

Received 7 October 2024, accepted 29 October 2024, date of publication 1 November 2024, date of current version 11 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3489772

RESEARCH ARTICLE

A Proactive Model for Intrusion Detection Using Image Representation of Network Flows

RIMSHA SAEED¹, HASSAAN KHALIQ QURESHI², CHRISTIANA IOANNOU^{3,4},
AND MARIOS LESTAS^{3,5}, (Member, IEEE)

¹School of Interdisciplinary Engineering and Sciences, National University of Sciences and Technology, Islamabad 44000, Pakistan

²School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan

³Department of Electrical and Computer Engineering and Informatics, Frederick University, 1036 Nicosia, Cyprus

⁴School of Science, University of Central Lancashire Cyprus, 7080 Lamaca, Cyprus

⁵Frederick Research Center, 1036 Nicosia, Cyprus

Corresponding author: Rimsha Saeed (rsaeed.mscse21rcms@student.nust.edu.pk)

This research has received support from the project BRIDGE2HORIZON/0823A/004 (AUTOMETA), which is implemented under the Cohesion Policy Funds “THALIA 2021-2027” with EU co-funding.

ABSTRACT Many interconnected IoT devices driven by imperatives of efficiency and convenience often lack adequate security measures, making them susceptible to exploitation by cyber-criminals. Effective network security necessitates meticulous intrusion detection, which typically involves scrutinizing the network traffic using deep packet or stateful protocol inspection techniques. However, traditional inspection methods often require manual feature engineering, which can result in loss of payload information and thus, false alarms. In this study, a controlled testbed environment is established to capture botnet traffic. The paper introduces a detection approach that involves converting raw NetFlow data to IDX, short for ‘Index,’ image representations. A hybrid deep learning architecture is designed, integrating VGG19 and GRU structures to learn the spatial and temporal features, respectively. The detection results show that the proposed solution achieves 98.883% true positives rate and 0.9% false negatives rate, surpassing conventional anomaly detection. In addition, an adaptive sliding window technique is introduced for live intrusion detection and prevention. Through iterative testing and refinement, a runtime of 0.041 ms per image and 0.00171 ms per packet is achieved, confirming the robust nature of the proposed method.

INDEX TERMS Botnet detection, flow-to-image conversion, intrusion detection, intrusion prevention, sliding windows, spatial features, temporal features.

I. INTRODUCTION

In today’s interconnected world, security and integrity of computer networks have become paramount. As organizations and individuals rely heavily on networked systems for communication, data sharing, and essential services, the vulnerability to cyber threats has escalated exponentially. From large-scale data breaches to sophisticated hacking attempts, the realm of network intrusions has evolved into a complex and ever-present challenge [1], [2]. In response, the field of network intrusion detection has emerged as a crucial line of defense, aiming to detect and mitigate unauthorized access, malicious activities, and potential data breaches.

The associate editor coordinating the review of this manuscript and approving it for publication was Mueen Uddin¹.

By effectively identifying and responding to network intrusions, one can safeguard sensitive information, protect critical infrastructure, and preserve the trust of network users. This research endeavors to contribute to the field of network intrusion detection by addressing the challenges associated with accurately detecting and mitigating cyber threats in real-time.

Among the most prevalent and damaging types of network intrusions is the insidious menace of botnet attacks. A botnet refers to a network of compromised computers, known as “zombies” or “bots,” controlled remotely by a central command-and-control (C&C) server. These coordinated attacks can be orchestrated for various malicious purposes, such as distributed denial-of-service (DDoS) attacks, man-in-the-middle (MiTM) attacks, spam dissemination, data

exfiltration, and even crypto mining [3]. As per the statistics published by *Parachute*, botnets were responsible for 51% of all detected network attacks in 2022 [4]. The severity and abundance of botnet attacks despite the existing intrusion prevention infrastructure underscore the need for a robust network intrusion detection and prevention system capable of identifying and mitigating these evolving threats.

Traffic classification is an important aspect of botnet detection [5], [6]. It primarily analyzes specific fields within traffic packets to determine the likelihood of network vulnerability or anomalous activity that may pose a threat to network security. In order to detect and classify the abnormal traffic, the traffic packets are usually segmented into flows based on their five-tuple information (i.e., source IP, destination IP, source port, destination port, and protocol) and timestamp [7]. Presently, the port-based methods, payload-based methods, and statistical features-based methods are well-established traffic detection technologies employed for the identification and analysis of abnormal network behavior.

The port-based traffic detection method was widely adopted in the past due to the simplicity of network protocols and the utilization of fixed port numbers by specific applications [8]. However, with the advancement of dynamic port allocation and redirection, the approach fails to adequately capture the traffic attributes, resulting in declined detection effectiveness. The payload-based traffic detection method leverages information from the application layer, with deep packet inspection (DPI) technology being a notable example [9]. DPI involves decrypting and encrypting transmitted traffic data and effectively identifies malicious packets [10], [11]. Nonetheless, the growing use of encryption protocols such as Hypertext Transfer Protocol Secure (HTTPS) and increasing concerns for privacy have significantly limited the applicability of DPI. Additionally, the statistical feature-based traffic detection method relies on factors such as packet arrival time, packet size, and statistical characteristics of traffic packet fields to represent traffic attributes [12]. By employing artificially designed features and machine learning algorithms, this method offers relatively reliable techniques for analyzing and detecting abnormal traffic. However, accurate labeling of traffic data remains crucial when training supervised algorithms in the context of this approach.

Previous research has mainly focused on enhancing classification accuracy and other metrics by operating at the data link layer. Both traditional machine learning algorithms and various neural network algorithms in deep learning have utilized complex feature engineering to extract information from traffic data. However, feature engineering can lead to the loss or alteration of original temporal and spatial features of traffic packets. For example, Yeo et al. [13] extracted temporal features such as flow inter-arrival time, burst inter-arrival time, and flow duration, while Yu et al. [14] extracted temporal features like flow activation time, packet arrival time, and time interval, as well as spatial features such

as packet number, IP address, and transmission direction. Although these extracted traffic features enable classification algorithms to make use of missing data information, the classification accuracy and other metrics have maxed out, posing significant challenges for further enhancement.

A. CONTRIBUTIONS

In particular, the paper contributes: i) a traffic handling approach that eliminates the need for manual feature engineering and addresses the information loss problem by working directly with raw NetFlow data and generating IDX format images, a format designed to store multi-dimensional arrays, consisting of a header followed by binary data, ii) an integrated deep learning-based intrusion detection algorithm that learns the flow features two-fold, and iii) an adaptive sliding windows-based intrusion prevention framework that proactively monitors the incoming traffic and identifies network intrusions in real-time. Beyond the novelty of the integrated approach which is shown to outperform existing solutions, the novelty in the constituent elements includes the introduction of the adaptive sliding window approach, which enables our system to dynamically adjust its monitoring window size and type based on the changing network conditions. Furthermore, our work introduces a pioneering method for generating IDX format images from raw NetFlow data, obviating the need for manual feature engineering and overcoming information loss issues prevalent in conventional methods. The indexed image format allows for efficient storage and retrieval of large datasets, which is crucial for applications like machine learning and pattern recognition where performance is key. Additionally, the paper's contribution extends to the establishment of a comprehensive testbed, offering a new dataset that will be made publicly available and can be readily used by the research community.

It must be noted that the proposed method has the potential to be leveraged for emerging paradigms, such as misbehaving source detection in Collective Perception Message Dissemination for autonomous driving [15]. Such a system would rely on edge computing nodes, such as RSUs, to perform real-time detection of misbehaving messages at the network edge. The conversion to image format is particularly attractive for such applications, as it allows the creation of synthetic data for training through the employment of generative AI approaches [16] which have been particularly successful for image data.

The remainder of the paper is organized as follows. Section II details the previous research carried out for network intrusion detection and prevention using statistical methods, pattern matching, expert systems, and machine learning algorithms and the reliability achieved so far. Section III presents the proposed methodology and the implementation details, including the equipment used for setting up the testbed, data collection procedure, and detection approach being designed in this research, while Section IV

discusses the results. Finally, we draw concluding remarks in Section V.

II. RELATED WORK

Intrusion Detection Systems (IDS) constitute a class of software tools designed to identify instances of unauthorized access to computer networks or systems. The concept of IDS was first introduced by Anderson in 1980, where he emphasized the need to develop a threat model to identify types and sources of potential threats [17]. He categorized threats as “internal or external penetrations, and misfeasance”, and devised a security monitoring system to detect abnormal user behavior. Subsequent developments were nicely reviewed by Axelsson in [18], conducting an extensive survey and taxonomy of intrusion detection systems, delving into various aspects, particularly focusing on the principles of detection.

Notable works include the work of Fadlullah et al. in [19] presenting a novel approach – Detection and TRAcBack (DTRAB), to counter attacks on encrypted protocols by utilizing traffic-feature analysis. The paper focused on developing an anomaly-based IDS using the non-parametric cumulative sum (CUMSUM) algorithm for strategically distributed monitoring stubs. DTRAB achieved a 100% detection rate in detecting individual attacks when the number of flows remained below 7, but its performance was observed to degrade as the number of flows increased, highlighting a limitation in scalability. Nadiammai and Hemalatha [20] explored the application of data mining algorithms to intrusion detection databases, comparing the performance of various traditional rule and function-based classifiers. The results indicated that the sequential minimal optimization (SMO) classification algorithm outperformed others in terms of accuracy, specificity, and sensitivity, reaching an accuracy of 97.78% with a detection time of 2.56 seconds. However, the focus on traditional ML techniques may limit adaptability to emerging attack vectors and the dynamic nature of today’s network ecosystem. Gogoi et al. [21] presented a multi-level hybrid intrusion detection system (MLH-IDS) that combined supervised, unsupervised, and outlier-based techniques to improve detection efficiency for both existing and zero-day attacks. The performance evaluation of MLH-IDS demonstrated a detection rate ranging from 81.43% for U2R attacks to 99.99% for DoS attacks. However, the performance of MLH-IDS is contingent on the quality of input features; if the feature selection process is not optimized, it may lead to suboptimal results. Bang et al. [22] utilized LTE signaling to extract spatial and temporal characteristics of traffic data in wireless sensor networks. They employed the semi-Markov model to detect potential attacks. Their approach was successful in differentiating attack nodes and maintaining a very low rate of false positives. Nonetheless, this reliance on LTE signaling limits the applicability of their method to environments where such infrastructure is available, potentially excluding many scenarios in diverse network topologies. Danish et al. [12] introduced a novel

LoRaWAN-based Intrusion Detection System (LIDS) for jamming attacks. They developed a real experimental testbed, and trained LIDS on real join request data utilizing two statistical algorithms: Kullback Leibler Divergence (KLD) and Hamming distance (HD). Their performance evaluations based on Receiver Operating Characteristic (ROC) produced high detection rates, with KLD achieving up to 98% and HD up to 88%, both with a 5% false positive rate. However, the reliance on statistical algorithms can lead to challenges in generalizing results across diverse network conditions and attack scenarios. Liu [23] proposed a unique approach for features extraction and encoding from NetFlow data into NetFlow images using feature correlation analysis and surrounding correlation (SC) matrix. These NetFlow images were then utilized as input to the 3-layer CNN and ResNet18 models, resulting in an accuracy of 93% and 95.86%, respectively. While this method showcases the potential of using image representations for network data, it primarily focuses on static feature extraction, which may not adequately address the dynamic nature of evolving cyber threats. Souza et al. [24] proposed a hybrid IDS approach using a Fuzzy Neural Network. The IDS operated as a host intrusion detection system and underwent malware detection tests on three publicly available malware pattern datasets. Their simulation results demonstrated a striking accuracy of 99.02% with a detection time of 4.03 seconds. However, this approach may face challenges in scalability and adaptability in real-time scenarios where rapid response is crucial.

We now shift our focus to Intrusion Prevention Systems (IPS), exploring the research efforts dedicated to developing and evaluating methodologies that not only detect unauthorized activities but also incorporate preventive measures. Zoppi et al. [25] proposed an unsupervised anomaly detection approach using sliding windows for monitoring dynamic IoT systems. The study explored the limitations and drawbacks of anomaly detection in systems where the expected behavior changes over time and conducted a quantitative analysis using Statistical Predictor and Safety Margin (SPSM) algorithm to compute adaptive bounds for anomalous activities as well as K-Means and Histogram-based Outlier Score (HBOS). While their scores were not as favorable, the methodology employed in this research holds promise for future investigations in the field. Moreover, the gap lies in their reliance on unsupervised techniques, which can struggle with high false positive rates in highly dynamic environments. Krishna et al. [26] proposed the utilization of deep learning-based Multi-Layer Perceptron architecture for immediate classification of DoS, Probe, R2L, and U2R attacks which resulted in an accuracy of 91.41%. Additionally, they implemented a prevention phase using a background script and IPtable that make decisions based on the classification results. Despite these advancements, the approach primarily focuses on immediate classification without adequately addressing the evolving nature of attack vectors, which may affect long-term effectiveness. Furthermore, the reliance on a background script for prevention may introduce latency and is limited

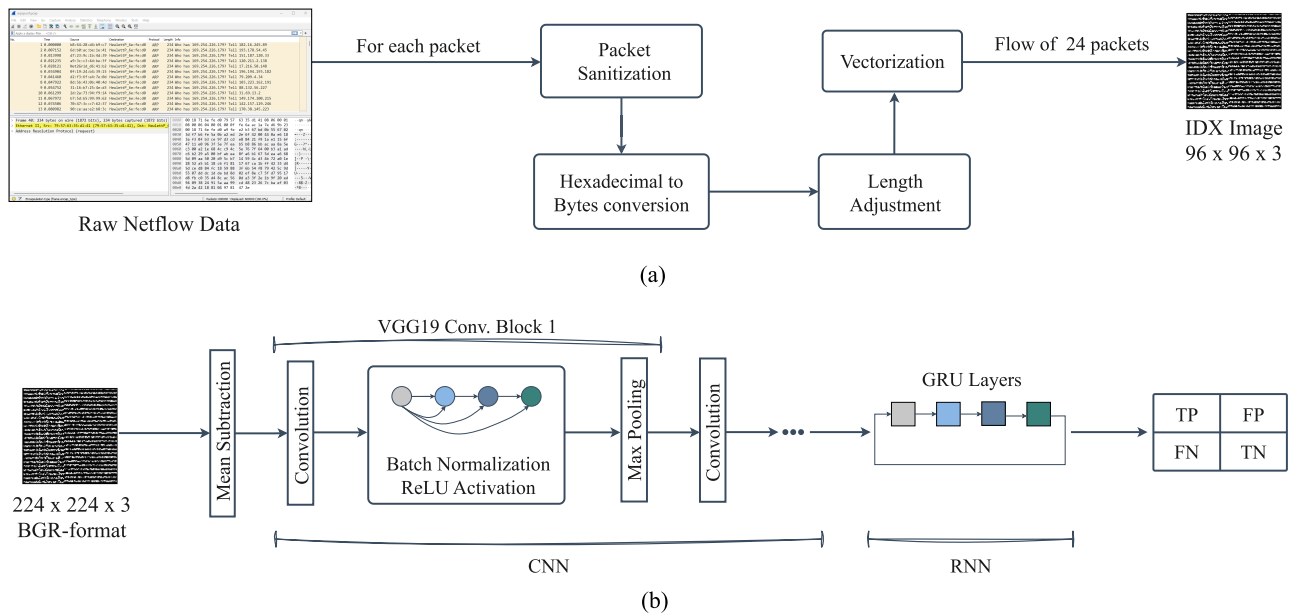


FIGURE 1. Overall Workflow (a) Preprocessing Pipeline, (b) Intrusion Detection Schematic.

in its ability to adapt to new threats in real time. Later, in 2021, a game theory-based preventive approach was proposed by Govindaraj et al. [27] to mitigate DDoS attacks and ensure network stability. They modeled a system as a 2-player Bayesian signaling zero sum game, where the Nash Equilibrium represented optimal strategies for both the attacker and the system. The results of the study, simulated on ns3 network simulator, demonstrated an 80% detection rate, 90% prevention rate, and a false positive rate of 6%. While this approach provides a novel perspective on DDoS mitigation, it primarily focuses on a theoretical model, as well the reliance on game theory can introduce computational overhead. Baldini and Amerini [28] presented a real-time algorithm based on a sliding window with the application of Morphological Fractal Dimension (MFD) for detecting DDoS attacks. They aimed to address the computational cost associated with existing IDS and explored the use of MFD. Their results demonstrated a significant improvement in DDoS attack detection compared to entropy-based methods. In addition, they introduced a novel algorithm for automatically defining the sliding window size that achieved a high detection accuracy of 99.30%, outperforming similar approaches on the same dataset, while maintaining a fast execution time of 34ms. However, while MFD is effective for certain patterns of DDoS attacks, it may not capture the full spectrum of network behavior associated with different types of intrusions. Kumar et al. [29] proposed the implementation of a Network-based Intrusion Prevention System (NBIPS), utilizing the existing Snort framework, to protect cloud servers and IoT systems from unauthorized access. The proposed NBIPS, positioned behind a firewall, acts as a defense mechanism by inspecting network activity

streams, identifying misuse instances, and taking preventive measures. Their system, installed as either an inline or passive model, monitors and blocks traffic using signature-based protocols, thereby preventing attacks on IoT systems. Their results indicate communication cost minimization up to 384 bits. While the NBIPS demonstrates effectiveness in preventing known attacks through signature-based detection, it may struggle to address emerging threats that do not match existing signatures. This reliance on pre-defined signatures can leave the system vulnerable to zero-day attacks. Patil et al. [30] developed an architecture that combines the Snort IDS/IPS with machine learning to create a robust and secure system for detecting and preventing attacks in domestic networks. Their system, named JARVIS, utilized Random Forest models to detect malicious patterns in real-time network traffic data. By dynamically updating Snort rules based on the model's suggestions, they achieved resource consumption optimization and improved the system's efficiency. The results showed an accuracy of 97% in detecting incoming attacks and suggesting deployable rules through the web interface, enabling effective prevention of further damage. While JARVIS effectively enhances detection rates, its utilization of Random Forest models may limit its capacity to handle adaptive attacks that do not conform to historical patterns. Yungaicela-Naula et al. [31] proposed a novel approach using Software-Defined Networking (SDN) to automate detection and mitigation of slow-rate DDoS attacks. By leveraging SDN's centralized control and programmability, their system was configured to conduct state assessment and to implement blocking mechanisms to prevent DDoS attacks. The scheme's effectiveness was demonstrated through experiments using the live generated

SDN-SlowRate-DDoS dataset, achieving accuracy ranging from 91.66% to 100%. Nonetheless, their dependency on a centralized architecture raises concerns about potential single points of failure, which could be exploited by attackers to disrupt the detection and mitigation processes.

We found that most previous research employed shallow classification models, yielding excellent results when the feature dimension is small. However, in scenarios with large datasets and large feature dimensions, the classification performance is compromised. Furthermore, the existing landscape of intrusion prevention faces several challenges, such as high-speed network traffic, complex rule-based approaches, and a significant number of false alarms. To address these gaps, this paper proposes a flexible network intrusions mitigation solution that, in particular, incorporates live data capture in a realistic network setting, eliminating the need for prior feature engineering, and ensuring no incoming packet information is discarded. By prioritizing dynamic implementation, our proposed scheme leverages a two-fold detection approach via image-based deep nested classifiers. We establish real-time evaluation and deployment support utilizing adaptive sliding windows for efficient handling of network traffic considering the real-world network constraints. We resolve to define a time-efficient adaptive method that relies simply on flow statistics and channel parameters to determine the optimal window size.

III. METHODOLOGY

This section is structured into two subsections. The initial subsections provide an extensive overview of our proposed approach highlighting the testbed and IDS design. The detailed workflow, including preprocessing stage and IDS design, is illustrated in Figure 1. The final subsection focuses on the practical implementation of the proposed model within a dynamic network environment, specifically incorporating adaptive sliding windows.

A. TESTBED SETUP AND LOCAL AREA NETWORK CONFIGURATION

A testbed is elemental in facilitating extensive experiments and evaluations in the field of network testing and analysis. The objective is to establish a setting that emulates

real-world networking scenarios while enabling controlled experimentation. In this research work, we design a comprehensive testbed configuration that comprises a) four personal computers (PCs) and b) twelve PCs to be used as bots, along with a Fast Ethernet Switch, a Raspberry Pi module, and a mobile workstation functioning as a Command & Control (C&C) server. The PCs are strategically connected to the Ethernet Switch via Cat 6 Ethernet cables to ensure seamless communication and dependable data transmission. This connection configuration ensures that all bots are within the same local network connectivity, facilitating network traffic flow as validated by Figure 2. For the performance evaluation of the proposed scheme we have opted for a testbed implementation, which better demonstrates the amenability for real deployment and in addition offers to the research community a new data set for further exploitation.

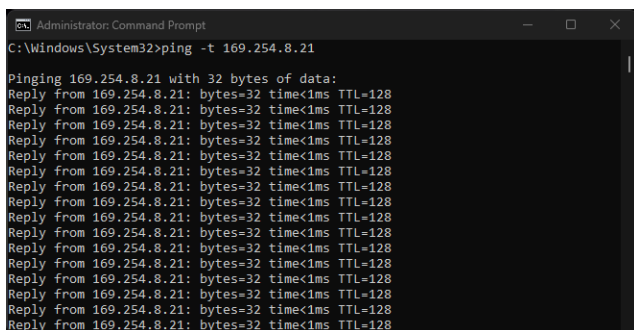
However, it is important to acknowledge that practical scenarios might involve bots located in different network environments or distributed geographically, potentially encountering network congestion. The testbed operations are largely dependent on the C&C server, which plays a pivotal role in orchestrating network attacks. To address concerns regarding high availability and potential single points of failure, we have implemented failover clustering and data replication mechanisms in the C&C server's configuration. Failover clustering allows us to maintain a standby C&C server that is continuously synchronized with the primary server. In the event of a failure, the standby server can quickly take over operations with minimal downtime. Additionally, data replication ensures that all operational data is continuously updated between the primary and secondary servers, minimizing the risk of data loss during a failover. This enhancement not only improves the resilience of the system but also ensures continuous operation in real-world scenarios.

Furthermore, the integration of the Raspberry Pi device with the central switch provides the advantage of real-time capture of network traffic data, further enhancing the effectiveness of our testbed in simulating various network conditions.

B. MALICIOUS TRAFFIC GENERATION AND CAPTURE

Once the testbed is configured, the next stage is to generate and capture malicious network traffic. We designed a systematic scheme to launch network attacks from the C&C server and subsequently capture the generated malicious traffic using Wireshark, installed on the Pi module. A set of nine distinct attacks, namely: Ping of Death, ARP Spoofing, SYN Flood, UDP Flood, HTTP Flood, Rudy, TCP SYN Scan, TCP Connect Scan, UDP Scan, are employed to simulate various forms of network intrusions.

It is important to note that we simulated each attack in bursts of 12 packets without presetting the inter-arrival time between requests to create a manageable yet impactful test scenario, except for the Rudy attack, which featured headers sent with an inter-arrival time of 5 seconds between



```
Administrator: Command Prompt
C:\Windows\System32\ping -t 169.254.8.21

Pinging 169.254.8.21 with 32 bytes of data:
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
Reply from 169.254.8.21: bytes=32 time<1ms TTL=128
```

FIGURE 2. Ping test: Network accessible.

Algorithm 1 Flow-to-Image Conversion

Input: PCAP files
Output: IDX format images

- 1: **Step 1:** Get flows of 24 packets each
- 2: **for** each PCAP **do**
- 3: Initialize lists: $pkts = [], flows = []$
- 4: Read Ethernet packets
- 5: **for** each packet **do**
- 6: *Packet Sanitization:* Mask src & dst IP and MAC address
- 7: *Hex-to-Byte:* Convert hexadecimal packet information into byte sequence
- 8: **if** packet size < 384 bytes **then**
- 9: *Zero-pad* the packet to adjust length
- 10: **else if** packet size > 384 bytes **then**
- 11: *Truncate* the packet to adjust length
- 12: **end if**
- 13: $pkts \leftarrow$ packet
- 14: **end for**
- 15: **for** i from 0 to $len(pkts) - 1$ with step size 24 **do**
- 16: $flows \leftarrow$ packets in $pkts$ from index i to $i + 24$
- 17: **end for**
- 18: **for** each flow **in** $flows$ **do**
- 19: **if** flow size < 24 **then**
- 20: Pad flow with $[bytes([0]) \times 384]$ repeated (24–flow size) times
- 21: **end if**
- 22: *Vectorize:* Reshape flow to 384×24 -dimensional array
- 23: **end for**
- 24: **end for**
- 25: **Step 2:** Transform each flow into IDX image
- 26: **for** each flow **in** $flows$ **do**
- 27: Write IDX header
- 28: Write 9, 216-dimensional flow to IDX data field
- 29: **end for**

slow HTTP requests. The choice to send 12×4 packets or 12×12 packets in quick succession from both the four-bot and twelve-bot configurations, with no intentional delay, aimed to overwhelm the target and quickly saturate the traffic levels typically observed in real botnet attacks. This approach allows us to evaluate how rapidly and effectively our intrusion detection system can respond to resource exhaustion tactics, providing insights into its performance under stress.

C. DATA PREPROCESSING

Preprocessing is an essential aspect in the preparation of network traffic data for subsequent analysis and classification. The preprocessing stage involves multiple steps that aim to transform the raw network traffic data into IDX format image representation. Unlike conventional methods that use feature engineering, our approach preserves all the inherent feature information present in each packet. This eliminates the need for filtering or manual extraction of specific traffic features. Our preprocessing pipeline, outlined in Algorithm 1, begins with packet sanitization wherein we anonymize the sensitive information fields (such as,

the source and destination MAC and IP addresses) in the Data Link and Internet Protocol Layers. Upon examining the network traffic packets in Wireshark (as shown by the sample in Figure 1a), it could be observed that the information is reserved in hexadecimal codes. We convert these hexadecimal streams into bytes sequences. Next, we adjust the packet size and extract 384 bytes including the Application Layer (i.e., payload information) from each packet to ensure uniformity and maintain a balance between performance and computational efficiency. After certain trails, we have come to the conclusion that this selection allows us to capture essential features of the traffic while minimizing the processing load on our system. To achieve this, we zero-pad the packets less than 384 bytes and truncate those longer than 384 bytes, thereby standardizing the input. Finally, we combine a set of 24 packets as one flow and vectorize the bytes sequences into a $384 \times 24 = 9,216$ -dimensional feature vector. This approach facilitates efficient processing and classification while still capturing significant data characteristics. The next step is to generate 2D IDX images from bytes-type feature vectors. The IDX file format has the capacity to represent multidimensional

numerical arrays, thus rendering it suitable for storing image data. Finally, we generate IDX header and pack the traffic data in IDX format to be used for training. It is important to mention that prior to training, the feature vectors are reshaped into 96×96 8-bit unsigned integer arrays.

D. INTRUSION DETECTION PIPELINE

We develop a nested classification scheme integrating Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The convolution operation of CNNs stands out in spatial perception and image recognition tasks. In network environments, traffic packets often fragment during transmission, whereby these fragments look like IP packets. The IP field of each packet carries significant spatial characteristics related to the flow. Therefore, recognizing the importance of spatial attributes, the first block of our nested classifier employs the CNN architecture to extract spatial features. Further, the second layer of our classifier is based on the RNN architecture to extract temporal features of the flow. The recurrent connections of RNN are notable in capturing temporal dependencies in sequential data. In the context of network traffic, the flow of traffic packets adheres to a particular chronological order, while their arrival at the receiving end is influenced by delays, giving rise to a distinct sequence. At the same time, the number of traffic packets transmitted within a specified timeframe displays variation, and these attributes together indicate the existence of temporal characteristics. The details of our convolution and recurrence stacks are summarized below.

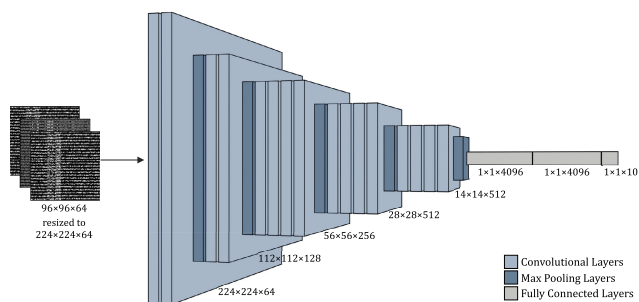


FIGURE 3. VGG19 Architecture.

1) CNN MODEL

We utilize a pretrained VGG19 model as our CNN-based spatial feature extractor. The VGG19 model, illustrated in Figure 3, is composed of a series of convolutional layers, followed by max-pooling layers. This architecture is renowned for its depth, consisting of 19 weight layers, hence the name VGG19. The convolutional layers use small receptive fields (3×3) with a stride of 1, and padding is applied to maintain spatial resolution. This design choice enables the model to learn complex spatial features while keeping the number of parameters manageable. After each convolutional layer in VGG19, a rectified linear unit (ReLU) activation function is applied. This activation function, defined in Eq. 1, introduces

non-linearity into the network, enhancing its ability to capture complex patterns. The convolution operation itself involves sliding 3×3 filters, ω , over the input feature map, denoted as x , and performing element-wise multiplication. The bias term, b , is then added, and the resulting values are passed through the activation function, which yields the output of the convolutional layer, as given in Eq. 2.

$$f(x) = \max(0, x) \tag{1}$$

$$y = f(\omega * x + b) \tag{2}$$

Subsequently, max-pooling layers are inserted periodically within the VGG19 architecture to reduce the spatial dimensions of the feature maps while preserving important information. Max-pooling involves dividing the input feature map into non-overlapping regions and selecting the maximum value within each region. This downsampling operation aids in capturing translation invariance and reduces the computational load. In summary, the model’s output functions as a feature map, which encodes the high-level spatial information obtained from the NetFlow images.

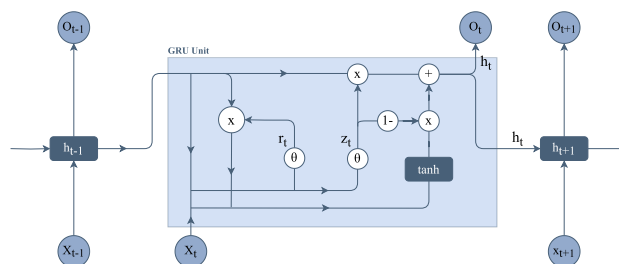


FIGURE 4. GRU Architecture.

2) RNN MODEL

We utilize the Gated Recurrent Unit (GRU) architecture as the RNN-based temporal feature extractor due to its inherent computational efficiency compared to other RNN configurations like LSTMs. This is crucial for our application, as accelerated processing of network traffic is essential for effective intrusion detection. The GRU cell, depicted in Figure 4, consists of multiple recurrent layers, each comprising a series of recurrent neurons. The primary concept underlying the GRU architecture is the integration of gating mechanisms, which enable selective retention or updating of information within the hidden state. This would allow our model to dynamically adapt to evolving network traffic patterns, making it more robust and resilient to novel attacks in the long term. The gating mechanisms in the GRU control the flow of information within the recurrent layers by utilizing an update gate and a reset gate, as given by Eq. 3 and 4, respectively.

$$z_t = \sigma(W_z \odot [h_{t-1}, x_t] + b_z) \tag{3}$$

$$r_t = \sigma(W_r \odot [h_{t-1}, x_t] + b_r) \tag{4}$$

Here, x_t represents the input at time step t , h_{t-1} denotes the previous hidden state, W denotes the weight matrices,

and b denotes the bias vectors. The sigmoid function (σ) is used to evaluate the gates, and \odot signifies element-wise multiplication. The hidden state, as defined in Eq. 5, is updated at each iteration based on the outputs of the gates and the input. The inclusion of the hyperbolic tangent function (\tanh) introduces non-linearity, enabling the GRU model to capture complex relationships within the data.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \quad (5)$$

In the context of our nested classifier, the GRU model receives feature maps extracted by the VGG19 network. By sequentially processing these feature maps through the recurrent layers, the GRU model effectively captures temporal patterns in the network traffic data. This capability allows the model to detect abnormalities and make informed decisions based on the temporal characteristics of the traffic.

E. INTRUSION PREVENTION FRAMEWORK

Intrusion prevention is a critical aspect of network security, aiming to mitigate potential threats before they compromise a system or network. It extends the implementation of an IDS without sanitizing the packets so as to actively prevent unauthorized access, attacks, and malicious activities that could jeopardize the confidentiality, integrity, and availability (CIA) of network resources. Our proposed solution for real-time intrusion prevention, highlighting key components and effective safeguarding strategies, is presented next.

1) ADAPTIVE SLIDING WINDOWS APPROACH

We propose an adaptive sliding window approach that accommodates two types of windows: content-based and time-based. Content-based windows are utilized when the traffic volume exceeds a predefined threshold and focus on smaller subsets of traffic data, facilitating real-time analysis and prevention of attacks based on packet content. Time-based windows capture a larger time span of network traffic data, enabling identification of anomalies or malicious patterns over a defined period. The decision to employ a content-based or time-based window is dynamically taken based on the rate of incoming traffic and the volume of data. If the traffic rate exceeds a threshold, the system automatically transitions to content-based windows, ensuring timely analysis. Conversely, if the traffic rate falls below the threshold, time-based windows are employed, providing a broader context for packet inspection. This dynamic window selection optimizes resource utilization and enhances the effectiveness of the Intrusion Prevention System (IPS).

Within each sliding window, the proposed intrusion prevention solution, placed right after the firewall, incorporates the intrusion detection system previously trained on a large database of benign and malicious traffic, and makes decisions in real-time as illustrated in Figure 5.

However, in a real-world implementation of an IPS using adaptive sliding windows, network bandwidth limitations,

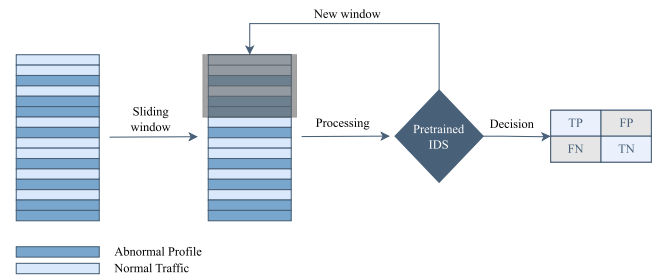


FIGURE 5. IPS workflow.

computational resource availability, and latency concerns need to be addressed. For this purpose, we employ quality of service (QoS) mechanisms to prioritize traffic and allocate sufficient bandwidth to the IPS. The underlying detection algorithms and processing workflows are ensured to be flexible, minimizing latency and ensuring real-time analysis. Computational resources, including CPU, memory, and buffer storage, are allocated adequately based on the expected traffic volume. By implementing these strategies, our proposed IPS solution operates optimally within our testbed environment, providing robust intrusion prevention.

2) CONTINUOUS TRAFFIC MONITORING & ALERTING

The proposed solution incorporates a comprehensive continuous monitoring and alerting feature, which plays a crucial role in facilitating real-time threat detection and ensuring timely response to potential security breaches. Initially, all incoming traffic is processed by our pretrained intrusion detection system. When the IDS detects any malicious activity, it promptly triggers an alert by initiating a notification process. We define an alerting function that establishes a connection to the Windows Event Log and creates a custom event entry. Upon detecting abnormality in the incoming traffic profile, it generates an alert with the specified event type, event ID, message, and source. The alert is meticulously crafted to deliver precise and detailed information about the detected intrusion, including the source IP, destination IP, and attack type.

Furthermore, all alerts are logged for future reference, providing a historical record that can be reviewed during incident investigations or for compliance purposes. In the future, we plan to leverage this logged data to develop a reinforcement learning framework aimed at enhancing our current intrusion detection and prevention pipeline.

IV. EXPERIMENTATION AND EVALUATION

The section presents experimental results of our proposed intrusion detection solution using machine learning-based, (i) binary classification, and (ii) multi-class classification of the network traffic.

A. IMPLEMENTATION DETAILS

We train our classifiers on a compute engine consisting of an Intel Xeon CPU operating at 2.20 GHz and equipped with 2 vCPUs and 13 GB of RAM, alongside an NVIDIA T4 GPU with 16 GB of VRAM. We begin by simulating botnet traffic using the testbed described in section III-A and then transform the traffic flows into images as per the pre-processing pipeline described in section III-C. These images are then fed into the nested classifier, which seamlessly integrates the robustness of a pretrained VGG19 model with the dynamic adaptability of a GRU architecture. Detailed analysis regarding the classifier's training performance is presented below.

1) MULTI-CLASS CLASSIFICATION

For multiclass classification, we use our live captured traffic dataset comprising nine distinct network attacks and one normal traffic profile, resulting in a total of 10 classes. The training parameters include the Mean Squared Error (MSE) as the loss function, the Root Mean Squared Propagation (RMSprop) as the optimization algorithm, and a learning rate of 0.001 at a momentum of 0.5. The training is conducted on a total of 0.175M images, organized into batches of 64, and iterated over 30 epochs. With each epoch consisting of 2734 iterations, the training phase continues for 28.57 hours and concludes with an impressive training accuracy of 99.8276% and a minimal MSE loss of 0.0841. The learning curves, depicted in Figure 6, illustrate the steady convergence of the training process. Notably, the validation process closely aligns with the training curve, reaching a peak accuracy of 99.6324%. With the highest validation loss of 5.9893 during the first epoch, the classifier manages to minimize it up to 0.0324 at the end of 30th epoch. These findings suggest the absence of overfitting and highlight the classifier's generalizability. It is worth mentioning that these results are achieved by resizing the input images from their original dimensions of $96 \times 96 \times 3$ to conform to the input size requirement of the VGG19 model i.e., $224 \times 224 \times 3$.

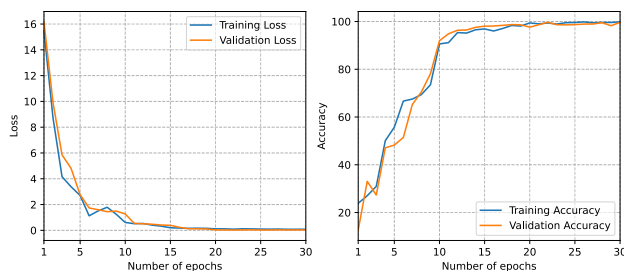


FIGURE 6. Learning curves for multiclass classification.

2) BINARY CLASSIFICATION

We also investigate the performance of our proposed detection model from a simpler perspective by reducing the problem into a binary decision, distinguishing between

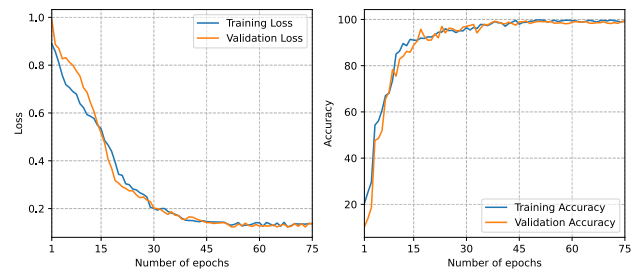


FIGURE 7. Learning curves for binary classification.

network attacks and normal traffic. We partition our live captured dataset into two distinct classes: one representing network attacks and the other representing normal traffic. The parameters defined for the binary training process include the Binary Cross Entropy (BCE) as the loss function, the Adam optimization algorithm, and a learning rate of 0.003. A total of 70k images undergo training, organized into batches of 48, and iterated over 75 epochs. Each epoch comprises 1459 iterations, allowing the classifier to learn effectively from the dataset and generate accurate predictions. The training phase extends over a duration of 21.86 hours, yielding a remarkable training accuracy of 99.3750% and a negligible binary cross entropy (BCE) of 0.1369. The learning curves showing the model's performance in terms of accuracy and loss, illustrated in Figure 7, demonstrate the consistent convergence of the training process similar to the multiclass classifier. Additionally, the validation curves closely parallel the trajectory of the training curve, achieving a peak accuracy of 99.0623% and a minimal entropy loss of 0.1357. These insights indicate the absence of overfitting and emphasize the classifier's ability to generalize effectively to unseen data.

B. TESTING AND EVALUATION

We evaluate the multi-class classifier on 37.5k unseen images with each class in the test set consisting of a total of 3750 images. The model exhibits very good performance, with an evaluation time as low as 0.17ms per image and an accuracy of 99.5982%. Similarly, we evaluate the binary classifier on 15k unseen images with each class in the test set consisting of a total of 7500 images. The binary classification model also exhibits very good performance, with an evaluation time as low as 0.041ms per image and an accuracy of 99.6145%.

Additionally, we evaluate the performance of both classifiers on the basis of four key indicators: True Positives (TP) representing correctly identified traffic flows belonging to the specific class, True Negatives (TN) indicating correctly identified traffic flows that do not belong to the class, False Positives (FP) corresponding to the traffic flows incorrectly classified as belonging to a class, and False Negatives (FN) representing the traffic flows incorrectly classified as not belonging to the class. Based on these indicators, we leverage

TABLE 1. Performance metrics for multiclass classification.

Traffic Profile	FPR	FNR	Precision	Recall	F1-score
Benign	0.09717	0.00815	0.98132	0.99161	0.98644
ARP Spoof	0.09586	0.00275	0.96947	0.99713	0.98311
HTTP Flood	0.09847	0.01307	0.97992	0.98673	0.98332
Ping of Death	0.09914	0.02846	0.95597	0.97113	0.96358
RuDY	0.0989	0.00773	0.96646	0.99219	0.97915
SYN Flood	0.09942	0.02826	0.99129	0.97158	0.98134
SYN Scan	0.09927	0.02162	0.96247	0.97823	0.97028
TCP Connect Scan	0.09945	0.01688	0.97973	0.98303	0.98137
TCP SYN Scan	0.09947	0.04207	0.93206	0.95771	0.94471
UDP Flood	0.09976	0.0056	0.99974	0.99438	0.99705
Macro Average	0.07592	0.01343	0.97189	0.98240	0.97707

TABLE 2. Performance metrics for binary classification.

Traffic Profile	FPR	FNR	Precision	Recall	F1-score
Benign	0.05587	0.00992	0.94698	0.99008	0.96805
Malware	0.05974	0.00887	0.95051	0.99113	0.97039
Macro Average	0.057805	0.009395	0.948745	0.990605	0.96922

the measures of False Positive Rate, False Negatives Rate (or Specificity), Precision, Recall (or Sensitivity), and F1-score (calculated using Eq. 6–10, respectively) to comprehensively benchmark our IDS. We aim to minimize FPs and FNs to achieve accurate intrusion detection while reducing false alarms and missed intrusions. The performance evaluation metrics, summarized in Table 1 and 2, provide an inclusive and quantitative assessment of the model's generalizability across multi-class and binary classification scenarios.

$$FPR = \frac{FP}{FP + TN} \quad (6)$$

$$FNR = \frac{FN}{FN + TP} \quad (7)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (8)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (9)$$

$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)$$

1) PERFORMANCE BENCHMARKING AGAINST STATE-OF-THE-ART

Further, we compare our model's performance by benchmarking it against some state-of-the-art methods for NIDS that use machine learning or deep learning, such as CNN + ResNet18 [23], MLP [26], CNN [32], SVM [33], ViT [34], DBN [35], DBN + KELM [35], LeNet-5 + LSTM [36], and SAE + SVM [37]. Each classifier is trained on the raw net-flow data for multi-class classification and evaluated in terms of accuracy, precision, recall, false positives rate, and evaluation time per packet. The results are listed in Table 3. It can be observed that our method using VGG19 and GRU architecture for two-fold spatial and temporal

feature extraction outperforms all the other methods across all metrics examined.

Moreover, to validate the generalizability of our detection model, we conduct additional evaluations using one of the widely popular open-source datasets, CIC-IDS2018 [38]. The results obtained from this evaluation further reinforce the efficacy of our approach, demonstrating promising performance on a diverse set of network attacks. On top of that, we also compare our results with state-of-the-art methods, documented in Table 3. It could be observed that our classifier demonstrates superior performance over all the other methods in terms of precision, recall, false positives rate, and evaluation time per packet, except for the accuracy which falls short by 0.00312% compared to SVM's accuracy.

2) PACKET VOLUME VS. PERFORMANCE TRADE-OFFS

After conducting various preliminary experiments, we identified that both packet size and flow size significantly affect computational efficiency and the detection rates of our hybrid classification. We determined an optimal packet size of 384 bytes and a count of 24 packets per flow based on empirical trials, which demonstrated that this volume effectively captures essential traffic features while balancing computational efficiency and detection performance. We observed that smaller packet sizes tend to miss critical contextual information needed for accurate detection, while excessively large packet or flow sizes can lead to increased processing times without substantial gains in performance. Table 4 and 5 provide a comprehensive overview of the trade-offs associated with different packet sizes and flow sizes, highlighting how these parameters influence the size of feature vector, accuracy, precision, recall, and runtime. These insights not only contribute to the understanding of

TABLE 3. Performance analysis of state-of-the-art ML algorithms.

Model	Accuracy	Precision	Recall	FPR	Runtime (ms/pkt)
Our Net-Flow Dataset					
CNN + ResNet18 [23]	0.84347	0.84981	0.80076	0.29137	3324.523
MLP [26]	0.77781	0.78152	0.68009	0.60654	10948.116
CNN [32]	0.81891	0.83515	0.72667	0.30415	10094.63
SVM [33]	0.97621	0.73626	0.61033	0.63479	310.8400
ViT [34]	0.95842	0.93032	0.90138	0.16415	121.9200
DBN [35]	0.82145	0.74951	0.80231	0.43177	28743.99
DBN + KELM [35]	0.90137	0.72364	0.84209	0.40214	30214.17
LeNet-5 + LSTM [36]	0.91891	0.95314	0.87841	0.19587	343.4101
SAE + SVM [37]	0.80145	0.76745	0.75555	0.66714	1043.501
VGG-19 + GRU (Ours)	0.99614	0.97189	0.98240	0.07592	0.001710
CIC-IDS2018 [38]					
CNN + ResNet18 [23]	0.89002	0.87479	0.85049	0.31913	2924.713
MLP [26]	0.77041	0.79104	0.76077	0.57156	10078.109
CNN [32]	0.83014	0.83723	0.77704	0.28739	9278.990
SVM [33]	0.98634	0.71230	0.66022	0.67021	298.8000
ViT [34]	0.97719	0.95707	0.95358	0.15529	207.0600
DBN [35]	0.82662	0.76856	0.81577	0.42581	29454.15
DBN + KELM [35]	0.91093	0.79003	0.81085	0.41133	29089.37
LeNet-5 + LSTM [36]	0.97052	0.97844	0.94251	0.22587	414.1978
SAE + SVM [37]	0.82417	0.75514	0.75049	0.69544	998.0071
VGG-19 + GRU (Ours)	0.98322	0.98096	0.97346	0.06199	0.006170

TABLE 4. Trade-offs in flow size and performance with a fixed packet size of 384 bytes.

Flow Size (pkts)	Feature Vector Size (bytes)	Accuracy (%)	Precision (%)	Recall (%)	Runtime (ms/pkt)
8	3,072	0.78819	0.83551	0.71571	0.00086
12	4,608	0.83245	0.73579	0.80444	0.00119
24	9,216	0.99614	0.97189	0.98240	0.00170
48	18,432	0.92436	0.99760	0.79914	1.88401
96	36,864	0.50117	0.49981	0.49026	9.96130

TABLE 5. Trade-offs in packet size and performance with a fixed flow size of 24 pkts.

Packet Size (bytes)	Feature Vector Size (bytes)	Accuracy (%)	Precision (%)	Recall (%)	Runtime (ms/pkt)
128	3,072	0.77289	0.83412	0.74256	0.00088
256	6,144	0.91321	0.87134	0.89567	0.00159
384	9,216	0.99614	0.97189	0.98240	0.00170
512	12,288	0.98782	0.96175	0.94832	1.02917
1024	24,576	0.51847	0.43289	0.48673	6.75910

impact of packet volume on performance but also guide future designs of intrusion detection systems.

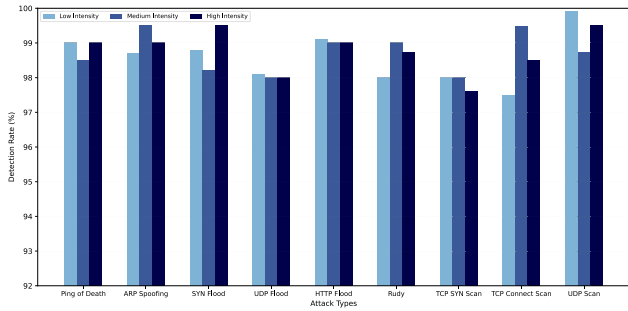
C. LIVE INTRUSION PREVENTION EVALUATION

Once we have the trained model, we evaluate it in real-time as an intrusion prevention system. First, we optimize various network properties to enhance the effectiveness and efficiency of the system. We allocate a substantial buffer capacity within the network infrastructure, enabling efficient handling and processing of incoming packets. This larger buffer size allow us to minimize packet loss and ensure comprehensive analysis for intrusion detection purposes. Additionally, we ensure sufficient bandwidth availability,

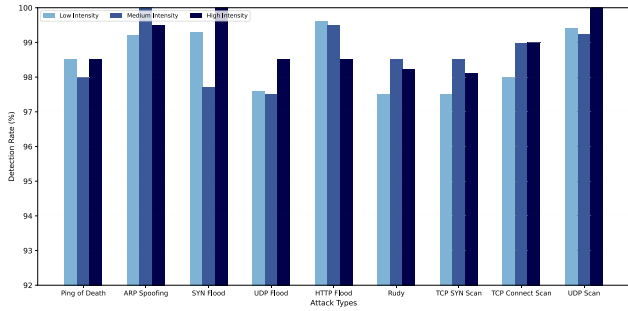
enabling the timely transmission of packets to the intrusion detection system. This facilitates real-time analysis, allowing for quick decisions. We conducted extensive performance evaluations under varying conditions:

1) ATTACK LOAD VARIATIONS

To assess the robustness of our intrusion prevention system, we tested it under different attack loads categorized as low, medium, and high intensities, with low intensity defined as a sustained rate of 12–24 packets per second (pps) simulating light network traffic, medium intensity involving an increase to 24–64 pps representing a more aggressive attack environment, and high intensity simulating extreme conditions with



(a) Four-Bot Attack



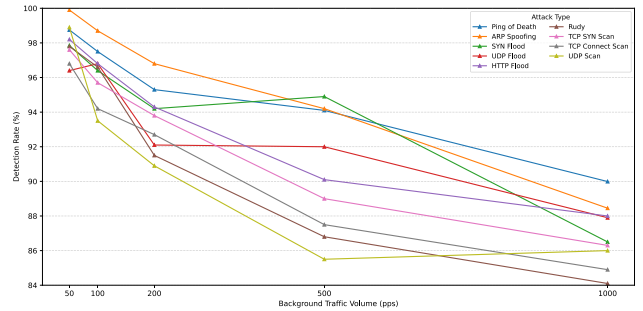
(b) Twelve-Bot Attack

FIGURE 8. Detection rates under different attack loads.

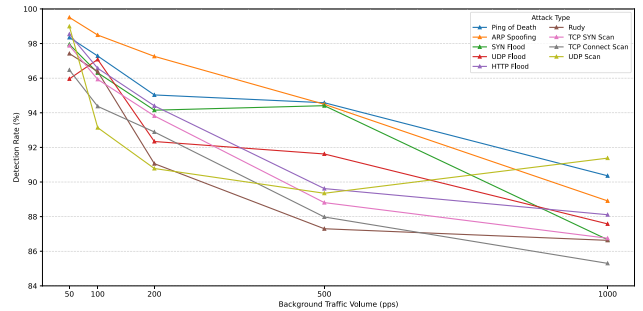
rates exceeding 100 pps. The results, presented in Figure 8, illustrate the detection rates (DR) achieved by our model under these varying attack intensities for both the four-bot and twelve-bot scenarios. While some fluctuations in DRs can be observed, the rates remained consistently high across all attack types. Notably, the lowest DR is recorded at 97.5% in case of medium intensity UDP flood attack from 12-bot configuration, indicating that even under minimal attack conditions, our proposed system effectively identified the intrusion.

2) BACKGROUND TRAFFIC CONDITIONS

As another evaluation dimension, we examined our model's performance under varying background traffic conditions, which encompass the legitimate network traffic that occurs alongside potential botnet activity. To replicate real-world scenarios, we introduced different volumes and types of legitimate traffic, including typical user interactions, web browsing, and application data transfers. This allowed us to analyze and assess how background noise impacts the detection capabilities of our system and the timeliness of alert generation. The results, illustrated in Figure 9, highlight the effectiveness of our approach across various background traffic volumes for both the four-bot and twelve-bot scenarios. Under the four-bot attack and low background traffic conditions (50-200 pps), our model achieved DRs of as high as 99.90% for ARP spoofing attack and as low as 96.40% for UDP flood attack. As we increased the traffic to moderate levels (500 pps), the DR ranged from 86.80% for rudy attack to 94.90% for SYN flood attack. Even



(a) Four-Bot Attack



(b) Twelve-Bot Attack

FIGURE 9. Impact of varying background traffic volume on detection rates.

under high background traffic conditions (1000 pps), our system maintained DRs between 84.93% for TCP connect scan attack and 91.21% for UDP scan attack. These results demonstrate that while the introduction of legitimate traffic introduced some challenges, our model consistently managed to identify threats effectively.

3) REAL-TIME PERFORMANCE AND INCIDENT RESPONSE

We also examined how our adaptive sliding window approach enhances real-time performance and incident response capabilities within the intrusion prevention system. The dynamic switching between content-based and time-based windows is crucial for optimizing detection efficiency. We set a traffic rate threshold of 250 pps to dictate this transition: when incoming traffic exceeds this threshold, the system activates content-based windows for detailed packet analysis. Conversely, when traffic falls below this threshold, time-based windows are employed, providing a broader context for monitoring potential threats. By minimizing background processes leading to latency, we ensure that packets are processed into IDX images and analyzed without significant delays, resulting in near real-time detection of intrusions. These optimizations, including buffer capacity, bandwidth allocation, and latency reduction, collectively contribute to the successful outcomes of our live intrusion prevention. Figure 10 shows snapshot of the IPS operating in real-time. To facilitate effective incident response, we further implement a robust real-time alerting mechanism within our live intrusion detection system. Whenever potential

```

C:\Users\Rimsha Saeed\Desktop>python IPS.py
Benign.....Runtime/pkt: 0.002190666300404211 ms
Benign.....Runtime/pkt: 0.0013577927923003314 ms
ARP Spoof.....Runtime/pkt: 0.002158597245301759 ms
ARP Spoof.....Runtime/pkt: 0.003561623959148314 ms
ARP Spoof.....Runtime/pkt: 0.0042407786545250825 ms
ARP Spoof.....Runtime/pkt: 0.0038274230506773377 ms
ARP Spoof.....Runtime/pkt: 0.0031852102696532226 ms
ARP Spoof.....Runtime/pkt: 0.0014302228564382686 ms
ARP Spoof.....Runtime/pkt: 0.0032406890190863733 ms
ARP Spoof.....Runtime/pkt: 0.00477369860543444 ms
UDP Flood.....Runtime/pkt: 0.002105976694546408 ms
ARP Spoof.....Runtime/pkt: 0.002756112512715844 ms
ARP Spoof.....Runtime/pkt: 0.0024218191876784092 ms
ARP Spoof.....Runtime/pkt: 0.0015796320960633001 ms
ARP Spoof.....Runtime/pkt: 0.003662148793807367 ms
ARP Spoof.....Runtime/pkt: 0.0025925422123320593 ms
ARP Spoof.....Runtime/pkt: 0.0033366394154782535 ms
ARP Spoof.....Runtime/pkt: 0.004235841801564437 ms
ARP Spoof.....Runtime/pkt: 0.004697221382985739 ms
ARP Spoof.....Runtime/pkt: 0.0012126962981866584 ms
ARP Spoof.....Runtime/pkt: 0.00479228990903465 ms

```

FIGURE 10. Real-time incident response.

intrusions are detected, our system promptly generates alerts by establishing a connection to the Windows Event Log.

V. CONCLUSION

In this paper, a network intrusion detection and prevention engine is presented to analyze and classify network traffic data by transforming raw NetFlows into images. We propose a deep nested architecture leveraging a pretrained VGG19 and a GRU network to learn spatial and temporal features of the flow. Our spatial feature extractor, based on the concept of transfer learning, and the temporal feature extractor, utilizing the gating mechanism is significantly better in terms of processing time and resource utilization than other network intrusion detection models. In this paper, we use real-world traffic data captured in a controlled environment employing adaptive sliding windows. The experimental results demonstrate the superior performance of the proposed approach in terms of accuracy, precision, recall and F1-score, accomplished in both binary and multi-class classification scenarios.

In the future, we plan to expand our testbed to include a larger number of bots and to incorporate other security vulnerabilities beyond network intrusions, such as phishing, spyware, and ransomware. As part of this extension, the scalability of the proposed scheme in terms of increasing penetration of IoT devices in real experiments will also be the topic of near future research activities. We will also simulate secured environments to evaluate the impacts of firewalls and data encryption, utilizing current findings as a baseline to establish a solid reference point for evaluating our model's adaptability to these secured environments. Furthermore, we aim at investigating the application of various image processing techniques, such as image filtering, image compression, local binary patterns (LBP), and Fourier transform, to assess the performance of the model trained on manipulated flow images.

REFERENCES

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–26, Dec. 2019.

[2] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, "A survey of CNN-based network intrusion detection," *Appl. Sci.*, vol. 12, no. 16, p. 8162, Aug. 2022.

[3] S. Taheri, M. Salem, and J.-S. Yuan, "Leveraging image representation of network traffic data and transfer learning in botnet detection," *Big Data Cognit. Comput.*, vol. 2, no. 4, p. 37, Nov. 2018.

[4] K. Kizzee. (2023). *Cyber Attack Statistics To Know*. [Online]. Available: <https://parachute.cloud/cyber-attack-statistics-data-and-trends/>

[5] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016.

[6] P. Amini, M. A. Araghizadeh, and R. Azmi, "A survey on botnet: Classification, detection and defense," in *Proc. Int. Electron. Symp. (IES)*, Sep. 2015, pp. 233–238.

[7] A. Shah, S. Clachar, M. Minimair, and D. Cook, "Building multiclass classification baselines for anomaly-based network intrusion detection systems," in *Proc. IEEE 7th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2020, pp. 759–760.

[8] A. Dainotti, F. Gargiulo, L. I. Kuncheva, A. Pescapè, and C. Sansone, "Identification of traffic flows hiding behind TCP port 80," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–6.

[9] I. M. Iqbal and R. A. Calix, "Analysis of a payload-based network intrusion detection system using pattern recognition processors," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, Oct. 2016, pp. 398–403.

[10] W. Ali Aziz, H. Khaliq Qureshi, A. Iqbal, A. Al-Dulaimi, and S. Al-Rubaye, "Towards accurate categorization of network IP traffic using deep packet inspection and machine learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2023, pp. 1–6.

[11] W. Ali Aziz, I. I. Ioannou, M. Lestas, H. Khaliq Qureshi, A. Iqbal, and V. Vassiliou, "Content-aware network traffic prediction framework for quality of service-aware dynamic network resource management," *IEEE Access*, vol. 11, pp. 99716–99733, 2023.

[12] S. M. Danish, A. Nasir, H. K. Qureshi, A. B. Ashfaq, S. Mumtaz, and J. Rodriguez, "Network intrusion detection system for jamming attack in LoRaWAN join procedure," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[13] M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song, and C. Park, "Flow-based malware detection using convolutional neural network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 910–913.

[14] Y. Yu, J. Long, and Z. Cai, "Session-based network intrusion detection using a deep learning architecture," in *Modeling Decisions for Artificial Intelligence*. Cham, Switzerland: Springer, 2017, pp. 144–155.

[15] J. Zhang, I. B. Jemaa, and F. Nashashibi, "Trust management framework for misbehavior detection in collective perception services," in *Proc. 17th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Singapore, Singapore, Dec. 2022, pp. 596–603.

[16] S. Göring, R. R. Ramachandra Rao, R. Merten, and A. Raake, "Analysis of appeal for realistic AI-generated photos," *IEEE Access*, vol. 11, pp. 38999–39012, 2023.

[17] J. P. Anderson. (1980). *Computer Security Threat: Monitoring and Surveillance*. [Online]. Available: <http://archive.org/details/ComputerSecurityThreatMonitoringAndSurveillance>

[18] S. Axelsson, *Intrusion Detection Systems: A Survey Taxonomy*. Gothenburg, Sweden: Chalmers University of Technology, 2000.

[19] Z. M. Fadlullah, T. Taleb, A. V. Vasilakos, M. Guizani, and N. Kato, "DTRAB: Combating against attacks on encrypted protocols through traffic-feature analysis," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1234–1247, Aug. 2010.

[20] G. V. Nadiammai and M. Hemalatha, "Perspective analysis of machine learning algorithms for detecting network intrusions," in *Proc. 3rd Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2012, pp. 1–7.

[21] P. Gogoi, D. K. Bhattacharyya, B. Borah, and J. K. Kalita, "MLH-IDS: A multi-level hybrid intrusion detection method," *Comput. J.*, vol. 57, no. 4, pp. 602–623, Apr. 2014.

[22] J.-H. Bang, Y.-J. Cho, and K. Kang, "Anomaly detection of network-initiated LTE signaling traffic in wireless sensor and actuator networks based on a hidden semi-Markov model," *Comput. Secur.*, vol. 65, pp. 108–120, Mar. 2017.

[23] X. Liu, Z. Tang, and B. Yang, "Predicting network attacks with CNN by constructing images from NetFlow data," in *Proc. IEEE 5th Intl. Conf. Big Data Secur. Cloud*, May 2019, pp. 61–66.

- [24] P. V. de C. Souza, A. J. Guimarães, T. S. Rezende, V. Souza Araujo, L. A. F. do Nascimento, and L. Oliveira Batista, "An intelligent hybrid model for the construction of expert systems in malware detection," in *Proc. IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*, May 2020, pp. 1–8.
- [25] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "An initial investigation on sliding windows for anomaly-based intrusion detection," in *Proc. IEEE World Congr. Services*, Jul. 2019, pp. 99–104.
- [26] A. Krishna, A. Lal, A. J. Mathewkutty, D. S. Jacob, and M. Hari, "Intrusion detection and prevention system using deep learning," in *Proc. Int. Conf. Electron. Sustain. Commun. Syst. (ICESC)*, Jul. 2020, pp. 273–278.
- [27] L. Govindaraj, B. Sundan, and A. Thangasamy, "An intrusion detection and prevention system for DDoS attacks using a 2-player Bayesian game theoretic approach," in *Proc. 4th Int. Conf. Comput. Commun. Technol. (ICCT)*, Dec. 2021, pp. 319–324.
- [28] G. Baldini and I. Amerini, "Online distributed denial of service (DDoS) intrusion detection based on adaptive sliding window and morphological fractal dimension," *Comput. Netw.*, vol. 210, Jun. 2022, Art. no. 108923.
- [29] A. Kumar, K. Abhishek, M. R. Ghalib, A. Shankar, and X. Cheng, "Intrusion detection and prevention system for an IoT environment," *Digit. Commun. Netw.*, vol. 8, no. 4, pp. 540–551, Aug. 2022.
- [30] A. P. Patil, H. Premkumar, K. M. H. M., and P. Hegde, "JARVIS: An intelligent network intrusion detection and prevention system," in *Proc. IEEE 4th Int. Conf. Adv. Electron., Comput. Commun. (ICAIECC)*, Jan. 2022, pp. 1–6.
- [31] N. M. Yungaiela-Naula, C. Vargas-Rosales, J. A. Perez-Diaz, E. Jacob, and C. Martinez-Cagnazzo, "Physical assessment of an SDN-based security framework for DDoS attack mitigation: Introducing the SDN-SlowRate-DDoS dataset," *IEEE Access*, vol. 11, pp. 46820–46831, 2023.
- [32] I. Al-Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection," *Big Data*, vol. 9, no. 3, pp. 233–252, Jun. 2021.
- [33] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, and B. Carro, "Network intrusion detection based on extended RBF neural network with offline reinforcement learning," *IEEE Access*, vol. 9, pp. 153153–153170, 2021.
- [34] C. M. K. Ho, K.-C. Yow, Z. Zhu, and S. Aravamuthan, "Network intrusion detection via flow-to-image conversion and vision transformer classification," *IEEE Access*, vol. 10, pp. 97780–97793, 2022.
- [35] Z. Wang, Y. Zeng, Y. Liu, and D. Li, "Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection," *IEEE Access*, vol. 9, pp. 16062–16091, 2021.
- [36] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37004–37016, 2019.
- [37] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.
- [38] (2018). *A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)*. [Online]. Available: <https://registry.opendata.aws/cse-cic-ids2018/>



works, network security and privacy protection, the IoTs, embedded systems, and vehicular networks.

RIMSHA SAEED received the B.S. degree in electrical engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2021, with a focus on communications and networks, and the M.S. degree in computational science and engineering from the National University of Science and Technology, Islamabad, Pakistan, in 2023, with a focus on applied computer science. Her research interests include the application of machine learning algorithms in computer net-



EU Erasmus Mundus Staff Research Mobility and a Postdoctoral Fellowship under the STRONG TIES Program and the INTACT Program, respectively.

HASSAAN KHALIQ QURESHI received the M.Sc. degree (Hons.) in electrical engineering from Blekinge Institute of Technology, Sweden, in 2006, and the Ph.D. degree in electrical engineering from City University, London, U.K., in 2011. He is currently a Professor with the School of Electrical Engineering and Computer Science, NUST. His research interests include wireless networks, the IoTs, cyber-physical systems, and blockchains. He was a recipient of the



systems using anomaly detection mechanisms for novel attacks.

CHRISTIANA IOANNOU received the B.Sc. degree in computer science from San Diego State University, the Postgraduate Diploma degree in management from the Mediterranean Institute of Management, the M.Sc. degree in computer science from Florida Institute of Technology, and the Ph.D. degree from the Department of Computer Science, University of Cyprus, in 2017. Her research interests include WSNs and the IoT, focusing on security and intrusion detection



of practical solutions in several intelligent networks and cyber-physical systems, for example, computer networks, the Internet of Things, transportation networks, power networks, bio-nano-networks, and nano-metasurface controller networks. In the aforementioned networks, he has investigated issues pertinent to congestion control, information dissemination, network vulnerability, demand response, and more recently privacy and security. He has participated in several projects funded by the Research Promotion Foundation and EU.

MARIOS LESTAS (Member, IEEE) received the B.A. and M.Eng. degrees in electrical and information engineering from the University of Cambridge, U.K., and the Ph.D. degree in electrical engineering from the University of Southern California, in 2000 and 2006, respectively. He is currently an Associate Professor with Frederick University, Cyprus. His research interests include the application of control theoretic tools and optimization methods toward the development

...