Payload-Byte: A Tool for Extracting and Labeling Packet Capture Files of Modern Network Intrusion Detection Datasets

Yasir Ali Farrukh
Clean and Resilient Energy System Lab
Texas A&M University
College Station, TX, USA
yasir.ali@tamu.edu

Irfan Khan
Clean and Resilient Energy System Lab
Texas A&M University
Galveston, TX, USA
irfankhan@tamu.edu

Syed Wali
Clean and Resilient Energy System Lab
Texas A&M University
College Station, TX, USA
syedwali@tamu.edu

David Bierbrauer
Army Cyber Institute
United States Military Academy
West Point, NY, USA
david.bierbrauer@westpoint.edu

John A. Pavlik
Army Cyber Institute
United States Military Academy
West Point, NY, USA
john.pavlik@westpoint.edu

Nathaniel D. Bastian
Army Cyber Institute
United States Military Academy
West Point, NY, USA
nathaniel.bastian@westpoint.edu

***Abstract*—Adapting modern approaches for network intrusion detection is becoming critical, given the rapid technological advancement and adversarial attack rates. Therefore, packet-based methods utilizing payload data are gaining much popularity due to their effectiveness in detecting certain attacks. However, packet-based approaches suffer from a lack of standardization, resulting in incomparability and reproducibility issues. Unlike flow-based datasets, no standard labeled dataset exists, forcing researchers to follow bespoke labeling pipelines for individual approaches. Without a standardized baseline, proposed approaches cannot be compared and evaluated with each other. One cannot gauge whether the proposed approach is a methodological advancement or is just being benefited from the proprietary interpretation of the dataset. Addressing comparability and reproducibility issues, we introduce Payload-Byte, an open-source tool for extracting and labeling network packets in this work. Payload-Byte utilizes metadata information and labels raw traffic captures of modern intrusion detection datasets in a generalized manner. Moreover, we transformed the labeled data into a byte-wise feature vector that can be utilized for training machine learning models. The whole cycle of processing and labeling is explicitly stated in this work. Furthermore, source code and processed data are made publicly available so that it may act as a standardized baseline for future research work. Lastly, we present a brief comparative analysis of machine learning models trained on packet-based and flow-based data.**

***Index Terms*—Network intrusion detection, Traffic classification, Packet capture, Cyber attack datasets, Payload extraction**

I. Introduction

Advancement in Information and Communication Technologies (ICT) brings exceptional convenience to our daily life. However, the world is becoming more dependent on these technological adaptations, as it impacts every aspect of society and people's lives in one way or another. The world we used to know is no longer similar, as it has transformed into a collaborative network in which everything is interconnected [1]. This inter-connectivity has led to an increase in cyber-attacks on ICT systems [2]. Cyber-attacks on the network of ICT systems often lead to data breaches and operational halts for the company [3], [4]. Therefore, ICT systems require effective and robust security solutions.

Network Intrusion Detection System (NIDS) is often considered a feasible option to protect against network-based attacks [5], as it identifies attack behavior by analyzing the network traffic of vital nodes in a network. NIDS utilizes various approaches for the detection of malicious attack instances. The most prominent of these approaches are rule-based, flow-based, and packet-based methods [6]. Rule-based methods are typically based on feature selection to construct domain-specific rules. Anomalies are detected by comparing the extracted signature of network flow with predefined rules. Rule-based methods are effective for detecting known attacks but they heavily rely on in-depth domain knowledge.

Recently, much attention has been diverted toward applying machine learning (ML) approaches to NIDS as ML algorithms are achieving striking results in domains where it is hard to specify a set of rules for their procedures [7]. One of the reasons for this shift from human-dependent approaches to ML is that humans cannot incorporate every possible scenario, and there will always be an unanticipated condition that might be devastating for the system. Since ML approaches operate differently, it utilizes the data it gets and attempts to learn the patterns between occurrences [8]. Thus, ML approaches

become dynamic and adaptive to various scenarios without human intervention [9]. Lately, many ML approaches have been proposed in the domain of NIDS; however, most of these approaches utilize flow-based information. In flow-based methods, network traffic is analyzed over a period to extract flow-based behavior features [10]. This approach can be implemented in a centralized server to monitor massive traffic. The anomalies are detected utilizing the correlation between the traffic behavior and the corresponding characteristics. These approaches require subject matter experts to select useful features from data to detect malicious network traffic. Moreover, the extracted features require a pre-processing application, often requiring mathematical techniques to prepare the data for the ML model [11], [12]. Another significant issue with flow-based approaches is that they might end up learning which IP addresses send malicious traffic or which ports are frequently attacked. Flow-based approaches basically monitor threats at the lower level of the TCP/IP protocol stack, thereby diminishing the chance of detecting higher-level threats [13]. For such approaches, the methods used to extract and represent the information in a packet are crucial and can affect the output of ML models.

On the other hand, packet-based approaches can unveil malicious network flow by inspecting the packet payload, which refers to the network packet's user data. The packet-based approach tries to learn characteristics of normal as well as possible attack instances that have potential abnormal characteristics in the packet payload. The anomalies in attack instances might appear as a number of specific strings. For example, an SQL injection attack injects anomalous codes such as " `or 1 = 1 - -` " into SQL queries to make them always true [6]. Moreover, many attacks such as Worms, Ransomware, and Trojans are based on payload delivery, and these types of attacks might be hidden from flow-based approaches. In short, models that utilize flow-based approaches rely upon the tool's correctness that extracts information from packets. Thus, an inadequacy in the dependent tool propagates to a complete failure whereas rule-based approaches are completely dependent on in-depth domain knowledge and can not cater every possible scenario. Therefore, packet-based approaches seems like a viable option for NIDS.

Network packet payload analysis might be an effective solution for detecting network attacks since application attacks are embedded in the payload rather than the header portion of the Internet Protocol (IP) packet [14]. However, the ability to detect payload embedded attacks remains a challenge due to the absence of properly labeled packet data, a standardized dataset baseline, and dynamic structuring of protocols. Unlike flow-based NIDS datasets, which are easily available and can be utilized as a baseline for developing and comparing any new proposition, packet-based datasets do not have standard labeling or dedicated dataset publicly available for everyone. This lack of standardization leads towards incomparable and irreproducible research. In addition, there is a lot of ambiguity in the process of labeling the raw packet capture (PCAP) file, as every paper adopts its own processing method based on their own set of rules. Therefore, addressing the issue of standardization, we developed a tool (Payload-Byte) capable of extracting and labeling raw packet data from PCAP files of already available datasets like UNSW and CICIDS. This tool, which is based upon initial work by Bierbrauer et al. [15], is publicly available, and researchers can utilize the tool to generate the data, which can be treated as a baseline for future exploration in packet-based approaches. Our goal is to provide standardization solution to the future researchers so that reproducibility and comparability can be enhanced. This will also enable researchers to directly compare new approaches with the previous ones. The main contribution for this paper is as follows:

- We developed a tool (Payload-Byte) for extracting and labeling raw packet data of NIDS datasets. The complete cycle of processing and labeling is explicitly stated in this paper for ease of usage. This tool addresses the major issue of comparability and reproducibility in packet-based NIDS.
- Transformation of payload data into byte-wise data utilizing a generalized feature vector has been presented. This feature vector is independent of any protocol structure.
- A processed packet-based dataset along with the tool is made publically available to facilitate future researcher[1].
- A brief comparative analysis has been performed utilizing the extracted payload data and flow-based data for network intrusion detection.

The rest of this paper is organized as follows: Section II covers the background knowledge related to the packet structure. In Section III, an overview of related work and gaps has been highlighted. Working and methodology has been presented in Section IV. Furthermore, results are shown in Section V. Finally, Section VI concludes the paper.

II. Background

Libcap (PCAP) format is considered as the de facto standard for network packet capture, which is widely utilized in packet sniffers and analyzers [16]. This format is based on binary format, supporting nanosecond precision timestamps. Although it may vary from implementation to implementation. A general structure for PCAP format is shown in Fig.1.

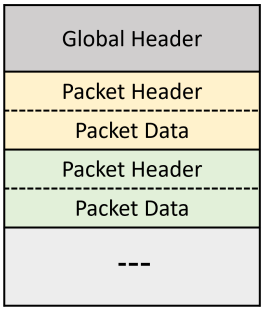

Fig. 1. The general structure of PCAP format

[1]GitHub: https://github.com/Yasir-ali-farrukh/Payload-Byte.git

For understanding the information extraction from a raw packet file, it is essential to develop an understanding of the format in which packets are stored. The PCAP format contains a global header followed by multiple packets containing packet header and packet data. The global header identifies the generic PCAP format and byte order using the "Magic Number", validates the time information stored for each capture, and permits length checks to accommodate the maximum length of captured packets (in octets). However, the individual packet header comprises packet information, like its origin, destination IP addresses, protocol, total length, and other similar features. The packet data is the actual data which is often referred as payload, containing the raw data. In practice, packets have more than one header, and each header is utilized by a different part of the networking process. Simply, packet headers are attached by certain types of networking protocols. Dividing packet into header and data is high level representation of packets, whereas if we dive deeper then there are several layers of additional information present within the raw network data. Following the TCP/IP model, each packet can be divided into four individual layers: Data Link Layer, Network Layer, Transport Layer and Application Layer. By referring to packet header, we are actually extracting information from Network Layer and Transport Layer header in our approach, which is covered in detail in Section IV. The PCAP format packets are layered following the TCP/IP model.

III. Related Work and Gaps

Much work has been done in the domain of NIDS utilizing ML approaches. However, most proposed methods use packet header information to extract features for training the model, also known as flow-based approaches. The authors in [17] present a comprehensive overview of the flow-based techniques. However, in our work, we have only focused on packet-based approaches.

A. Approaches Based on Outdated Data

Prior research has been performed utilizing several methods for detecting anomalies in a network payload. As per [14], the packet-based processing was first performed by [18], in which the authors utilized the Self Organizing Map (SOM) to distinguish between normal and abnormal characteristics of the network employing payload data. Building upon it, many payload-based approaches are presented based on Natural Language Processing (NLP) concept called n-gram. PAYL, an anomaly detector based on payload data, is presented in [19]. This approach utilizes the byte frequency distribution of normal packets to form a centroid model. However, the author use a knowledge-based structure to store the probability range occurrences of the n-gram technique to extract sequences from payload data. McPAD proposed in [20] uses a modified n-gram method to extract the features from the payload. Incorporating neural networks with ann-gram approach, the authors in [21] proposed Packet2Vec approach. In this approach, the authors utilize Word2Vec to develop a vector representation for individual most frequent n-grams. Another approach based on raw packet data using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) deep learning architectures is proposed as HAST-IDS in [22]. This approach captures low-level spatial and high-level temporal features through CNN and LSTM. Utilizing the same concept, the authors in [23] proposed AEIDS, based on an Autoencoder (AE), in which a reconstruction error and modified z-score are employed for classifying the incoming traffic instead of CNN and LSTM.

Other than these approaches, some modified approaches also curtail the total length of payload on some basis. PCNAD [24], modified version PAYL, utilizes Content-based portioning (CPP) to determine the length of payloads for different profiles. Similarly, authors in [25] proposed a payload-based attribution scheme named Compressed Bitmap Index and Traffic Down-sampling (CBID). CBID extracts feature utilizing the combination of bitmap index table and bloom filters from down-sampled traffic. Although all these approaches use payload data to detect anomalies, there is still a huge reproducibility and comparison analysis gap. Most methods in the literature utilize proprietary data or datasets that have been outdated and are already labeled. However, the obsolete dataset is not the issue here. The main problem is comparability and reproducibility. Since every method has utilized a different approach for extracting and labeling raw data without explicitly mentioning the whole process and assumptions, this has led to branching of the same tasks in several different ways [26].

B. Approaches Based on Modern Data

Since future research utilizes modern datasets containing updated attacks and data instances, our goal is to provide a standard baseline for researchers, just like flow-based approaches. Few works based on packet-based approaches have utilized updated datasets such as UNSW-NB15 and CICIDS datasets. A method based on a Recurrent Neural Network (RNN) with the attention mechanism ATPAD is proposed in [27]. This method employs the word embedding and RNN to extract features used to capture the correlation between detection results and the potential byte of the payload. This approach makes use of the CIC-IDS2017 dataset and utilizes binary classification. Moreover, no information is given regarding the extraction and labeling of raw packet files. Similarly, [6] also utilizes the CIC-IDS2017 dataset. In this approach, the author employs the payload data to construct a block sequence that contains two kinds of information that retain short-term and long-term dependency relationships among the malicious byte in payload data. In this approach, the author has stated the number of instances utilized for model training and testing. However, the amount of information given regarding the payload extraction is minimal. In terms of the UNS-NB15 dataset, an approach utilizing the header and payload data has been proposed in [8]. The authors have used a raw byte in conjunction with a specified feature vector comprising only TCP/UDP protocols. Since individual protocols have different header byte numbers, authors have fixed a feature vector to avoid ambiguity. Labeling information has been stated in this

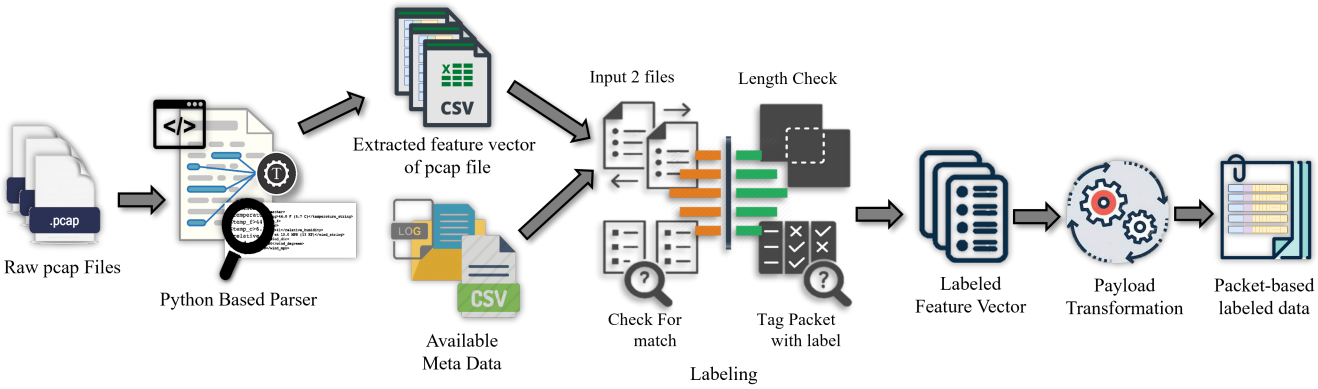

Fig. 2. Workflow representation of developed tool (Payload-Byte). Raw PCAP files are passed with available metadata for labeling and transformation of raw PCAP data into ML model readable form.

paper. However, the authors utilized only eight files of their own choice for their model training and validation. Moreover, there are more than 130 protocols in the UNSW-NB15 dataset that the author neglects. Another approach covering additional ICMP protocols has been presented in [28]. The authors presented a unified packet representation using raw packet information in this approach. The authors conducted their evaluation over ten different datasets. However, the proposed method utilizes every byte of the raw packet file, including headers containing information about IP addresses and ports. Using every byte of the packet may simply train the model to learn which IP addresses send malicious traffic or which port are attacked frequently, which is not a robust approach.

In short, every approach proposed in the literature utilizes its own methodology for extracting and labeling raw packet files for available data or its own proprietary dataset. In addition, most of the approaches used for labeling do not seem adequate and have a complication, as they use the 5-tuple approach. There is no doubt that packet-based NIDS has the potential to detect attacks in a network. Still, to gauge the true applicability, researchers need a standardized baseline that can utilized for the proposed scheme. In this way, the problem of reproducibility and comparability will be resolved. In this work, we developed a tool after undergoing a complex process of in-depth analysis of the datasets and methodology to provide a baseline solution for future researchers. The objective of the developed tool is to provide a generalized packet-based dataset that can be utilized by anyone according to their model. The detail of our adopted methodology is presented in Section IV. Moreover, labeled raw packet data has also been made available for researchers' ease.

IV. Methodology

A detailed overview of the adopted procedure for extracting and labeling raw packet data is provided in this Section. Since our goal is to provide ease and a frame of reference to future researchers, we only considered the modern and preferred network intrusion detection dataset to explain our tool. In addition, a concise summary of available network intrusion detection datasets and their characteristics is also presented. Lastly, a brief overview of the selected approach for evaluating packet-based approaches is also presented in this section.

A. Network Intrusion Detection Datasets

There are many publicly available datasets for researchers in the domain of cybersecurity. However, most network intrusion detection datasets only contain header information of network packets. Several datasets comprising real network traffic either do not have payload data or it has been removed due to privacy concerns [5]. The unavailability of labeled packet data is a significant issue in the packet-based NIDS. As a result, packet-based approaches are evaluated on proprietary or self-labeled data, resulting in reproducibility and comparability problems. Table I provides a comprehensive summary of the available datasets based on nine features. The features comprise the year of publication, accessibility of the dataset (whether the dataset is publicly available or not), format of the dataset (either flow or packet), size of the dataset, availability of labeled packet data, kind of traffic (real or emulated), whether the data is balanced or not, availability of modern network attacks, and whether the data contain metadata or not.

However, many datasets contain raw packet data information, and not every dataset is being utilized in the ongoing research. The reason is that every dataset has limitations and challenges due to the methods and environment used for creating them. Moreover, many of these datasets are outdated due to the technological advancement accompanied by new and more complex software and network structures. But still, the most widely used and up-to-date datasets available are CICIDS 2017 and UNSW-NB15 [29]. Therefore, we selected these two datasets for the explanation of our tool.

B. Workflow Overview

The developed tool, Payload-Byte, consists of three main components: python-based parser, labeling module, and payload transformation module. Python-based parser and payload transformation module are generalized components, and their methodology and approach are similar for every dataset. However, the labeling module is dataset specific, and its approach

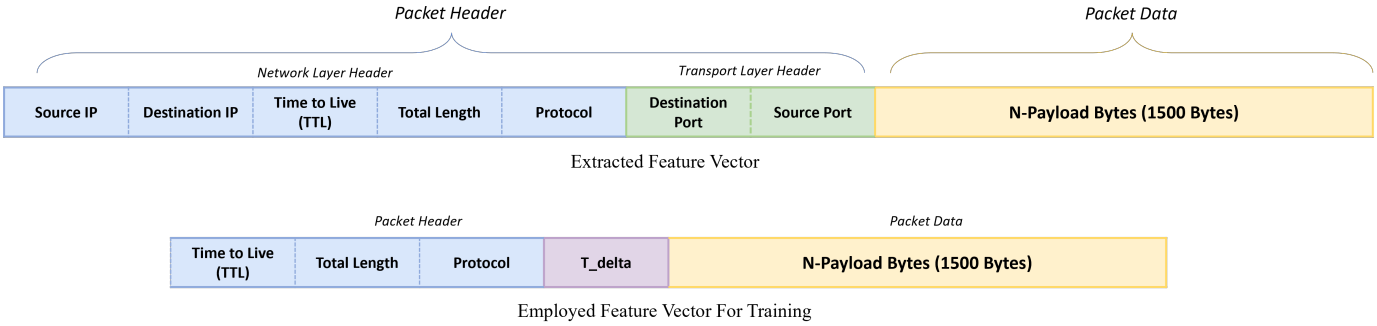

Fig. 3. Feature Vector representation of extracted data and data utilized for training the ML model. T-delta in employed feature vector is the time difference between packets.

TABLE I
OVERVIEW OF THE WIDELY UTILIZED INTRUSION DETECTION DATASET IN THE LITERATURE

Dataset	Year of Publication	Accessibility	Format	Size Count	Traffic	Modern Attacks	Balanced	Metadata	Labeled Packets
NSL-KDD [30]	1998	Public	Other	150k points	Emulated	No	No	No	-
DARPA [31], [32]	1998	Public	Packets, logs	4.9M points	Emulated	No	No	Yes	Yes
KYOTO 2006+ [33]	2006	Public	Other	93M points	Real	No	No	No	-
UNIBS [34]	2009	On Request	Flow	79k flows	Real	No	No	No	-
Botnet [35]	2010	Public	Packet	14GB packets	Emulated	No	No	Yes	Yes
ISCX 2012 [36]	2012	Public	Packet, Flow	2M flows	Emulated	No	No	Yes	No
CIC DoS [37]	2012	Public	Packet	4.6GB packets	Emulated	No	No	No	Yes
CTU-13 [38]	2013	Public	Packet, Flow	81M flows	Real	No	No	Yes	No
UNSW-NB15 [39]	2015	Public	Packet,Flow	2M points	Emulated	Yes	No	Yes	No
CIC-IDS2017 [40]	2017	Public	Packet, Flow	4.6GB packets	Emulated	Yes	No	Yes	No
CIC-IDS2018 [41]	2018	Public	Packet, Flow, logs	450GB logs	Emulated	Yes	No	Yes	No

differs from dataset to dataset, which is explained later. An overview of the workflow is illustrated in Fig. 2. Raw PCAP files and metadata are fed into Payload-Byte as input, where processed and labeled payload data is received at the output. Payload-Byte can also obtain parsed PCAP files in the form of CSV and labeled PCAP file without transformation. Provision of these files enables researchers to employ their own inferring for the extracted payload data while still having that standard baseline.

Several tools are available for analyzing and extracting information from packet capture files, such as Wireshark, Chaos Reader, Tcpflow, Network miner, and many others. However, most of these tools are Graphical User Interface (GUI) based and require a lot of computation and processing power, which is unsuitable for any tactical environment. A programming-based parser is preferred for such an environment, which can be operated on any resource-constraint device. Keeping this need in view, we developed our parser based on the Scapy python module [42]. Since the first step in labeling any packet capture files is extracting information, we developed a generalized PCAP file parser that can be utilized for parsing PCAP format files.

There are many approaches for extracting information, as mentioned in Section III. Since our focus is on a payload-based intrusion detection system, we laid out our feature vector for packet-based approaches in such a way that raw bytes are captured from packet data, and features are extracted from the packet header. We have not utilized the raw bytes of header due to its dynamic structure. As there are many protocols, each protocol's header size and order are distinct. Therefore, utilizing raw bytes for the header would lead to learning complications for the model. Moreover, IP addresses in the network layer and port fields in the transport layer can cause the ML model to form an unreliable bias towards these bytes. Since these features are commonly associated with preferences and specific services, they can create a communication pattern. But they can change anytime; therefore, these features are unreliable for training a model. Unlike [11], in which the authors limited their information extraction to the transport layer and only involved two protocols, TCP and UDP, the developed parser can extract information from the application layer too. A pictorial representation of extracted feature and feature vector utilized in training the model is presented in Fig. 3. As shown in Fig. 3, IP addresses and ports are only extracted for labeling the raw PCAP file data with reference to available metadata. Since the length of the payload changes with each packet, the maximum length that a packet can attain is considered to avoid any overflowing or truncation of the payload byte. As per the de facto packet size limit of 1500 bytes [43], [44], we set a 1500 bytes range for the payload to incorporate every byte. Our goal is to extract the data in such a way that it is complete and researchers can reduce this range as per their need. Furthermore, the payload was divided with respect to bytes, transforming into 1500 features. This transformation is necessary for the training of the ML model. The utilization of NLP techniques for payload data is not adopted; instead, the payload is transformed from hex value to integer having a range of 0-255. Zero padding was employed where the number of payload bytes are less than 1500 to maintain the standard structure of the feature vector. Further

detail on parsing concerning individual datasets is provided in their subsequent heading.

The next step after extraction is labeling, which is performed by comparing the extracted features from the PCAP file and features from the ground truth table. However, the generalized labeling approach is not possible for both datasets due to dataset-specific complications explained in their respective subheading. Inner merge (utilizing the divide and conquer algorithm) is adopted for comparing and labeling the PCAP file with CSV while preserving the order of PCAP files. Since PCAP data has packets in the range of microseconds, the number of labeled PCAP data is higher than the data instance in the CSV file which is shown in Section V. Further, to facilitate the availability of data publicly and ease the data usage process, data reduction is carried out. All the data instances whose packet data has no payload are removed. Moreover, normal data instances are under-sampled to mitigate the unbalanced issue of the dataset. The processed payload data for both datasets are made available along with the source code of the developed tool so that researchers can cross-check the procedure or utilize it to generate the complete data without data reduction. The further detail of labeling and parsing for the individual dataset is explained below.

1) ***UNSW-NB15****:* The UNSW-NB15 intrusion detection dataset encompasses nine modern attacks and network traffic emulated in a small environment. The network traffic is captured for more than 31 hours and is spread out in 79 different PCAP files, having more than 99GB of data. The dataset comprises raw network traffic in the PCAP file format and labeled flow-based data with additional attributes. We have utilized the PCAP and CSV files having labeled data instances for our tool. Moreover, an in-depth dataset analysis was performed by deploying various approaches for accurate and effective comparison and labeling. However, several ambiguities were found in the dataset, which is discussed next.

First of all, CSV data requires a lot of prepossessing. There are many missing and null values in the dataset. Furthermore, there is inconsistency in the labeling of data instances. Similar attack classes are labeled differently. Around 480,630 data instances are duplicated, and more than 130 protocols are present in the dataset. The protocols are from different layers: the transport layer and the application layer. Secondly, some corrupted data are present in the dataset, such as the source and destination ports of ICMP protocols are in hex values. Thirdly, the time stamping in the UNSW-NB15 dataset is in Unix epoch format and has a starting and ending time for the packets. But the epoch time is rounded up to integers, losing its microseconds which could be useful for accurately labeling the packets. Also, the dataset has a feature duration which was used to generate the ending time for packets, but for some data instances, this feature does not add up correctly. Fourthly, several protocols, such as *udt* and *any* do not exist.

Several approaches were tried to achieve the most optimal approach for comparing and labeling the PCAP data. The first attempt was made by labeling the PCAP packets by utilizing five features: *Source IP, Destination IP, Source Port, Destination Port, and Starting time*. Protocol feature was not included as there are more than 130 different protocols. However, the obtained results were not satisfactory. Therefore, different protocols having exact naming to the ones present in the CSV file were incorporated into the python based parser. Since these protocols are from different layers, python based parser is programmed to extract the application layer protocols. The top 45 protocols concerning attacks label counts are hard coded in the program, and the rest of the protocols are mapped under *others*. As the number of data instances after the top 20 protocols is insignificant, they are mapped under *other* category. Similar mapping is performed for the CSV file too. In addition to protocol, *dur* feature from CSV and *t-delta* feature from pcap files is also utilized for mapping. Subsequently, the results obtained are not to par. Manual data exploration was performed and it was deduced that the *dur* feature also has ambiguity. Therefore, *t-delta* feature is added to starting time of the PCAP file to attain the ending time. Both of the Unix time stamps are rounded-off to transform them into integers. Here type casting is not performed directly, as type casting would truncate the floating points, which is not the case in the CSV file. While exploring data, it is also inferred that the Time to Live (TTL) feature of the pcap file maps to the source TTL feature of CSV. Therefore, eight different features are utilized for comparing and labeling pcap files. These features are: *Source IP, Destination IP, Source Port, Destination Port, starting time, ending time, protocol and time to live*.

Since ICMP protocol has corrupted destination port and source port, it was labeled without including these two features. Similarly, ARP protocols also do not have a destination or source port. Moreover, there are some protocols whose IP addresses are not available in PCAP file format; therefore, they were not labeled automatically. After labeling, duplicate data instances are removed, and benign data is under-sampled to 1.5 times of second highest attack instances. After that, data is transformed into 1504 features, converting payload hex string into 1500 byte-wise data represented in integers. The remaining 4 features are from packet header as shown in Fig. 3.

2) ***CIC-IDS2017****:* The CIC-IDS2017 intrusion detection dataset is specifically developed to represent more modern network flows, and attacks than the preceding datasets mentioned in Table I [40]. The dataset consists of 48.8GB of network traffic captured in five separate files over five days. The dataset is released in two different formats: raw network traffic in the PCAP file format and extracted flow-based data having a set of different features in CSV format. The authors have additionally provided metadata about IP addresses and attack duration. We utilized the PCAP and CSV files having labeled data instances for our tool. However, some ambiguity in the CSV data files and PCAP files led to labeling based on flow-ID (Source IP, Destination IP, Source Port, Destination Port, and Protocol). Some of the prominent ambiguities are highlighted below.

First of all, CSV data requires a lot of prepossessing, there are several missing values in the dataset, and four columns are duplicated. Secondly, only three protocols are available in

the CSV file: TCP, UDP, and others. Whereas, in PCAP files, we found that there are Address Resolution Protocol (ARP), Link Layer Discovery Protocol (LLDP), and Cisco Discovery Protocol (CDP) which are not part of IPv4 or IPv6. Moreover, the PCAP file also contains Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP), and Stream Control Transmission Protocol (SCTP) packets. These protocols are neglected as they are not included in the CSV file. Thirdly, time-stamping in the CSV file is in the general format of "`dd/MM/YYYY HH:mm:ss`" rather than Unix epoch time stamping as in the UNSW-NB15 dataset. However, 529,450 data instances (around 19%) in the CSV file are missing seconds in time format. This leads to inaccurate time calculation for comparison and labeling of the PCAP file. Moreover, the time format in the CSV file follows the 12-hour clock format, but it is found that every data instance is in AM, which is not the case with the PCAP file. Just for experimentation, we labeled the data with the inclusion of time-stamping, and found that only 80 data instances were matched and labeled for one PCAP file. That is why we have not utilized time stamping in our labeling approach for CIC-IDS2017. Furthermore, one important thing observed while labeling is that the time stamping in PCAP files is in UTC$\pm$0:00. In contrast, the CSV files are in Atlantic Daylight Time (ADT) which is equivalent to UTC$-$03:00. Therefore, time stamping in CSV files are converted accordingly for proper execution.

Keeping these issues in mind, we dropped the time stamp feature as our tool's comparison base for CIC-IDS2017. We utilized the *Source IP, Destination IP, Source Port, Destination Port, and Protocol* feature for matching the packets with labeled CSV data. After data instances are matched, we utilized the time duration of each attack as specified in metadata to cross-validate the data instances. Here we used the time-stamp of the CSV file rather than the PCAP file since metadata is based on CSV file time-stamps. For the given time of attacks as per metadata, we removed benign instances from them to eliminate complications. However, the CSV file also contains benign data instances in the time frame of attacks as specified by metadata. After cross-validation of labeled PCAP data, duplicated values are removed and benign data is under-sampled to 1.5 times of second highest attack instances. After that, data was transformed into 1504 features, converting payload hex string into byte-wise data represented in integers.

The adopted approach for both datasets is deduced after extensive exploration of the dataset and methods. Therefore, the extracted data can be utilized as a standard baseline for future and current work. The finding of the processed data is presented in the next Section.

C. Incorporated Models

We performed a comparative analysis using several ML approaches between packet-based and extracted flow-based data. The objective of this comparative analysis is to provide a comparison between both types of data. Therefore, no hyper-parameter tuning has been performed for the approaches, and default parameters are used. The goal here is not to achieve the best results but to provide a brief comparison. Moreover, the results are not compared with recent available payload-based approaches as they have not explicitly stated their data extraction and assumptions approach. Therefore, it is not a feasible option. Additionally, available approaches have only utilized binary class classification and presented it as an anomaly detector. However, we have performed a multi-class classification and results are provided in the results Section.

TABLE II
MODEL ARCHITECTURE FOR THE CNN-LSTM AND DNN

Parameter	Description	
Activation Function	Softmax	
Loss Function	Categorical cross entropy	
Optimizer	Adam & lr=default	
Epochs	30	
	DNN	**CNN-LSTM**
Number of Layers	3	5
Layer 1	Fully Connected (None,1024)	Conv1D (None, 1504, 64)
Layer 2	Fully Connected (None,512)	Maxpooling1D (None, 752, 64)
Layer 3	Fully Connected (None,10)	BatchNormalization (None, 752, 64)
Layer 4	-	LSTM (None, 64)
Layer 5	-	Fully Connected (None,10)

The approaches that are adopted are: Random Forest, Logistic Regression, K-Nearest Neighbour, AdaboostClassifier, Multilayer Perceptron, Deep Neural Network (DNN) and a simple combination of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). The architecture utilized for DNN and CNN-LSTM is shown in Table II. Similar architecture is utilized for the CSV and labeled PCAP files. However, input of the model is different for CSV and PCAP files.

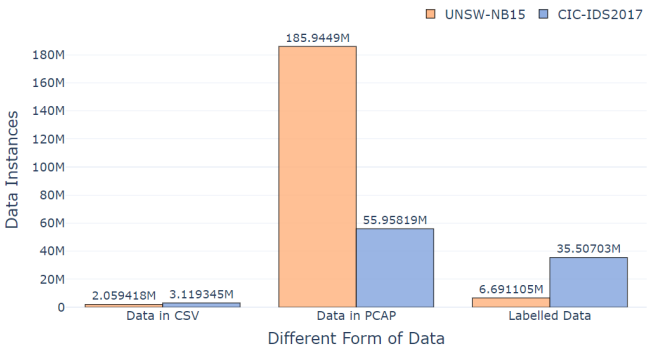

Fig. 4. An overview of the data processing and achieved outcome of the Payload-Byte. Both of the datasets are plotted side by side for better inferring.

V. RESULTS

A comprehensive overview in the form of quantitative data is presented in this Section, along with the comparative analysis of results obtained by packet and flow-based data. For the implementation of our developed tool, CSV files of both datasets are pre-processed before passing them into the Payload-Byte, whereas PCAP files are directly fed into it. The output of the Payload-Byte is a transformed and labeled pcap file, having 1504 features as shown in Fig. 3. However, initial stage files can also be extracted from the tool, such as parsed

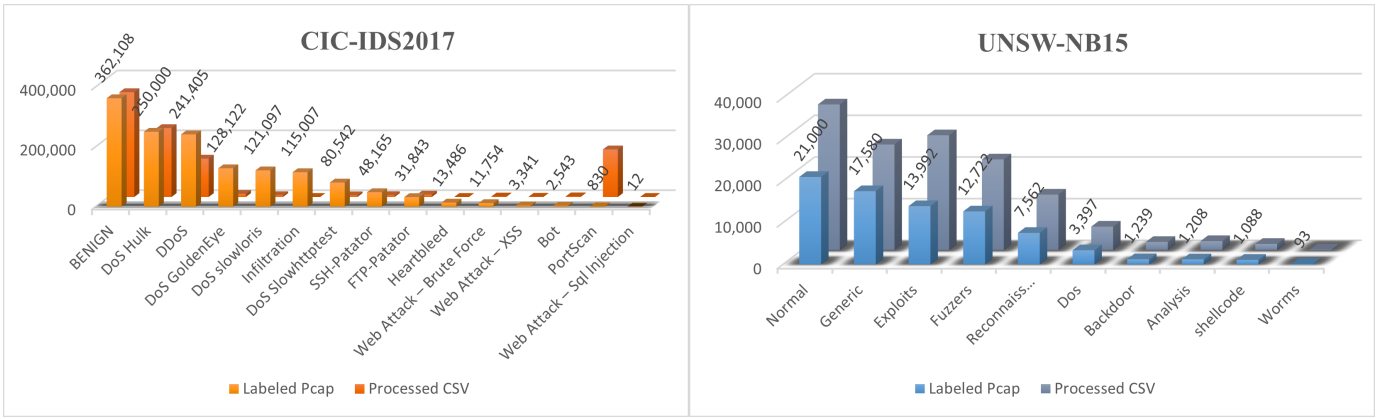

Fig. 5. A comparison of data instances for individual attack classes in Labeled PCAP and Processed CSV file. Numbers on the bar graph represents the data instances in labeled PCAP file. Normal instances are under-sampled for ease of data representation and usage.

PCAP files and labeled PCAP files having hex valued payload. An overview of the processed data is presented in the next heading.

A. Data Processing

Available CSV files for both datasets are distributed among several files. Therefore, they are combined into a single file for an individual dataset before any processing. Data prepossessing is also performed, cleaning and removing erroneous data instances. The crux of the whole processing is illustrated in Fig. 4 where data in CSV file represents the data instances in the available CSV file before prepossessing it. The total number of available packets in all PCAP files is shown as data in the PCAP file. Moreover, labeled data represents data instances obtained after labeling and removing non-payload data instances.

It can be deduced by looking at the statistical data provided in Fig. 4 that labeled data is dependent on the number of data instances present in the CSV file rather than the PCAP file. Moreover, the data instance in the PCAP file of UNSW is exceptionally high due to the number of available PCAP files. There are 79 PCAP files for the USNW-NB15 dataset, whereas only 5 PCAP files in CIC-IDS2017. Another reason for more labeled data instances in the CIC-IDS2017 dataset is that we only utilize five features for comparing and labeling, whereas eight features are utilized in UNSW-NB15.

TABLE III
SUMMARY OF UNSW-NB15 PCAP FILES

	22-01-2015	17-02-2015
Data Instances in CSV	1,082,221	1,036,218
Data Instances in PCAP	93,384,964	92,559,915
Unprocessed Labeled PCAP	11,084,503	9,745,079
Processed Labeled PCAP	6,691,105	

A detail summary of data processing based on individual PCAP files for respective dataset is presented in Table III and Table IV. Since there are 79 PCAP files in the UNSW-NB15 dataset, collective results are shown based on a particular day. On the other hand, CIC-IDS2017 only contain 5 PCAP files spanning over 5 days; therefore, they are shown individually. *Data Instances in CSV* in Table III and Table IV represents number of data points that CSV file contain in the time span of that individual PCAP file. *Unprocessed Labeled PCAP* depicts the number of packet instances labeled and *Processed Labeled PCAP* shows the number of remaining packets after processing and removal of non-payload data from *Unprocessed Labeled PCAP*.

TABLE IV
SUMMARY OF CIC-IDS2017 PCAP FILES

	Monday	Tuesday	Wednesday	Thursday	Friday
Data Instances in CSV	306794	335,415	590,692	285,825	675,965
Data Instances in PCAP	11,626,492	11,469,736	13,705,555	9,240,723	9,915,680
Unprocessed Labeled PCAP	17,096,760	20,692,983	48,931,426	15,727,986	19,740,075
Processed Labeled PCAP	5,633,567	3,516,569	16,436,931	4,071,767	5,848,193

Data instances obtained after labeling involve several duplicated values and instances where packets have no payload. Since the number of labeled packets is extensively high, they are processed by removing duplicates and instances having no payload. Afterward, the data is under-sampled to reduce its size by decreasing normal instances. The data obtained after labeling has a high number of normal instances, since normal instances in CSV files are also in significant quantity. Consequently, attack instances are not under-sampled to avoid any information loss. The total number of data instances for each attack class in CSV file and labeled PCAP file is illustrated in Fig. 5. In the figure, data labels on bar graph show the number of data instances in labeled PCAP files. The number of data instances shown in Fig. 5 is of the finalized file, processed and under-sampled. The final version of both the datasets is publicly available and can be utilized by future research work as a standard baseline. Furthermore, the complete data can also be generated by using the Payload-Byte.

B. Comparative Analysis

Labeled PCAP files and processed CSV files of the UNSW-NB15 dataset are utilized to compare both approaches briefly. The processing of CSV files include removal of features

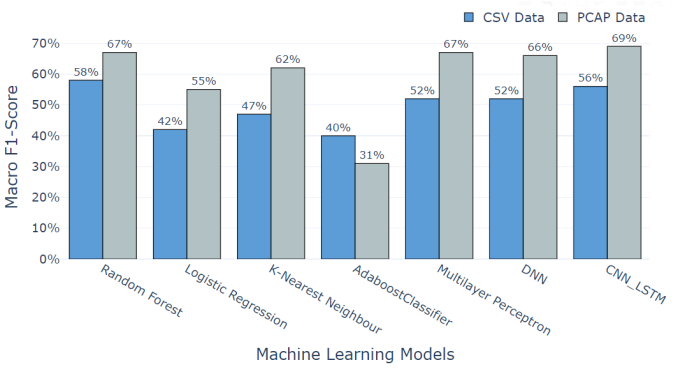

Fig. 6. A comparison of macro averaged F1-score for several ML model, utilizing packet(PCAP Data) and flow(CSV Data) based data. All models utilized have default parameters and no hyper-parameter tuning is performed.

Source IP, Destination IP, Start time, and Last time, as they can force the ML model to learn the relation between these features and attacks. Moreover, there are many missing values for the feature *ct_ftp_cmd*, therefore, it is also omitted from the CSV file. Additionally, normal data instances of CSV files are under-sampled just as PCAP files to maintain coherence. Both approaches' data is scaled utilizing Standard-Scaler before being implemented into the ML models. The resulting outcome of the comparison is shown in Fig 6. However, the overall performance of individual models can be improved by tuning hyper-parameter, which will be carried out in our future research work. The F1-score illustrated in Fig 6 is macro averaged, as it represents performance better in terms of multi-class classification. Through Fig 6 it can be deduced that PCAP data is performing well for almost every model compared to the available CSV data. A detailed result comprising of macro averaged Precision, Recall and F1-score for each model is presented in Table V.

TABLE V
DETAIL RESULT FOR PERFORMANCE COMPARISON

Models	CSV Data			PCAP Data		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Random Forest	62%	56%	58%	**66%**	**67%**	**67%**
Logistic Regression	48%	42%	42%	**57%**	**55%**	**55%**
K-Nearest Neighbour	54%	44%	47%	**63%**	**62%**	**62%**
AdaboostClassifier	**37%**	**54%**	**40%**	34%	34%	31%
Multilayer Perceptro	54%	51%	52%	**72%**	**68%**	**67%**
DNN	56%	51%	52%	**74%**	**66%**	**66%**
CNN_LSTM	61%	56%	56%	**75%**	**69%**	**69%**

VI. CONCLUSION

In this work, we highlighted the issue of comparability and reproducibility for packet-based NIDS. Since payload-based approaches can be an effective solution for detecting different network attacks, several works have been proposed in the literature. However, the data utilized for such work is either proprietary or processed data based on undefined procedures and assumptions. Moreover, there is no properly labeled packet-based dataset available that can be utilized as a standard baseline. Therefore, every new approach follows its own method for extracting and labeling packet data without explicitly stating the whole process and notion, which leads to branching the same tasks in many different ways. Addressing the issue, we developed an open-source tool (Payload-Byte) for parsing and labeling modern raw network traffic datasets. Payload-Byte standardizes dataset curation and provides a standardized baseline for future researchers to reproduce and compare other packet-based proposed approaches. Payload-Byte eliminates the engineering and language errors that stem from current datasets. Our tool can also parse high-level layers, extracting information from application layers. Since UNSW-NB15 datasets involve application layer protocols, Payload-Byte can extract and label every available protocol, which is being done for the first time. Moreover, any future datasets can also be parsed utilizing Payload-Byte as we developed a generalized parsing approach that is applicable for every packet capture file.

Furthermore, we stated the complete cycle of parsing and labeling of CIC-IDS2017 and UNSW-NB15 datasets so that it can be adopted for future research work. Payload-Byte can be utilized to reproduce the entire data and transform it accordingly. However, for ease of usage, we transform the payload data into a feature vector comprising features from the packet header and payload data, transformed into byte-wise (1500) features. Lastly, a brief comparison of flow-based and transformed packet-based data has been presented to show the effectiveness of packet-based approaches.

For our future work, we will carry out an in-depth comparison analysis of available proposed approaches on the extracted data. Moreover, we will look into additional transformation methods that can result in effective solution for packet-based approaches.

ACKNOWLEDGMENT

This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-22-2-0081, the U.S. Army Combat Capabilities Development Command (DEVCOM) C5ISR Center under Support Agreement No. USMA21056, and the National Security Agency Laboratory for Advanced Cybersecurity Research under Interagency Agreement No. USMA21035. 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. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The U.S. Government reserves a royalty-free, nonexclusive and irrevocable right to reproduce, publish, or otherwise use this data for Federal purposes, and to authorize others to do so in accordance with 2 CFR 200.315(b).

REFERENCES

[1] M. A. Khan, M. R. Karim, and Y. Kim, "A Scalable and Hybrid Intrusion Detection System Based on the Convolutional-LSTM Network," *Symmetry 2019, Vol. 11, Page 583*, vol. 11, no. 4, p. 583, 4 2019. [Online]. Available: https://www.mdpi.com/2073-8994/11/4/583/htm https://www.mdpi.com/2073-8994/11/4/583

[2] S. Wali and I. Khan, "Explainable ai and random forest based reliable intrusion detection system," 2021.

[3] 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, 7 2021.

[4] Y. A. Farrukh, Z. Ahmad, I. Khan, and R. M. Elavarasan, "A sequential supervised machine learning approach for cyber attack detection in a smart grid system," in *2021 North American Power Symposium (NAPS)*, 2021, pp. 1–6.

[5] 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.

[6] 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, 5 2022.

[7] E. Alhajjar, P. Maxwell, and N. Bastian, "Adversarial machine learning in Network Intrusion Detection Systems," *Expert Systems with Applications*, vol. 186, p. 115782, 12 2021.

[8] F. Dimitrios Tsokos Supervisor and P. Georgios, "Network Dataset Generation and Implementation of a Network Intrusion Detection System using Neural Networks," 2021.

[9] R. R. Mehdi, E. A. Mendiola, A. Sears, J. Ohayon, G. Choudhary, R. Pettigrew, and R. Avazmohammadi, "Comparison of three machine learning methods to estimate myocardial stiffness."

[10] J. T. Zhou, H. Zhang, D. Jin, and X. Peng, "Dual Adversarial Transfer for Sequence Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 434–446, 2 2021.

[11] M. De Lucia, P. E. Maxwell, N. D. Bastian, A. Swami, B. Jalaian, and N. Leslie, "Machine learning raw network traffic detection," p. 24, 4 2021.

[12] P. Maxwell, E. Alhajjar, and N. D. Bastian, "Intelligent feature engineering for cybersecurity," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 5005–5011.

[13] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, "Real time data mining-based intrusion detection," *Proceedings - DARPA Information Survivability Conference and Exposition II, DISCEX 2001*, vol. 1, pp. 89–100, 2001.

[14] A. F. Mercurio, "A Critical Analysis of Payload Anomaly-Based Intrusion Detection Systems," p. 363, 2010. [Online]. Available: https://epublications.regis.edu/theses/363

[15] R. K. P. M. . N. D. B. David A Bierbrauer, Michael J De Lucia, "Transfer learning for raw network traffic detection. expert systems with applications," unpublished.

[16] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32, 3 2020.

[17] F. Pacheco, E. Expósito, M. Gineste, C. Baudoin, J. Aguilar, E. Exposito, and C. Baudoin, "Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey," *Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02423375

[18] "A neural network approach wowards intrusion detection — CiNii Research." [Online]. Available: https://cir.nii.ac.jp/crid/1572543024790112512

[19] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection*, E. Jonsson, A. Valdes, and M. Almgren, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 203–222.

[20] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: A multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 4 2009.

[21] E. L. Goodman, C. Zimmerman, and C. Hudson, "Packet2Vec: Utilizing Word2Vec for Feature Extraction in Packet Data," 4 2020. [Online]. Available: http://arxiv.org/abs/2004.14477

[22] 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, 12 2017.

[23] B. A. Pratomo, P. Burnap, and G. Theodorakopoulos, "Unsupervised Approach for Detecting Low Rate Attacks on Network Traffic with Autoencoder," *2018 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2018*, 12 2018.

[24] S. A. Thorat, A. K. Khandelwal, B. Bruhadeshwar, and K. Kishore, "Payload content based network anomaly detection," *1st International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2008*, pp. 127–132, 2008.

[25] S. M. Hosseini and A. H. Jahangir, "An Effective Payload Attribution Scheme for Cybercriminal Detection Using Compressed Bitmap Index Tables and Traffic Downsampling," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 4, pp. 850–860, 4 2018.

[26] J. Holland, P. Schmitt, P. Mittal, and N. Feamster, "Towards Reproducible Network Traffic Analysis," 3 2022.

[27] Z.-Q. Qin, X.-K. Ma, and Y.-J. Wang, "Attentional payload anomaly detector for web applications," in *Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds. Cham: Springer International Publishing, 2018, pp. 588–599.

[28] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New Directions in Automated Traffic Analysis," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 3366–3383, 11 2021.

[29] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 9 2019.

[30] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set." [Online]. Available: http://nsl.cs.unb.ca/NSL-KDD/

[31] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2000*, vol. 2, pp. 12–26, 2000.

[32] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 34, no. 4, pp. 579–595, 10 2000.

[33] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation."

[34] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT: Picking up the Truth from the Ground for Internet Traffic *."

[35] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," *2014 IEEE Conference on Communications and Network Security, CNS 2014*, pp. 247–255, 12 2014.

[36] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers and Security*, vol. 31, no. 3, pp. 357–374, 5 2012.

[37] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling," *Computer Networks*, vol. 121, pp. 25–36, 7 2017.

[38] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers and Security*, vol. 45, pp. 100–123, 2014.

[39] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, 12 2015.

[40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," 2018.

[41] "IDS 2018 — Datasets — Research — Canadian Institute for Cybersecurity — UNB." [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[42] "Scapy." [Online]. Available: https://scapy.net/

[43] N. M. Garcia, M. M. Freire, and P. P. Monteiro, "The ethernet frame payload size and its effect on IPv4 and IPv6 traffic," *2008 International Conference on Information Networking, ICOIN*, 2008.

[44] R. J. Anthony, "The Resource View," *Systems Programming*, pp. 203–276, 1 2016.