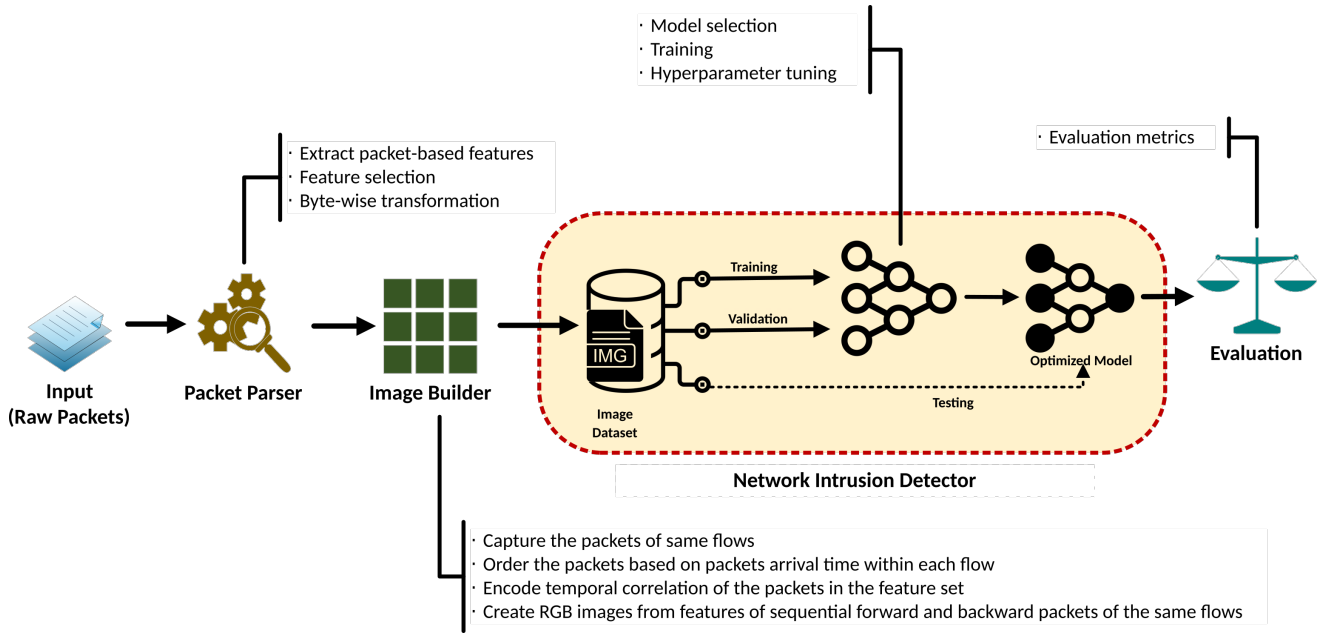


Fig. 1: Framework for the Sequential Packets Image-based Network Intrusion Detection System (SPIN-IDS)

and payload data, the incorporation of all header data can introduce bias into the model training process and impact the model's accuracy when deployed in a different network environment.

3 AI-enabled Methodological Framework

The objective of this research study is to address the aforementioned gaps by developing an accurate and quick intrusion detection mechanism that can detect a network attack with a high confidence by analyzing a minimum number of packets in an ongoing flow. Our proposed AI-enabled framework, shown in Figure 1, aims to achieve this objective by taking into account both header and payload data of the sequential packets in an evolving flow and transforming them into a two dimensional (2D) image representation. The sequential packets image-based network intrusion detection system (SPIN-IDS) framework consists of three components: (i) a packet parser, (ii) an image builder, and (iii) a network intrusion detector. The first component extracts packet-based features from the raw network traffic data. The second component preprocesses the extracted feature data to generate 2D images. Finally, the third component determines if the network traffic is malicious or not. Next, we describe each of these components in detail.

3.1 *Packet parser*

To understand packet-based feature extraction from a raw packet file, it is essential to know the structure in which packets are stored. The TCP/IP model is the standard model used in network communication to regulate the procedure of information sharing across the internet. It consists of four layers: the network access layer (also known as the host-to-network layer), the internet layer, the transport layer, and the application layer. The transmission control protocol is responsible for breaking the message into TCP segment packets and reassembling them at the destination. Figure 2 displays the different TCP/IP layers, along with the number of information bytes at each layer. Network traffic data is stored in the libpcap (pcap) format, which is considered to be the *de facto* standard for network packet capture and widely used in packet sniffers and analysis tools like Wireshark [31].

The packet parser component takes the network traffic (in real-time or *pcap* files) as input data. Each packet transmitted through the TCP contains up to 1594 information bytes. Information related to the environment and protocols can bias the model and make it less applicable to different environments. Hence, to remove this bias, the Ethernet (ETH) header information (14 bytes), the IP version (one byte), the differentiated services field (1 byte), the protocol (one byte), and the source and destination IP addresses information (four bytes each) from the IP header are eliminated. The source and destination ports information bytes (two bytes each) from the TCP header of each packet data are also removed. Additionally, the IP options and TCP options, which can cause misalignment between two packets of the same flow and introduce noise in the model, are removed. Misalignment occurs when the bytes in two feature representations of packets with and without options are not aligned, leading to a decrease in model performance and interpretability [28]. To encode temporal relationship between packets in the same flow, a delta time feature is introduced that calculates the time difference between two packets of the same flow using the epoch time of each packet. After removing these information bytes (a total of 109 bytes, shown in red in Figure 2) and encoding the temporal correlation feature into the feature space, the resulting packet-based feature representation, V_{packet}, contains a maximum of 1486 bytes of information. Each byte represents a feature in the packet-based feature representation. Next, a byte-wise transformation is applied to the packet-based features, converting the hexadecimal byte values to the respective decimal values. The decimal

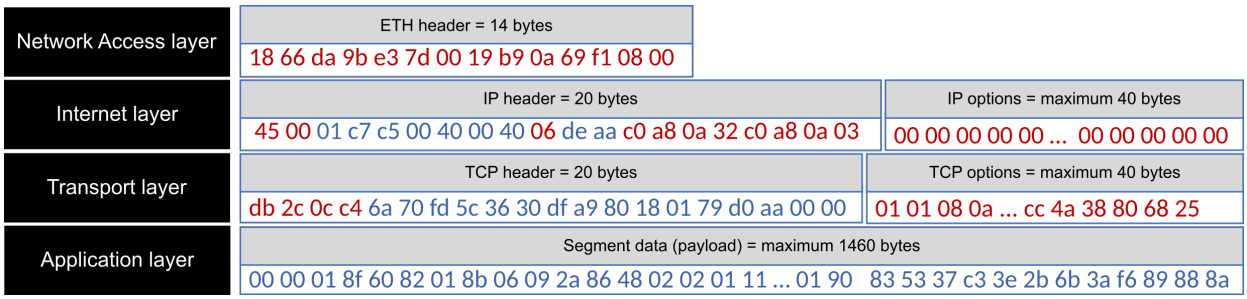


Fig. 2: TCP/IP model layers with information byte details

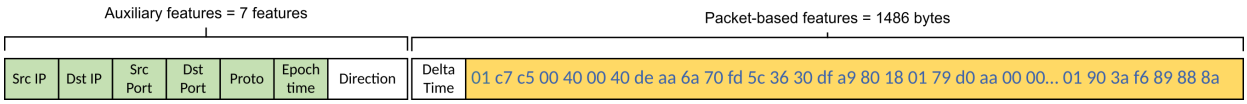


Fig. 3: Auxiliary and packet-based features representation

value ranges from 0 (for 00 byte) to 255 (for ff byte), which is suitable for an image representation of the packet data. Since the number of bytes varies depending on the packet type, zero-padding is applied to the feature space to maintain a standard structure, resulting in a fixed number of features (N) for each packet.

There are attacks aimed at keeping the destination (server) busy and consume its resources by sending multiple null packets with no particular malicious data (e.g., *SYN flooding*) [32]. It becomes important to differentiate between forward (attacker to server) and backward (server to attacker) packets. Hence, the direction of each packet is encoded as an auxiliary feature in the packet representation scheme. In total, seven auxiliary features are captured from each packet, which are used in the image creation process in the image builder component. These features include source IP address (SrcIP), destination IP address (DstIP), source port (SrcPort), destination port (DstPort), protocol (Proto), epoch time, and direction of the packet. Figure 3 shows the packet-based feature scheme. The output of the packet parser is a packet-based data which contains seven auxiliary features and a total of 1486 packet-based features. The direction and delta time features are initially set to null values for each packet. The direction and delta time features are computed in the image builder component. Algorithm 1 presents the packet parser process.

Algorithm 1 Algorithm for the packet parser process

Input: Real-time network data/captured network data (*pcap*) files.
Output: Transformed packet data
/* Set the packet-based features length to N for all packets. */
for *each packet* **do**
  Obtain the packet header and payload data
  Do feature selection from the header and payload data
  Do byte-wise transformation to convert hexadecimal bytes to decimal values (0-255)
  Assign transformed data to packet feature vector V_{packet}
  Add the delta time feature to V_{packet} /* Set to $Null$ */
  Do zero-padding if length(V_{packet}) $\neq N$
  Capture and add the auxiliary features to V_{packet}
end

3.2 *Image builder*

To capture the temporal-spatial relationships among the packets within a flow, we develop a $2D$ ($P \times Q$) image builder that uses sequential packets to generate snapshots of the evolving flow as new packets arrive. The transformed packet-based data obtained from the packet parser component serves as an input to the image builder component. With the help of the auxiliary features, the packets belonging to the same flow are extracted, and delta time and direction information is computed. Delta time is encoded as a feature, and the same-flow packets are stacked across the P dimension of the image to capture the temporal relationships. The spatial and semantic correlations in the packets are preserved by constructing a static representation of packet features across the Q dimension of the image. The P dimension can be determined by the security team of an organization and can be derived using statistical measures such as the mean or the median number of packets found in each flow in their respective network environment. The Q dimension is the length of the packet-based feature vector (1486), as defined in the packet parser component. It is to be noted that the construction of an image does not require all P number of packets from an ongoing flow.

The packet-based features are stored using an image with three channels, red (R), green (G), and blue (B). We propose using an RGB image because the current state of the forward and backward packets is essential in identifying the underlying patterns in network attacks. Using a grey-scale mode would damage this pattern in the data. Instead, the RGB channels provide information about how the forward and backward packets are played out in the flow. The forward packet information is stored in the R channel of the images, and the backward packet information is stored in the G channel of the images. The third channel, B, is zero-padded to maintain the same structure for all images generated by the image builder component. This process is conducted sequentially, and every time a packet arrives, the direction of the packet is checked, and the feature vector values are assigned to the respective channel, leaving the other two channels blank. For training the network intrusion detector (the third component in the framework), historical

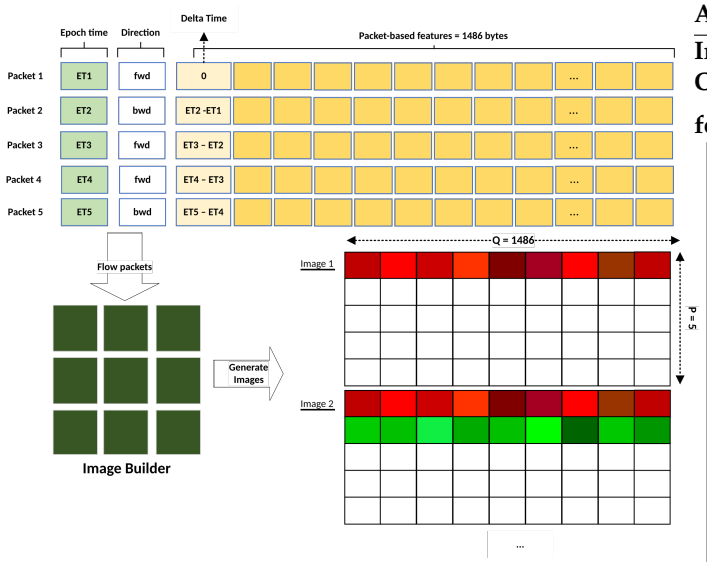


Fig. 4: Schematic of the image builder process showing generation of the first two images in a flow

data sets available in the organizations can be used to create the training data. Alternatively, the training data can be created from the modern and preferred publicly available network intrusion data sets, such as those provided by the Canadian Institute of Cybersecurity (CIC) (CICIDS-2017 [25] and CICIDS-2018 [4]). Each image created using the image builder component for the training phase is assigned the label of the network flow found in the historical/publicly available data. For instance, an image created from the packet data in CICIDS-2017 data set will belong to one of the 15 different labels, including one benign and 14 attack types found in that data set. Figure 4 illustrates the image builder process for an instance of an evolving flow with five packets. The first packet is transmitted in the forward direction. Consequently, the feature values of this packet are stored in the R channel of the image representation, while zero-padding the G and B channels for the first row. Conversely, the second packet is transmitted in the backward direction, and thus its feature values are assigned to the G channel of the image representation. In this case, we zero-pad the R and B channels for the second row of the image. This process continues until all the five packets are represented in this image. Algorithm 2 presents the pseudo-code for the image builder process.

Algorithm 2 Algorithm for the image builder process

Input: Packet-based data (from Algorithm 1), P.
Output: Image data

for *each packet in packet-based data* **do**
  Obtain the $flow$ packets as follows:
  Identify packets with same tuple $(SrcIP, DestIP, SrcPort, DestPort, Proto)$ and label them $forward$ packets
  Identify packets with same tuple $(DestIP, SrcIP, DestPort, SrcPort, Proto)$ and label them $backward$ packets
  Store the $flow$ packets based on $epoch\ time$
  for *each* packet *in* $flow$ **do**
    $(delta\ \ time)_{packet}$ = $(epoch\ \ time)_{packet}$ - $(epoch\ time)_{previous\ packet}$
    Transform $(delta\ time)_{packet}$ between $[0, 255]$
    Encode $(delta\ time)$ in the packet-based feature vector (V_{packet})
  end
  Set the image dimension to $(P \times 1486 \times 3)$
  zero-pad red (R), green (G), and blue (B) channels
  $p \leftarrow 0$ /* enumerates row of images for sequential process*/
  for *each packet in flow* **do**
    if $p \geq P$ **then**
      break
    else if *packet direction is forward* **then**
      Assign feature vector of packet to row p of R channel
    else
      /* the packet direction is backward*/
      Assign feature vector of packet to row p of G channel
    end
    Create the image of p packets in the $flow$
    $p++$
  end
  Remove $flow$ packets from packet-based data
end
return *Image data*

3.3 Network intrusion detector

The objective of the network intrusion detector is to determine whether an image, obtained from the image builder component, contains benign or malicious traffic. To achieve this objective, we pose the problem of intrusion detection as a binary image classification problem. We propose the use of a CNN architecture for the development of the detector. CNN models are well known for their superior performance in classifying image data as well as their fast inference time [33]. They have a distinct multilayer neural network architecture compared to the feedforward models. In particular, CNNs consist of an input layer, an output layer, and multiple hidden layers, typically comprising of convolutional, pooling (such as maxpooling), and fully

connected layers. The core operations in the CNN model include convolution and sampling processes [34]. During the convolution process, various filters are applied to the original data or feature map and a bias is added. The convolution process for an input image sample is based on the following equation, in which L represents the input image's length, K is the kernel size, Z denotes the amount of zero-padding added to both ends of the image dimension, and S is the stride of the kernel on a convolution layer.

$$L' = \frac{(L - K + 2Z)}{S} + 1 \tag{1}$$

Algorithm 3 Algorithm for training and validating the network intrusion detector

Input : Training image data, validation image data, hyperparameter set, CNN model architecture.
Output: Optimized model
while *Training* $\neq$ ***done*** **do**
  for *every hyperparameter selection* **do**
    For each epoch in epoch range
      Create image batches from the training and validation image data
      Train the CNN model with training image batches
      Evaluate the model performance with validation image data
      if *EarlyStopping is True* **break else continue**
      Record hyperparameter selection and model performance on validation image data
    end
  end
end
Choose the best hyperparameter selection for the model
return *Optimized model*

Although using multiple convolution layers can potentially lead to better learning of images with complex features, the number and performance of these layers are not always proportional. Hence, an ideal architecture must be selected during the training phase from a wide range of possibilities, such as from shallow (with only one convolutional layer) to deep (with multiple convolutional layers) architectures along with different padding and stride strategies. An appropriate loss function and activation function must also be selected that suits the problem type (binary classification), along with the optimal tuning of other hyperparameters. During the training phase, different images are created from each flow, with an incremental number of sequential packets observed in that communication. The objective, through experimentation, is to find image data with an appropriate number of packets that will help the detector learn patterns in the network communication to produce a higher accuracy in malicious traffic detection. The training and validation process of the network intrusion detector model is outlined in Algorithm 3.

3.3.1 Evaluation

The performance of the detector is evaluated using a range of metrics, commonly employed for intrusion detection models. These metrics are calculated using the following model prediction values on the testing data samples:

- True positives (TP): number of data instances correctly classified as an attack
- False negatives (FN): number of attacks incorrectly classified as benign traffic
- False positives (FP): number of benign samples incorrectly classified as an attack
- True negatives (TN): number of data instances correctly classified as benign traffic

Below are the various metrics used in the experiments to evaluate the model performance:

- **Accuracy:** It is the ratio of the total number of correctly classified samples of both classes and the total number of samples, as given below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- **Precision:** It is the ratio of the number of correctly classified attacks and the total number of samples predicted as attacks by the model, as given below:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

- **True positive rate (TPR) or Recall:** It is also known as the detection rate (DR) and is the ratio of correctly classified attack samples and the total number of attack samples, as given below:

$$TPR\,(Recall) = \frac{TP}{TP + FN} \tag{4}$$

- **F1 score:** It is the harmonic mean of precision and recall values for examining the accuracy of the model, as given below:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

- **True negative rate (TNR):** It is the ratio of the number of correctly classified benign samples and the total number of benign samples, as given below:

$$TNR = \frac{TN}{TN + FP} \tag{6}$$

- **False negative rate (FNR):** It is the ratio of the number of incorrectly classified attack samples and the total number of attack samples, as given below:

$$FNR = \frac{FN}{TP + FN} \tag{7}$$

- **False positive rate (FPR):** It is also known as the false alarm rate (FAR) and is the ratio of incorrectly classified benign samples and the total number of benign samples.

$$FPR\,(FAR) = \frac{FP}{FP + TN} \tag{8}$$

TABLE 1: CICIDS-2017 pcap file details

Day/Date	File Size	Activity
Monday/ July 3, 2017	11 GB	Benign
Tuesday/ July 4, 2017	11 GB	Brute Force and Benign
Wednesday/ July 5, 2017	13 GB	DoS and Benign
Thursday/ July 6, 2017	7.7 GB	Web Attack, Infiltration, and Benign
Friday/ July 7, 2017	8.2 GB	Botnet, Port Scan, DDoS, and Benign

4 Numerical Experiments

This section presents an overview of the numerical experiments conducted to assess our methodology. We, first, provide details of the network traffic data used in the experiments, followed by the process of creating image representation of the data. We describe how we split the data into training, validation, and testing data sets to develop the network intrusion detector. The experiments were conducted using a 12^{th} Generation Intel Core i9-12950HX processor (30 MB cache, 24 threads, 16 cores). In order to speed up the training process, NVIDIA RTX A5500 graphics card (16GB GDDR6 SDRAM) was utilized with the latest installation of CUDA, a universal parallel computing framework, and cuDNN, a deep neural network acceleration library.

4.1 *Data description*

We conducted numerical experiments using the raw pcap files from two well-known network intrusion detection data sets, namely CICIDS-2017 [25] and CICIDS-2018 [4]. These data sets comprise both benign and attack communications and offer a more practical representation of contemporary network traffic in comparison to older network intrusion data sets such as NSL-KDD and KDD-CUP [35]. The pcap files for CICIDS-2017 contain network traffic data for five consecutive days (Monday to Friday), each featuring distinct attack types and sizes, as shown in Table 1. We used Python-based dpkt [36] and Scapy [37] to parse the raw network traffic and create the packet-based feature data set, as described in Section 3. Table 2 presents the number of forward and backward packets that we extracted for each attack type. We validated the output of the packet parser component for each attack type using the corrected version of the public NetFlow CSV files for the CICIDS-2017 data [38].

To assess the transferability of the trained network intrusion detector model on distinct network traffic data (domain adaptability characteristic), we utilized CICIDS-2018 data. The CICIDS-2018 data features various attack types and sizes of pcap files, as shown in Table 3. To demonstrate the adaptability of our AI-enabled SPIN-IDS framework with the trained network intrusion detector, we extracted packets for the following attack types, *DoS*, *Web Attack*, *Infiltration*, and *Brute Force*, from these large pcap files. The number of forward and backward packets extracted for these attacks is presented in Table 4.

TABLE 2: Packet details per attack type in CICIDS-2017

Attack Type	Number of Packets	
	Forward	Backward
DoS GoldenEye	66795	39382
DoS Hulk	1246802	1000016
DoS Slowhttptest	32635	7027
DoS Slowloris	37236	10350
DDoS	754735	510922
Heartbleed	28412	20884
SSH Patator	65695	97616
FTP Patator	43507	67229
Botnet	4788	5083
Infiltration	29881	29873
Port Scan	162630	160677
Web Attack-Brute force	19755	10304
Web Attack-XSS	6361	3277
Web Attack-SQL Injection	67	59

TABLE 3: CICIDS-2018 pcap file details

Day/Date	File Size	Activity
Wednesday/ Feb 14, 2018	40 GB	SSH and FTP Patator, and Benign
Thursday/ Feb 15, 2018	41.3 GB	DoS GoldenEye, DoS Slowloris, and Benign
Friday/ Feb 16, 2018	38.6 GB	DoS Slowhttptest, DoS Hulk, and Benign
Thursday/ Feb 22, 2018	50.3 GB	Web Attack and Benign
Friday/ Feb 23,2018	60 GB	Web Attack and Benign
Wednesday/ Feb 28, 2018	53.3 GB	Infiltration and Benign

TABLE 4: Packet details per attack type in CICIDS-2018

Attack Type	Number of packets	
	Forward	Backward
DoS GoldenEye	154774	98976
DoS Hulk	1028619	107067
Infiltration	236560	86054
FTP Patator	193360	193360
Web Attack-Brute force	20735	14158
Web Attack-XSS	22289	11586

4.2 *Image data creation*

After extracting packet-based feature data from both CICIDS-2017 and CICIDS-2018 pcap files, the image builder component was utilized to create image data sets following the procedure outlined in Section 3. We used the CICIDS-2017 data for the development of the network intrusion detector in our framework. The CICIDS-2018 data, representing a different network environment, was used to generate images to evaluate the transferability of the trained detector. It is to be noted that the objective of our approach is to detect maliciousness in a network communication as early as possible. By keeping the value of P (indicating number of sequential packets in the same flow) small, we are able to create a smaller dimensional image. Table 5 shows the flow-related statistics for each attack type in CICIDS-2017 data, from which we obtain the value of P.

The table displays different statistics, including *average, median, and mode* for the number of packets in those flows. We selected the median number of packets per attack type

TABLE 5: Flow statistics per attack type in CICIDS-2017

Attack Type	Activity Label	Total Flow	Average Packets	Median Packets	Mode Packets
Benign	0	103138	85.76	15	14
DoS Slowloris	1	1554	30.62	10	7
DoS Slowhttptest	2	1586	25.07	13	2
DoS Hulk	3	128566	17.47	14	14
DoS GoldenEye	4	6704	15.83	14	14
Heartbleed	5	1656	29.76	21	6
FTP-Patator	6	475	233.13	6	4
SSH-Patator	7	457	357.35	6	4
Web Attack-Brute Force	8	222	135.45	17	2
Web Attack-XSS	9	69	139.68	3	2
Web Attack-SQL Injection	10	2	63	63	63
Infiltration	11	4254	14.04	6	2
Botnet	12	736	13.41	9	9
PortScan	13	7674	42.13	22	6
DDoS	14	167232	7.56	2	2
Average		28288.33	80.68	**14.73**	10.06

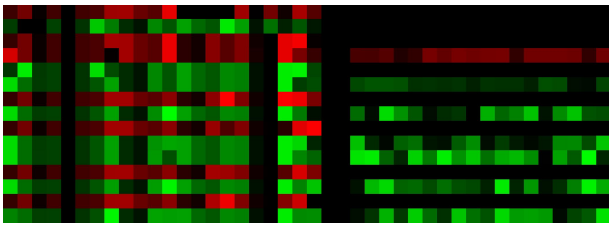
TABLE 6: Data sets (image representations) of generated images with different number of packets

Activity Label	Number of samples per image representation											
	1 pkt	2 pkt	3 pkt	4 pkt	5 pkt	6 pkt	...	11 pkt	12 pkt	13 pkt	14 pkt	15 pkt
0	103138	102889	102688	101637	101547	101320	...	92548	89580	83490	74974	61159
1	1554	1529	1522	1342	1331	1269	...	769	721	651	606	585
2	1586	1581	1296	1109	1093	1074	...	883	830	796	774	737
3	128566	128339	128207	127508	127457	127297	...	120492	116460	102844	83426	53841
4	6704	6607	6580	6539	6484	6312	...	5852	5645	4864	3849	2301
5	1656	1656	1652	1650	1642	1632	...	1334	1304	1271	1245	1211
6	475	464	461	437	275	260	...	121	106	96	93	89
7	457	457	454	420	265	254	...	142	135	126	120	115
8	222	221	166	160	147	144	...	135	133	131	130	127
9	69	66	41	27	21	19	...	17	16	16	15	15
10	2	2	2	2	2	2	...	2	2	2	2	2
11	4254	3798	3334	2921	2550	2253	...	1203	1057	933	834	743
12	736	736	736	736	736	736	...	116	45	45	44	44
13	7674	7671	7506	7386	7306	7284	...	5953	5813	5730	5633	5466
14	167232	167118	33812	33507	33073	32996	...	30490	28721	25573	20575	15636

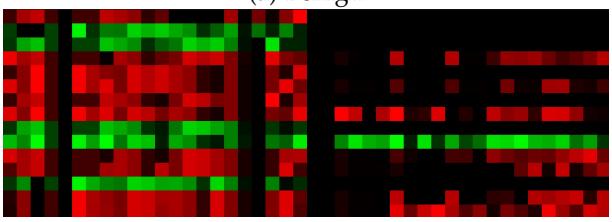
to determine the value of P for image construction as it can be observed that there is a significant variation in the average number of packets in the flows per attack type. This fluctuation is due to the presence of outliers in some of the attack types with a large number of packets. We used the average median value of 15 for P and set the dimension of the images to be generated in the image builder component to $(15 \times 1486 \times 3)$. For each flow, we created various images ranging from one packet to P number of sequential packets, resulting in up to P number of images per flow. The average image generation process time per image was 0.04 milliseconds. The resulting image data set from the image builder component is presented in Table 6. Each column in this table denotes an image representation with a fixed number of packets extracted from the flows belonging to various network activities (one benign and 14 attack types). For instance, the last column (15 pkt) represents the image data containing 61159 images belonging to the benign class (label 0) and 15636 images belonging to the *DDoS* class (label 14), along with images from other classes as shown in that column. All these images (in the last column) contain the first 15 sequential packets found in the communication of the respective flows. Figure 5 shows some samples of the image data (with truncated width).

Next, we created three separate data sets, one each for training, validating, and testing the network intrusion detector. We took the following into account while splitting the image data:

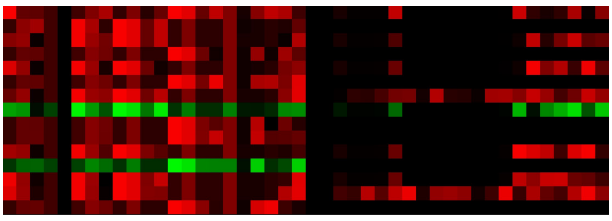
- To ensure that our CNN-based network intrusion detection model is trained effectively, we excluded the *Web Attack-XSS*, *Web Attack-SQL Injection*, and *Botnet* attack classes since there were small numbers of images for these classes across the different data sets, as shown in the highlighted rows of Table 6.
- To ensure that the model is not biased towards any particular representation, we sampled the same number of images during training from each representation (1 pkt to 14 pkt) as found in the last data set (15 pkt).
- To ensure that the model is not validated or tested using different image representations of the same flows that were used during its training, we created three data sets (training, validation, and testing), each containing all P image representations of distinct flows.
- To ensure that the model is trained with a balanced data set, we sampled (near-)equal number of images from each attack type in a way that the total number of images belonging to the malicious class (i.e., all attack types combined) matches that of the benign class. In particular, we sampled 100 same flow images from each attack type, except for the *FTP-Patator* class, for which we sampled 89 images from each of



(a) Benign



(b) DDoS



(c) Heartbleed

Fig. 5: Image samples generated using image builder component with 15 sequential packets

the 15 representations.

Finally, after taking the aforementioned factors into consideration, the samples were divided into three data sets, training (70%), validation (15%), and testing (15%).

4.3 Network intrusion detection model development

We tested various CNN architectures such as shallow (with only one convolutional layer) to deep (with seven convolutional layers) architectures along with different padding and stride strategies. Also, different pooling layer strategies, including maximum and average pooling were experimented to find the optimal architecture of the CNN model for network intrusion detector. The resulting optimal structure is depicted in Figure 6. The convolutional layers in the model use *same* padding strategy so that the output image size is the same as the input size for each convolutional layer, and the stride was set to the default value of $S = (1, 1)$. To speed up the learning process and improve stability, a batch normalization layer was used after the first pooling layer. To prevent overfitting, two dropout layers with a drop rate of 20% were included during training.

Since the CNN model is used for binary classification (benign or malicious), we used *Binary Crossentropy* as the loss function and the *Sigmoid* activation function in the last dense layer to produce output values in the range of $[0, 1]$. We optimized the important hyperparameters of the CNN architecture using the training and validation data sets. To achieve this, we implemented the CNN model in

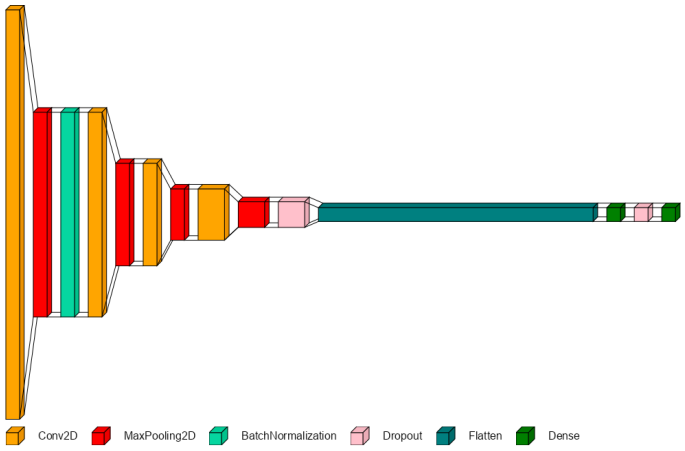
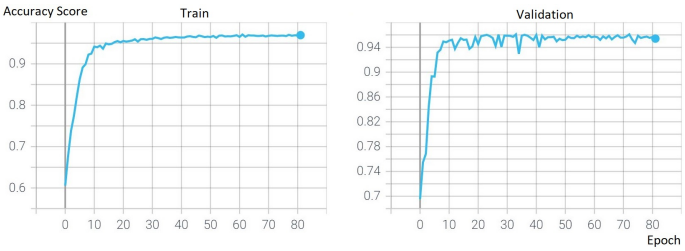
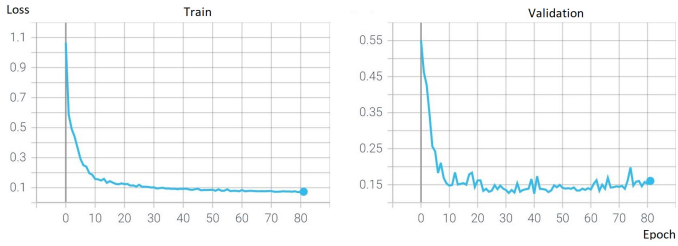


Fig. 6: Architecture of the CNN model for network intrusion detector



(a) Accuracy



(b) Loss

Fig. 7: Learning curves of the optimized model

Tensorflow [39] and utilized KerasTuner [40] and TensorBoard for hyperparameter optimization. Table 7 shows the hyperparameter values used in our experiments and the best values selected for the model. The number of *epochs* for training was set to 100 for all trials. The learning curves for the best parameter values are shown in Figure 7. We then used this optimized model on the testing data set to evaluate the performance of our approach using different test case scenarios.

5 Results and Analysis

This section presents the results and analysis of the experiments conducted using our AI-enabled SPIN-IDS framework. We first demonstrate the efficacy of our approach using the test data set (CICIDS-2017). We analyze the performance across the different image representations and extract insights into the potential timing of malicious information transmission. We then assess the detection capabilities across the different types of network attacks and perform

TABLE 7: Hyperparameter values for the CNN-based network intrusion detector model

Hyperparameter	Hyperparameter Values	**Best Value**
Activation function	[ReLU, tanh, LeakyReLU]	**ReLU**
Kernel initializer	[GlorotNormal [41], HeNormal [42], RandomUniform]	**GlorotNormal**
Convolutional layer 1 filter	(min_value=16, max_value=128, step=16)	**96**
Convolutional layer 1 kernel	[(3, 3), (4, 4), (5, 5)]	**(5, 5)**
Convolutional layer 2 filter	(min_value=32, max_value=256, step=32)	**128**
Convolutional layer 2 kernel	[(3, 3), (4, 4), (5, 5)]	**(5, 5)**
Convolutional layer 3 filter	(min_value=32, max_value=512, step=32)	**192**
Convolutional layer 3 kernel	[(3, 3), (4, 4), (5, 5)]	**(3, 3)**
Convolutional layer 4 filter	(min_value=32, max_value=512, step=32)	**384**
Convolutional layer 4 kernel	[(3, 3), (4, 4), (5, 5)]	**(4, 4)**
Dense layer 1 units	(min_value=16, max_value=128, step=16)	**64**
Batch size	[64, 128, 256, 512]	**256**
Learning rate	[1e-2, 1e-3, 1e-4]	**0.001**
Optimizer	[SGD, RMSprop, Adam]	**RMSprop**

statistical analysis to detect deviations from benign behavior during network attack communications. We also validate the resilience of our approach against adversarially crafted examples and examine the adaptability of the trained network intrusion detector in a new target network environment (CICIDS-2018). Finally, we compare our approach with other packet-based methods from recent literature.

5.1 Performance across different image representations

As described in Section 4, the test data samples comprise of images from 15 different image representations for the *same flows*, i.e., from one packet to 15 sequential packets obtained from the same network flow in the respective images. We used each of the 15 representations as a subset of the test data. Subset 1 consisted of images generated from the first packet of the flows, subset 2 contained images created from the first and second packets of the *same flows*, and subsequent subsets with images in the same sequential pattern for the first 15 packets. In each of these 15 subsets, we kept near-equal number of samples from each attack type, all of which constituted the malicious class. Also, we selected near-equal number of samples from the benign class to keep the balance with malicious class samples. Each subset contained 328 images, with 165 belonging to the benign class and 163 to the malicious class.

Table 8 shows the model's performance on these subsets of images using different metrics. It can be observed that the performance improves as the number of packets in flows increases, as evidenced by all the metric values. The best performance across all the metrics is observed with nine sequential packets in the image data with a TPR of 98.77%. Figure 8 provides a visual of the performance improvement with the increase in the number of packets in the images. The performance is shown to plateau after nine packets in the image representation, indicating that the maliciousness in network traffic can be identified with high accuracy within the first nine sequential packets of the evolving flow. It can also be observed that the model cannot accurately identify malicious patterns in images when only the first packet of malicious traffic is transmitted, as indicated by the TPR of 52.76%. This result indicates that the first packet of the communication may not have malicious intent and that the malicious packets are transmitted later in the communication. Remarkably, the model achieves a TPR of 92.02%

TABLE 8: Model performance on different image representations of CICIDS-2017 test data

Subset	Accuracy	Precision	F1	TNR	FNR	TPR (Recall)	FPR (FAR)
1	0.7439	0.9247	0.6719	0.9576	0.4724	0.5276	0.0424
2	0.8933	0.9638	0.8837	0.9697	0.1840	0.8160	0.0303
3	0.9177	0.9658	0.9126	0.9697	0.1350	0.8650	0.0303
4	0.9543	0.9868	0.9524	0.9879	0.0798	0.9202	0.0121
5	0.9665	0.9872	0.9655	0.9879	0.0552	0.9448	0.0121
6	0.9787	0.9815	0.9785	0.9818	0.0245	0.9755	0.0182
7	0.9787	0.9815	0.9785	0.9818	0.0245	0.9755	0.0182
8	0.9817	0.9758	0.9817	0.9758	0.0123	0.9877	0.0242
9	0.9878	0.9877	0.9877	0.9879	0.0123	0.9877	0.0121
10	0.9817	0.9816	0.9816	0.9818	0.0184	0.9816	0.0182
11	0.9817	0.9816	0.9816	0.9818	0.0184	0.9816	0.0182
12	0.9817	0.9816	0.9816	0.9818	0.0184	0.9816	0.0182
13	0.9817	0.9816	0.9816	0.9818	0.0184	0.9816	0.0182
14	0.9817	0.9816	0.9816	0.9818	0.0184	0.9816	0.0182
15	0.9817	0.9816	0.9816	0.9818	0.0184	0.9816	0.0182

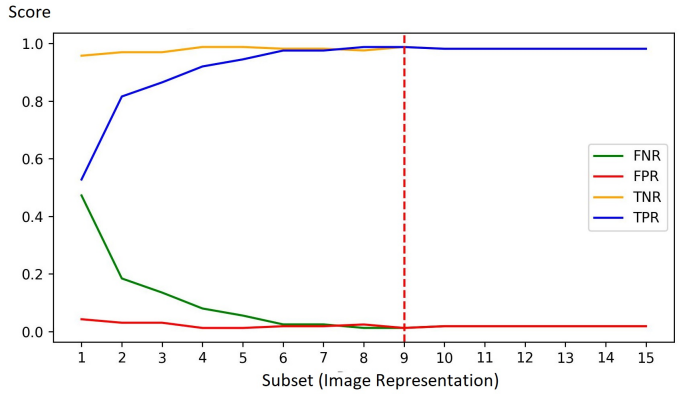


Fig. 8: Visual of model performance on different image representations

after the transmission of only four packets. This result indicates that typically there is no maliciousness in the first three packets, as they are a part of the TCP three-way handshake used to establish a reliable connection. Experiment results show that our methodology is able to accurately detect malicious activity early in the communication, within the first nine packets compared to flow-based approaches that have to wait for, on an average, 80 packets to be transmitted (see Table 5). Next, we present the performance of the model in detecting images belonging to the various types of network attacks.

TABLE 9: Detection analysis per attack type

Attack	Size	Accuracy	Precision	F1	TNR	FNR	TPR	FPR
DoS Slowloris	1728	0.9838	0.9860	0.9838	0.9861	0.0185	0.9815	0.0139
DoS Slowhttptest	1680	0.9875	0.9858	0.9875	0.9857	0.0107	0.9893	0.0143
DoS Hulk	164735	0.9787	0.9936	0.9855	0.9819	0.0223	0.9777	0.0181
DoS GoldenEye	11864	0.9729	0.9612	0.9733	0.9602	0.0143	0.9857	0.0398
Heartbleed	2670	0.9843	0.9850	0.9843	0.9850	0.0165	0.9835	0.0150
FTP-Patator	150	1	1	1	1	0	1	0
SSH-Patator	134	1	1	1	1	0	1	0
Web Attack-Brute Force	78	0.9872	0.9750	0.9873	0.9744	0	1	0.0256
Infiltration	2854	0.9926	0.9951	0.9926	0.9951	0.0098	0.9902	0.0049
PortScan	12492	0.9789	0.9796	0.9789	0.9797	0.0218	0.9782	0.0203
DDoS	64512	0.9854	0.9812	0.9851	0.9820	0.0110	0.9890	0.0180

5.2 Performance in detecting different network attacks

For this experiment, we considered all the remaining images with nine sequential packets from all attack types that were not used during testing and validation of the network intrusion detection model. Table 9 shows the performance metric values obtained using the trained model with the nine sequential packets image data for each attack type. The average TPR or recall score across all attack types exceeds 98.5%, indicating that the model successfully identified the underlying malicious patterns for the attacks. The model was able to attain 100% accuracy in detecting both *SSH-Patator* and *FTP-Patator* attack images, among other results. The findings indicate that our SPIN-IDS framework with the CNN-based network intrusion detector and image representation with nine sequential packets performs very well in detecting all types of network attacks and keeping the false negative and positive values significantly low.

Next, we perform a statistical analysis to determine when the deviation in benign behavior occurs in the malicious flows. We use a statistical measure for image comparison to assess how malicious traffic varies compared to a normal (benign) traffic flow. For this, a representative image is created for each attack type by averaging the pixel values of all sample images that belong to a specific image representation (one packet, two packets, ..., 15 packets) of the flow. Likewise, a representative image is generated for the corresponding image representation of normal traffic. The two representative images are then compared using the peak signal-to-noise ratio (PSNR) similarity metric [43]. The PSNR score is determined using the following equation:

$$\text{PSNR} = 10\log_{10}\frac{(2^d-1)^2 PQ}{\sum_{i=1}^{P}\sum_{j=1}^{Q}(p[i,j]-p'[i,j])^2} \quad (9)$$

where d refers to the number of channels in the images, Q and P represent the width and height of the images, respectively, and $p[i,j]$ and $p'[i,j]$ denote the pixel values in the i-th row and j-th column of the normal and malicious representative images, respectively. A PSNR score of 1 indicates that the two images are the same, while a score of 0 implies complete dissimilarity between the two images.

We conducted our analysis using all 15 subsets (image representations) for each attack type. We present our findings for the four attack types, namely, *SSH-Patator, DoS Slowloris, DoS Slowhttptest*, and *Infiltration*. Figure 9 shows the PSNR scores obtained for these attack types relative to the benign image. Figure 10 shows the recall scores or the detection rate for the four attack types using each of the subsets for a correlation analysis. We also observed similar patterns for the other attack types.

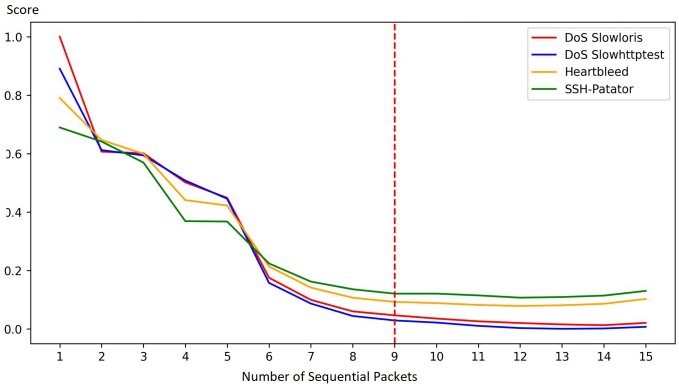


Fig. 9: PSNR scores for different image representations of various attack types compared to benign traffic

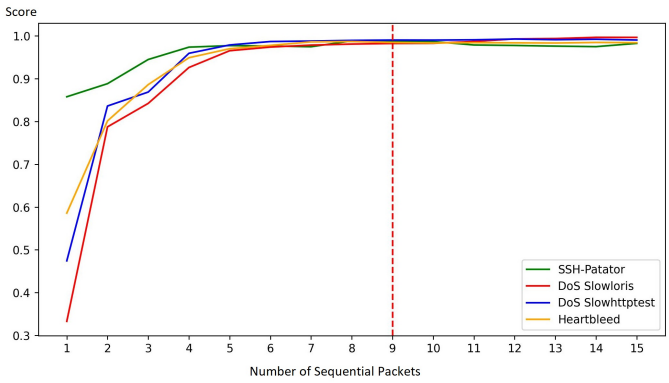


Fig. 10: Recall scores for different image representations of various attack types

The recall and PSNR score graphs presented for each attack type reveal a strong correlation between specific packets and the model's performance, consistent with the PSNR scores. For example, the recall score for the *DoS Slowloris* attack type for the subset containing only one packet is the lowest (33.3%) compared to other attack types. From the PSNR score, it is apparent that there are significant similarities between the representative image of the *DoS Slowloris* attack type with one packet and that of the benign class. However, when the second packet is added to the *DoS Slowloris* flow, the recall score improves sharply to 78.7%, which is the steepest increase, and the PSNR score decreases significantly to 60.6%, which is the steepest decline in the PSNR graph. Similar observations can be made for other attack types. Though the first three packets are a part of the TCP three-way handshake, there are certain bytes in packet headers relaying critical information that help in identifying ongoing traffic's maliciousness such as time-to-live (TTL) value, IP flags, urgent pointer, and window size bytes. Our approach captures both the header and payload data, and hence, the network intrusion detector is able to detect deviations from a benign network communication in such samples. Importantly, it is observed that the PSNR and recall scores plateau after nine sequential packets have been transmitted for each attack type, strongly indicating that packets with malicious intent were a part of this initial

transmission in the evolving flow. Next, we evaluate the robustness of our approach against adversarial examples crafted for deceiving the NIDS.

5.3 Performance against adversarial examples

Adversarial examples are created with an objective of evading the NIDS detection mechanism. These can be generated by making subtle modifications to legitimate network traffic or by carefully crafting malicious traffic that fools the underlying ML/DL algorithms. These modifications, which include changing packet bytes or perturbing their timing/order, can exploit vulnerabilities in the NIDS, leading to false negative decisions. This can enable attackers to infiltrate networks undetected and disrupt operations.

To evaluate the robustness of our approach, we crafted samples with multiple valid perturbations drawing on domain knowledge, literature studies [44], [45], [46], and insights from security experts at a collaborating cybersecurity operations center. Since the adversary's control in an evasion attack is limited to network packets sent from a single direction (i.e., the source), perturbations are applied to the forward network packets of each attack type. Adversarial examples are crafted, while preserving their communication functionality. Table 10 presents details on the perturbations, their corresponding values, and the specific packet bytes requiring modification to maintain packet functionality. We generated adversarial examples from the nine-sequential packets data set for each attack type using the above-mentioned perturbations. We present two different perturbation use cases, among various others that we examined. (i) We perturbed half of the forward packets in each flow (containing nine sequential packets) by applying all five perturbation methods on each of the selected packets. (ii) We perturbed all the forward packets in each flow using the five perturbation methods.

To compare the performance of our approach, we tested the perturbed samples against the packet and flow-based NIDS used in recent literature studies [47], [46], which included decision tree, random forest, support vector machine, k-nearest neighbor, and deep neural network models. The evasion rate (ER) against these ML/DL-based NIDS ranged from 70% to 99% across different attack types. In contrast, using our approach, the ER of these samples was significantly reduced to a maximum of 2.04%, which was observed only for certain attack types, including *DoS Slowloris, DoS Hulk, Infiltration,* and *Port Scan*. Table 11 shows the results from our approach using the SPIN-IDS framework for these two cases, demonstrating strong resilience against evasion attacks.

5.4 Performance in a new target environment

To gauge the domain adaptability of our trained model, we conducted experiments using data from a new network environment. We used CICIDS-2018 data, which was processed using the SPIN-IDS framework. First, the packet parser component extracted packet-based features from the pcap files, which was followed by the generation of images using the image builder component. These images were then passed through the network intrusion detector model, which was trained using CICIDS-2017 image data. Our experiments involved a similar setup as used with the CICIDS-2017 data. We considered various image representations with different numbers of sequential packets extracted from the flows for evaluating the performance of the model. Table 12 presents the results from these experiments showing a similar performance of the trained model on a new target network environment data set as observed on the source environment data set. The image representations with eight and nine sequential packets have the highest performance metric values among others, thereby strongly indicating that the malicious intent can be accurately detected early in a network communication. Table 13 shows the performance of the trained network intrusion detector in detecting images with nine sequential packets belonging to a sample set of network attack types in the CICIDS-2018 data set. The results show that despite being trained only on the CICIDS-2017 image data set, our model was able to achieve a good performance in detecting attacks in a different environment. Our approach using the SPIN-IDS framework has shown that the network intrusion detection model can perform well on data from a different target environment without the need to retrain and hence, reinforcing its potential as an effective intrusion detection tool for cybersecurity operations centers.

5.5 Performance comparison with other methods

The primary goal of our SPIN-IDS framework is to accurately detect malicious network traffic in (near) real-time by examining a minimum number of packets within the ongoing traffic flows. It is to be noted that the comparison of SPIN-IDS with other state-of-the-art approaches is not straightforward due to differences in methodology and data representation. Despite this, we compare results obtained using our approach with other existing methods in the literature that leveraged packet information in their proposed NIDS, using the same data set (CICIDS-2017) and key performance metrics. Table 14 presents the performance metrics of our methodological framework and those reported by the state-of-the-art DL-based models, namely, AEIDS [48], HAST-II [49], PL-RNN [50], Packet2Vec [51], PayloadEmbeddings [52], and PBCNN [30]. The table also shows each model's training time and testing (inference) time per unit in milliseconds, where a unit represents either a packet or an image, based on the data format used in the respective method. As indicated in the table, SPIN-IDS outperforms the other methods in terms of key performance metric values and testing time.

6 Conclusions and Future Directions

In this study, we proposed an AI-enabled SPIN-IDS framework, aimed towards (near-)real-time detection of network attacks. Our methodology addressed the current limitations in packet-based NIDS by analyzing header and payload data and considering temporal connections among packets of the same communication as they are transmitted through the network. The sequential packets in an evolving flow are transformed into a two-dimensional image, and then passed through the CNN-based intrusion detector in our framework to detect maliciousness. The results demonstrated

TABLE 10: Summary of the perturbation methods used for generating adversarial examples

Perturbation method	Description	Affected packet bytes
Adding payload data	Injecting 40 bytes of dead payload bytes such as 0xFF at the end of the original packet payload.	It affects payload bytes and TCP checksum bytes from the TCP header (byte number 17 and 18).
Changing delta time	The delta time feature of the selected packet in a flow is increased by adding a delay, e.g., 0.001 ms.	It only changes the temporal relationship between packets of a flow and does not affect the packet bytes.
Changing window size	Increasing or decreasing (+/- 256) the window size bytes. Any valid perturbation to these bytes will result in a final window size value between 1-65535.	It changes the following bytes from the TCP header: window size bytes (byte number 15 and 16) and TCP checksum bytes (byte number 17 and 18).
Changing time-to-live (TTL)	Decreasing or increasing (+/- 1) the TTL of the packets in the range of 0-255 (The range for TTL is from 0 to 255 and a recommended initial value for TTL is 64).	It changes the following bytes from the IP header: TTL byte (byte number 9) and IP checksum bytes (byte number 11 and 12).
Modifying fragmentation	Modifying the fragmentation bytes from do not fragment to do fragment. This perturbation can be applied to packets where fragmentation is turned off.	It changes the following bytes from the IP header: fragmentation bytes (byte number 7 and 8), TTL byte (byte number 9), and IP checksum bytes (byte number 11 and 12).

TABLE 11: Model performance against adversarial attacks

Attack	Percentage of forward packets perturbed	
	50%	100%
	ER (%)	ER (%)
DoS Slowloris	0	2.04
DoS Slowhttptest	0	0
DoS Hulk	2.04	2.04
DoS GoldenEye	0	0
Heartbleed	0	0
FTP-Patator	0	0
SSH-Patator	0	0
Web Attack-Brute Force	0	0
Infiltration	0	2.04
PortScan	0	2.04
DDoS	0	0

TABLE 12: Domain adaptability results using different image representations of CICIDS-2018 data

Subset	Size	Accuracy	Precision	F1	TNR	FNR	TPR	FPR
1	230650	0.8207	0.9371	0.8171	0.9399	0.2756	0.7244	0.0601
2	185787	0.8462	0.9348	0.8034	0.9604	0.2956	0.7044	0.0396
3	178059	0.9004	0.9519	0.8725	0.9701	0.1946	0.8054	0.0299
4	175466	0.9524	0.9481	0.9432	0.9627	0.0617	0.9383	0.0373
5	174967	0.9625	0.9619	0.9549	0.9729	0.0519	0.9481	0.0271
6	174397	0.9687	0.9676	0.9624	0.9769	0.0427	0.9573	0.0231
7	171170	0.9719	0.9717	0.9669	0.9791	0.0379	0.9621	0.0209
8	169967	0.9743	0.9736	0.9699	0.9804	0.0339	0.9661	0.0196
9	160211	0.9744	0.9721	0.9677	0.9817	0.0368	0.9632	0.0183
10	158801	0.9740	0.9727	0.9671	0.9822	0.0385	0.9615	0.0178
11	140846	0.9746	0.9696	0.9627	0.9844	0.0442	0.9558	0.0156
12	136870	0.9742	0.9726	0.9622	0.9858	0.0480	0.9520	0.0142
13	130653	0.9750	0.9788	0.9649	0.9883	0.0487	0.9513	0.0117
14	121919	0.9737	0.9831	0.9652	0.9898	0.0520	0.9480	0.0102
15	107890	0.9724	0.9869	0.9675	0.9904	0.0511	0.9489	0.0096

TABLE 13: Detection analysis per attack type (CICIDS-2018)

Attack	Size	Accuracy	Precision	F1	TNR	FNR	TPR	FPR
DoS Slowloris	6186	0.9510	0.9664	0.9505	0.9672	0.0650	0.9350	0.0328
DoS GoldenEye	36513	0.9714	0.9665	0.9699	0.9698	0.0267	0.9733	0.0302
SSH-Patator	84921	0.9782	0.9709	0.9782	0.9707	0.0143	0.9857	0.0293
Web Attack-Brute Force	545	0.9450	0.9542	0.9434	0.9567	0.0672	0.9328	0.0433
Web Attack-XSS	325	0.9508	0.9321	0.9497	0.9349	0.0321	0.9679	0.0651
Infiltration	2181	0.9431	0.9616	0.9389	0.9668	0.0829	0.9171	0.0332

that our SPIN-IDS framework with the network intrusion detector and nine sequential packet image representation of the data set performs very well in detecting network attacks. The recall score ranged between 97.7% to 99% across all attack types, indicating that the model successfully identified the underlying malicious patterns of the attacks. The findings indicate that the malicious intent can be detected with a very high accuracy by the transmission of the ninth packet in the two-way communication. Thereby, the malicious activity can be detected very early compared to flow-based approaches that, on an average, would have to wait for 80 packets or more to be transmitted for analysis. The trained network intrusion detector model demonstrated a high transferability property with an average recall or detection rate of over 95% on a new target data set, enabling cybersecurity teams to potentially deploy the trained model in different environments without the need for training from scratch. The SPIN-IDS framework was found to be robust against adversarial examples. The network intrusion detector was able to accurately detect network attacks with multiple carefully crafted perturbations in the packets compared to other ML/DL-based NIDS, which consider these packets independent during evaluation, resulting in a high false negative rate.

To further advance the research and benefit practitioners, the effectiveness of the SPIN-IDS framework could be evaluated in a real-world network environment, while performing a red team evaluation using AI-enabled adversarial agents. Such an evaluation would provide valuable insights into the resilience of the framework and shed light on the level of sophistication required for circumventing its intrusion detection mechanism through adversarial evolution.

TABLE 14: Comparison of SPIN-IDS with other packet-based approaches

Methods	Performance Metrics					
	Accuracy	Precision	Recall	F1	Training Time	Testing Time
PL-RNN	0.6825	0.8335	0.5638	0.6934	720000	2.91
Packet2Vec	0.6625	0.8435	0.7284	0.6273	515000	3.54
HAST-II	0.6423	0.8856	0.6898	0.7498	676000	4.2
AEIDS	0.7736	0.6325	0.5872	0.6182	**275800**	2.65
PayloadEmbeddings	0.9538	0.9536	0.9535	0.9534	355000	2.73
PBCNN	0.9821	0.9831	0.983	0.9835	368000	7.89
SPIN-IDS	**0.9878**	**0.9877**	**0.9878**	**0.9878**	314800	**0.12**

ACKNOWLEDGMENTS

The authors would like to thank Dr. Tapas K. Das, Soumyadeep Hore, and Diwas Paudel at the University of South Florida for many helpful discussions. This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-22-2-0045, as well as the U.S. Army Combat Capabilities Development Command C5ISR Center under Support Agreement No. USMA21056. The views and conclusions expressed in this

paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.

REFERENCES

[1] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2023.

[2] W. Lee, S. Stolfo, P. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, "Real time data mining-based intrusion detection," vol. 1, 02 2001, pp. 89–100 vol.1.

[3] J. Liu, X. Song, Y. Zhou, X. Peng, Y. Zhang, P. Liu, D. Wu, and C. Zhu, "Deep anomaly detection in packet payload," *Neurocomputing*, vol. 485, pp. 205–218, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221016349

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

[5] "Zeek," [Online].Available:https://zeek.org, 2020, accessed: Dec. 11, 2022.

[6] P. Cheng, M. Han, and G. Liu, "Desc-ids: Towards an efficient real-time automotive intrusion detection system based on deep evolving stream clustering," *Future Generation Computer Systems*, vol. 140, pp. 266–281, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X22003351

[7] C. Kreibich, M. Handley, and V. Paxson, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," in *Proc. USENIX Security Symposium*, vol. 2001, 2001.

[8] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[9] Y. Jia, M. Wang, and Y. Wang, "Network intrusion detection algorithm based on deep neural network," *IET Information Security*, vol. 13, no. 1, pp. 48–53, 2019.

[10] S. Hettich, "Kdd cup 1999 data," *The UCI KDD Archive*, 1999.

[11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[12] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[13] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48 697–48 707, 2018.

[14] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48 231–48 246, 2018.

[15] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2018, pp. 178–183.

[16] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[17] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *2016 15th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2016, pp. 195–200.

[18] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41 238–41 248, 2018.

[19] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE access*, vol. 8, pp. 42 169–42 184, 2020.

[20] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*. IEEE, 2019, pp. 0452–0457.

[21] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, "Dl-ids: Extracting features using cnn-lstm hybrid network for intrusion detection system," *Security and communication networks*, vol. 2020, pp. 1–11, 2020.

[22] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE access*, vol. 8, pp. 32 464–32 476, 2020.

[23] Y. Yu and N. Bian, "An intrusion detection method using few-shot learning," *IEEE Access*, vol. 8, pp. 49 730–49 740, 2020.

[24] Z.-Q. Qin, X.-K. Ma, and Y.-J. Wang, "Attentional payload anomaly detector for web applications," in *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part IV 25*. Springer, 2018, pp. 588–599.

[25] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "A detailed analysis of the cicids2017 data set," in *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*. Springer, 2019, pp. 172–188.

[26] J. Liu, X. Song, Y. Zhou, X. Peng, Y. Zhang, P. Liu, D. Wu, and C. Zhu, "Deep anomaly detection in packet payload," *Neurocomputing*, vol. 485, pp. 205–218, 2022.

[27] Y. Farrukh, I. Khan, S. Wali, D. Bierbrauer, and N. Bastian, "Payload-byte: A tool for extracting and labeling packet capture files of modern network intrusion detection datasets," 09 2022.

[28] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3366–3383.

[29] X. Zhang, J. Chen, Y. Zhou, L. Han, and J. Lin, "A multiple-layer representation learning model for network-based attack detection," *IEEE Access*, vol. 7, pp. 91 992–92 008, 2019.

[30] L. Yu, J. Dong, L. Chen, M. Li, B. Xu, Z. Li, L. Qiao, L. Liu, B. Zhao, and C. Zhang, "Pbcnn: packet bytes-based convolutional neural network for network intrusion detection," *Computer Networks*, vol. 194, p. 108117, 2021.

[31] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32, p. 200892, 2020.

[32] M. M. Alani, *Guide to OSI and TCP/IP models*. Springer, 2014.

[33] M. Masum and H. Shahriar, "Tl-nid: Deep neural network with transfer learning for network intrusion detection," in *2020 15th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2020, pp. 1–7.

[34] F. Özyurt, E. Ava, and E. Sert, "Uc-merced image classification with cnn feature reduction using wavelet entropy optimized with genetic algorithm," 2020.

[35] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020.

[36] "dpkt 1.9.2 documentation," [Online].Available:https://dpkt.readthedocs.io/en/latest/, 2009, accessed 16-Feb-2023.

[37] "Scapy 2.5.0 documentation," [Online].Available:https://scapy.readthedocs.io/en/latest/, 2010, accessed 11-Jan-2023.

[38] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cicids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 7–12.

[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[40] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.

[41] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,"

CoRR, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852
[43] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[44] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating dnn-based traffic analysis systems in real-time with blind adversarial perturbations." in *USENIX Security Symposium*, 2021, pp. 2705–2722.
[45] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1962–1976, 2021.
[46] H. Yan, X. Li, W. Zhang, R. Wang, H. Li, X. Zhao, F. Li, and X. Lin, "Automatic evasion of machine learning-based network intrusion detection systems," *IEEE Transactions on Dependable and Secure Computing*, 2023.
[47] B.-E. Zolbayar, R. Sheatsley, P. McDaniel, M. J. Weisman, S. Zhu, S. Zhu, and S. Krishnamurthy, "Generating practical adversarial network traffic flows using nidsgan," *arXiv preprint arXiv:2203.06694*, 2022.
[48] B. A. Pratomo, P. Burnap, and G. Theodorakopoulos, "Unsupervised approach for detecting low rate attacks on network traffic with autoencoder," in *2018 international conference on cyber security and protection of digital services (Cyber Security)*. IEEE, 2018, pp. 1–8.
[49] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE access*, vol. 6, pp. 1792–1806, 2017.
[50] H. Liu, B. Lang, M. Liu, and H. Yan, "Cnn and rnn based payload classification methods for attack detection," *Knowledge-Based Systems*, vol. 163, pp. 332–341, 2019.
[51] E. L. Goodman, C. Zimmerman, and C. Hudson, "Packet2vec: Utilizing word2vec for feature extraction in packet data," *arXiv preprint arXiv:2004.14477*, 2020.
[52] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan, and R. Agrawal, "Intrusion detection using payload embeddings," *IEEE Access*, vol. 10, pp. 4015–4030, 2022.

Jalal Ghadermazi is a Ph.D. Candidate in the Industrial and Management Systems Engineering department at the University of South Florida, specializing in the field of cybersecurity. His research interests span the areas of robustness, real-time analytics, and adversarial machine learning in cybersecurity applications. He integrates principles from computer science, operations research, data science, and information technology to develop innovative solutions for enhancing cybersecurity.

Ankit Shah is an Assistant Professor of Industrial and Management Systems Engineering at the University of South Florida. His research interests lie at the intersection of computer science, operations research, data science, and information technology, with a strong focus on artificial intelligence for cybersecurity. His current research work is in the area of deep reinforcement learning and adversarial machine learning for threat reduction and secure systems.

Nathaniel D. Bastian is Chief Data Scientist, Data & Decision Sciences Division Chief, and Senior Research Scientist at the Army Cyber Institute at West Point, also serving as Assistant Professor of Operations Research and Data Science at the United States Military Academy (USMA). He holds a Ph.D. degree in Industrial Engineering and Operations Research from the Pennsylvania State University (PSU), a M.Eng. degree in Industrial Engineering from PSU, a M.S. degree in Econometrics and Operations Research from Maastricht University, and a B.S. degree in Engineering Management (Electrical Engineering) with Honors from USMA.