Received 12 September 2023, accepted 12 October 2023, date of publication 18 October 2023, date of current version 25 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325391

RESEARCH ARTICLE

CNNPRE: A CNN-Based Protocol Reverse Engineering Method

JAVAD GARSHASBI, (Graduate Student Member, IEEE), AND MEHDI TEIMOURI, (Member, IEEE)
Information Theory and Coding Laboratory, University of Tehran, Tehran 1417466191, Iran
Corresponding author: Mehdi Teimouri (mehditeimouri@ut.ac.ir)

ABSTRACT Given the growth in computer networks and Internet usage, the traditional network environment has evolved into a more intricate system. Many applications utilize unknown communication protocols, for which the specification documentation is not available. The use of undocumented network protocols raises various security and management concerns. Protocol reverse engineering based on network traffic aims to infer the behavior and format of unknown network protocols. Clustering same-type messages or packets is a crucial initial step in correctly performing reverse engineering of protocol syntax or behavior. Therefore, this paper proposes a new method called CNNPRE, utilizing deep learning techniques to identify and group traffic message types. Our method employs network traffic and traffic features as input. Specifically, we use convolutional neural networks and deep transfer learning for feature extraction and message type identification and to tackle the challenge of unlabeled training data in the real world scenarios of protocol reverse engineering. The experimental results demonstrate that our proposed method works well and outperforms other methods for different protocols and achieves an average Homogeneity score of more than 0.87 on all datasets. This means that the method can identify message types according to the changing characteristics of messages and traffic features without the need for human expert intervention.

INDEX TERMS Convolutional neural networks, deep learning, deep transfer learning, message type identification, network protocols, protocol reverse engineering.

I. INTRODUCTION

Network protocols are pre-defined set of rules used by entities to ensure consistent communication. These protocols can be classified into two types: textual and binary protocols. Textual protocols are defined using readable characters and typically comprise words and symbols. Symbols are primarily used for field segmentation and serve as delimiters [1]. Binary protocols, on the other hand, are composed of bytes or bits and are commonly utilized in communication networks to ensure efficient data transmission. While several well-known open-standards protocols exist, numerous proprietary and private protocols are owned by various companies and organizations for which documentation and specifications are unavailable. Consequently, Protocol Reverse Engineering (PRE) is necessary to deduce the intricate details of unknown protocols for several applications such as network management, interoperability, network protocol security audit, simulation and conformance testing, and quality of service [2]. In contemporary security environments, the timely analysis of PRE holds significant importance, necessitating an automated approach or the acceleration of certain steps of this process. The conventional approach to PRE is performed manually by experts and is highly time-consuming, taking several months or even years, with additional challenges of recruiting, training, and retaining such human experts. Consequently, the need to automate PRE tasks was first identified, leading to the development of various approaches in recent years.

PRE approaches can generally be classified into two categories based on their input: execution trace-based and network trace-based methods. Execution trace-based methods typically rely on analyzing program execution behavior to infer protocol information from program binaries and are often more accurate [3], [4] [5]. However, the difficulty of accessing most unknown protocol programs limits the

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

practicality of these methods [6]. In contrast, network trace-based methods are more applicable as data about the target protocol is usually more accessible. These methods require network traffic between the nodes using the unknown protocol as input for PRE, which can be intercepted using various techniques.

The process of protocol reverse engineering based on network traffic traces can be divided into four main steps. The first step involves the preprocessing of the protocol data. Following preprocessing, it is crucial to group protocol messages into message types, such as Discover, Offer, and ACK for the DHCP protocol. The third step involves structural analysis to derive the protocol formats or syntax, including data fields of individual message types within the protocol. This information can then be used to derive a state machine that describes the behavior of the protocol in the final step. We note that the derivation of the protocol format and state machine is beyond the scope of our work, as there are existing methods available [7], [8], [9], [10] that can be combined with our work.

To accurately derive message formats, semantics, and behavior, the initial and crucial step is to accurately identify the message type. However, this identification process is especially challenging in two scenarios. Firstly, sophisticated protocols like binary protocols hide their structural characteristics within their compact data representation. Secondly, when the input to the PRE consists of multiple unknown protocols, it becomes more crucial to accurately identify message types and differentiate between different unknown protocols. This differentiation can significantly impact the outcome of the PRE process.

Most of the current techniques employed in PRE based on traffic analysis prioritize textual protocols that incorporate delimiters and terminology recognizable by Natural Language Processing (NLP) [11]. Binary network protocols encode data as bit patterns and often lack the structural characteristics necessary for NLP-based analysis. Among the limited number of methods designed specifically for binary protocols, Needleman-Wunsch [12] is one of the most prominent. Using such algorithms, network protocols are commonly studied as byte sequences. To address the exponential complexity of naive multiple sequence alignments [13], many message analysis applications employ the agglomerative clustering algorithms of UPGMA [14]. This algorithm recursively combines similar pairs of clusters, resulting in the construction of a phylogenetic guide tree with complexity equal to the square of the number of sequences [13].

Within the field of bioinformatics, phylogenetic trees prove advantageous as they provide insights into the evolutionary history of genome sequences. However, in the case of protocol messages, no such evolutionary relationships can be represented. Consequently, the use of agglomerative clustering, which incurs significant computational overhead, is no longer necessary. Furthermore, aligning byte values in isolation from their contextual information can lead to the loss of crucial details. This loss of context can result in the formation of erroneous value relationships among messages, particularly within independent messages of binary protocols where identical values cannot be expected unconditionally.

To identify message types, various methods have been proposed, including those that require significant manual effort to gather relevant context beforehand, and those that use clustering [11]. Unsupervised clustering is typically required for PRE since there are no known samples of correct message types available as labeled data and predicting the number of message types in advance is impossible. Hierarchical agglomerative methods are often preferred to conventional clustering methods from machine learning. However, even with unsupervised clustering, a threshold must be specified beforehand by the analyst to group similar messages together. If the threshold is set too low, there may be under-specification of message types and unrelated groups of messages. If it is set too high, multiple clusters may be created for one message type. While there are several options available for grouping messages according to their textual protocols, there is a lack of coverage of binary protocols. To improve message type identification, it is important to increase overall performance and automation efficiency for determining the optimal clustering result and parameter tuning. These methods are discussed in Section II.

In the domain of network analysis, researchers adopt a data visualization technique that entails converting network traffic data into 2D data, which enables the representation of network traffic in the form of digital images [15]. In this method, each byte in the network traffic corresponds to a pixel. It is noteworthy that the brightness of each pixel corresponds to the value of a specific byte or element, with brighter pixels indicating higher byte values. By converting 1D network traffic into fixed-length samples of higher-dimensional data, binary, gray level, or color images can be generated. The resulting image obtained via this approach resembles a texture image, encompassing a range of network traffic data.

Fig. 1 illustrates instances of network traffic images that have undergone processing via the method outlined in [16]. Through the use of flow wrapping, grayscale images are generated from matrices, wherein black pixels are assigned a hexadecimal value of 0×00, while white pixels have a hexadecimal value of 0xff. As depicted in the figure, distinct samples of the same traffic (rows) exhibit similarity, while images derived from different protocols (columns) differ from one another.

Taking network traffic samples and interpreting them as images enabled the application of image-processing techniques to network traffic, such as CNN-based neural networks [17]. This rationale prompted our decision to leverage CNN models for feature extraction to differentiate between message types within a target protocol, as well as messages arising from two or more unknown protocols.

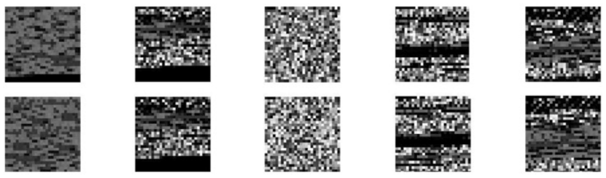

FIGURE 1. **Traffic visualization of different protocols (from left to right, Zeus, Skype, Outlook, Htbot, and Virut [17]).**

Nonetheless, training CNN models necessitates labeled data, which is not readily obtainable within the PRE domain. To overcome this challenge, we identified pre-trained models and Deep Transfer Learning (DTL) [18], [19] as the most suitable strategies. Deep transfer learning can leverage knowledge learned in previous tasks (i.e., source tasks) based on a massive semi-related dataset and apply it to new related tasks (target tasks) that have limited target data. In the field of image analysis, pre-trained models have gained popularity due to their effectiveness and ability to serve as backbones. Some of the well-known CNN models are LeNet [20], AlexNet [21], GoogLeNet/Inception [22], DenseNet [23], ResNet [24], VGG [25], XCeption [26], Inception [27], and MobileNet [28]. Our method here utilizes deep transfer learning, in which a pre-trained model is used to extract features from the network traffic data. Our experiments (presented in section V) led us to choose DenseNet-169 with weights from the ImageNet dataset [29] as the pre-trained CNN model.

The most prevalent approach in network-based PRE methods involves utilizing raw protocol data as input. Some researchers have also employed statistical features [7] or meta-data [30], [31] of network traffic as input; however, to the best of our knowledge, the combination of available network traffic features with flow payloads has not received attention. Consequently, we endeavored to capitalize on evident traffic features that are independent of packet sending or receiving times and do not necessitate any prior knowledge about the target protocol. In light of this impetus, we contend that traffic direction and traffic length serve as valuable traffic features for better message type identification.

Our proposed method focuses on automating the feature extraction and clustering of packets with similar message formats in the message type identification stage [11], based on network trace-based protocol reverse engineering. This step enables human expert analysts to analyze packets in each cluster more easily and efficiently, infer the message format of each message type, and continue the protocol reverse engineering process. Therefore, our framework provides a crucial and early step that affects subsequent steps of the protocol reverse engineering process and overall accuracy, even without prior knowledge of the protocol. Our approach, prioritizes efficacy and enhances the quality of the results simultaneously. Initially, we generate a 2D image of the network traffic instead of byte alignment, enabling us to extract message features via deep transfer learning and pre-trained CNN models. We identify message types by clustering messages according to extracted features, along with network traffic attributes such as traffic direction and message length as supplementary data. This method obviates the need for domain expertise about protocol characteristics. Since we are dealing with unknown protocols, the structure and data values of the data are obfuscated, necessitating reliance on available data about the target protocol, as in previous approaches.

We explore the most recent methods in the field to address this challenge, and the key contributions of our proposed method are as follows:

- We have introduced and built CNNPRE, a CNN-based method for feature extraction and message type identification for both binary and textual protocols in protocol reverse engineering.
- We have proposed a mechanism to use network traffic and available traffic features of the target protocol to improve performance in the message type identification step of the protocol reverse engineering process.
- To our knowledge, we are the first to propose a method that uses deep transfer learning and a pre-trained CNN model as a feature extractor on network trace data in the protocol reverse engineering field.

Experimental results on 12 different real protocols traffic datasets show that the performance of the proposed method is better than state-of-the-art related works.

CNNPRE and its proccessed datasets are freely accessible at https://github.com/JG91/CNNPRE

This paper is organized as follows. Section II describes related works in the field. Section III describes the methodology used to build the proposed CNN-based protocol reverse engineering method. Section IV presents the experiments and their environment. Section V presents the results of the applied method and discusses them. Finally, Section VI concludes the paper and outlines the ideas for future research.

II. RELATED WORK

PRE aims to obtain a comprehensive comprehension of a protocol's specifications via an analysis of the communication artifacts. This encompasses various components such as message formats, syntax, grammar, constants and keywords, message types, implicit or explicit state machines, and other relevant aspects. The subsequent subsections present an overview of some of the most commonly employed PRE techniques.

A. PROTOCOL INFORMATION (PI)

The genesis of the Protocol Information (PI) [32] project was motivated by a sequence alignment algorithm utilized in the field of bioinformatics. PI is designed to extract information from communication packets to determine the field boundary of unknown protocols. The Smith-Waterman [33] algorithm is employed to generate a distance matrix between packet sequences, followed by clustering analysis utilizing the un-weighted pair-group method and arithmetic methods. Subsequently, PI utilizes the Needleman-Wunsch algorithm [12] to ascertain the invariability of protocol fields and to determine the boundary of these fields. However, PI is only able to provide information on field boundaries and content types, necessitating further manual analysis to deduce the meaning of the content. PI's performance is limited when it comes to complex protocols, as it is only suitable for simple protocols.

B. DISCOVERER

Discoverer [1], proposed by Cui et al., is based on their earlier work, "RolePlayer" [34]. The majority of the algorithms employed in Discoverer are heuristic in nature, informed by expert knowledge. The first stage in Discoverer involves tokenization and initial clustering. Protocol packets are disassembled into text and binary fields according to their byte content, and fields are then segmented into tokens within protocol packets using delimiters. Since alignment results may be incorrect if the length is uncertain, the clustering process is performed using the token pattern rather than the sequence pattern. The second stage is token semantic inference and recursive clustering. Token properties such as the type of content and variable type are usually straightforward to identify. Discoverer establishes heuristic rules for modeling token semantics, such as length, offset, format distinguisher, and cookie. Based on the format distinguisher, Discoverer recursively clusters the initial clustering results. Finally, merging and sequence alignment are performed. Discoverer is an effective tool for reverse analysis of a broad range of protocols. However, it primarily relies on heuristic algorithms and expert knowledge and is only capable of handling text-based protocols.

C. DEEP LEARNING BASED METHODS

Recently, machine learning and deep learning approaches have been proposed for better feature extraction in PRE. In Natural Language Processing, the N-gram model is an important tool in extracting message features [35], [36]. Yang et al. [37] proposed a deep learning approach to infer field change features and propose a field sequence encoding approach that utilizes the LSTM-FCN model. However, the input contained only TCP and IPv4 protocol information, and the training model was not universal. In their study, Wang et al. [38] used CNN to classify binary message bytes and further segment the format. The accuracy of this method was higher than the clustering algorithm; however, the training phase is inapplicable in PRE field because of the nature of the problem. Zhao et al. [39] segmented the format using the LSTM-FCN model. Ning et al. [40] presented a reverse engineering tool for industrial control protocols, using bootstrap voting expert algorithm and BiLSTM. The PREUNN [41] study in 2022 proposed a reversing architecture using a variety of neural networks based on deep learning. PREUNN represents the most relevant work to our proposed approach as it employs a modular suite of deep learning techniques to develop a reverse tool for text-based protocols. The research indicates that the method achieves optimal results when an auto-encoder is used for feature extraction, a long short-term memory (LSTM) for feature and state recognition during reverse engineering, and a Self-Organizing Map (SOM) algorithm for clustering. However, PREUNN does not consider binary protocols and, there is one weakness in the paper that needs to be addressed. Specifically, the authors report making modifications to the dataset used in their experiments, such as balancing the dataset and stripping data fields from each packet. While the authors provide some justification for these modifications, it is not clear how they impact the performance of their method in a real-world setting. This introduces a potential source of bias or confounding that could weaken the validity of their results. As we move forward, PRE combined with deep learning is expected to be a rising and promising subfield.

D. OTHER METHODS

Beddoe [32] initially proposed utilizing sequence alignment to perform an analysis of network protocols. Since then, various algorithms from natural language processing and bioinformatics have been employed for this purpose.

ScriptGen [42], Discoverer [1], and Netzob [9] use sequence alignment to determine message types. ScriptGen also suggests deriving tokens from frequency, variance, and other byte characteristics across all messages in a trace. While ScriptGen and Discoverer claim their methods are universally applicable, analyzing binary protocols is left for future research. Notably, all presented methods are based on agglomerative clustering by UPGMA [14], which limits the analysis of large traces and disregards byte contexts, as described in the introduction.

The FieldHunter method [43] incorporates various concepts from related approaches such as Netzob, Discoverer, and ScriptGen to address challenges in the field of format inference, including characterizing field types. However, FieldHunter still relies on byte-value features, which do not provide information about the structure of messages.

In addition to network-based algorithms, certain algorithms combine network-based and execution-based approaches, including Polyglot [3], AutoFormat [44], Tupni [45], and Reformat [46]. These algorithms are utilized to extract information from unknown protocol packets to determine the protocol's field boundary and field semantics. However, since these algorithms necessitate access to the implementation of unknown protocols, we exclude them from consideration. Other algorithms are available for inferring state machines for protocols, such as PEXT [30] and Prospex [5], which are outside the scope of this research.

As delineated in the literature review, the current solutions for network protocol analysis have limited robustness as they perform byte-wise analyses and there is a dearth of coverage of binary protocols. In addition, the computational expenses linked with conventional sequence alignment and agglomerative clustering make it infeasible to analyze large data sets of traces.

III. PROPOSED METHOD

This section delineates the methodology and structure of the proposed PRE method, which comprises multiple steps executed sequentially. The process encompasses preprocessing of network traffic data, extraction of traffic features, extraction of payload features via a pre-trained Convolutional Neural Network (CNN) model, and clustering. Our proposed method differentiates itself from conventional approaches by ensuring high accuracy without necessitating manual parameter adjustments or feature selection by human experts. Fig. 2 depicts the processing steps involved in the proposed PRE method.

- Data pre-processing: We begin with data pre-processing, which prepares the network traffic for our method. The following steps are included in this phase: target protocol extraction from network data, calculating traffic characteristics, such as traffic direction and traffic length, trimming and padding, and converting traffic data to images to fit traffic data to the pre-trained CNN model.
- Feature extraction and dimension reduction: The second phase comprises feature extraction and dimension reduction. We used DenseNet-169 as the pre-trained CNN model for feature extraction, and for dimension reduction, we utilized the Uniform Manifold Approximation and Projection (UMAP) method [47].
- Message type identification: As a final phase in the proposed method, clustering will be performed to identify the different types of messages. To achieve this goal, we used the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering algorithm [48].

Detailed descriptions of the methods and steps involved in each phase of the proposed method are provided in the following subsections.

A. DATA PRE-PROCESSING

The data pre-processing steps are as follows.

1) DATA INGESTION

For this step, we developed a Python script. We used the Scapy library in our code to import network traffic data in PCAP formatted files, extracting target protocol data and traffic feature (direction and length) calculations for the next step. We assumed that all protocols at the network layers below the target protocol were known and their headers could therefore be stripped from the packet.

2) TRAFFIC FEATURES CALCULATION

The proposed method consists of a step to use protocol traffic features (also known as protocol flow features) as additional data for the next steps. In this method, we calculate protocol packet length (number of bytes) and protocol packet direction as supplementary features in the subsequent clustering stage to differentiate packets from different message types and also packets from different unknown protocols. Using the OSI model's layers 2, 3, and 4, we were able to determine the direction of each packet. The direction was determined by categorizing the packet's movement as either uplink or downlink, server-to-client or client-to-server, and broadcast or non-broadcast. However, it's important to note that there is only one of these directions for packets of each protocol i.e., for GSM we have uplink/downlink and for HTTP we have server-to-client/client-to-server directions.

As a sub-step, we used the K-Means clustering method to cluster messages based on their length. As a result of analyzing the lengths of packets in our dataset, we have found that on average, $K = 4$ is sufficient to group packets according to their lengths. However, analysts can experiment with different K values based on the packet length distribution of their target protocol. After clustering, we store each resulting cluster ID as a K-bit label with one-hot encoding. This encoding replaces numerical integer values with a binary vector such that all but one of its elements equals zero, and the unique element equals one. Such an approach applies to numerical variables belonging to a finite set of possible values (in this method, a cluster ID value belongs to the set of possible $K = 4$ values). The one-hot encoder thus inputs an integer of K values and outputs a binary vector of size K, containing values one at the position related to the current input values and zero elsewhere. A noteworthy point to emphasize is that K is the maximum number of clusters that can be formed. For instance, if a protocol has a fixed packet length, then only one one-hot encoded bit will be used. Moreover, we use a two-bit one-hot encoded vector to label each message's direction.

3) IMAGE CONVERSION

As the proposed method will use a DenseNet-169 model to extract protocol features, we need to change the dimension of the input (traffic data) to match with input size of this model. The input size of DenseNet-169 is a 224×224 RGB image. This means that the input of this model is a square image with a width and height of 224 pixels, and it has three color channels representing red, green, and blue. Then before feeding network traffic (messages) into the DenseNet-169 model, the message should be preprocessed by resizing it to the required input size of 224×224 and normalizing the pixel values. To do this, at the first step, we pad each message with zeros to make all messages the same size as the longest message, and then we resize each message to a 224×224 grayscale binary image using the OpenCV (available at: https://docs.opencv.org/4.8.0/, accessed on 15 October 2023) resize function. This function resizes the input image to the new size by calculating scaling factors for each dimension and applying the INTER_LINEAR interpolation method, which is a bilinear interpolation method to determine the pixel values for the new image. Bilinear interpolation estimates the new pixel value by computing a weighted

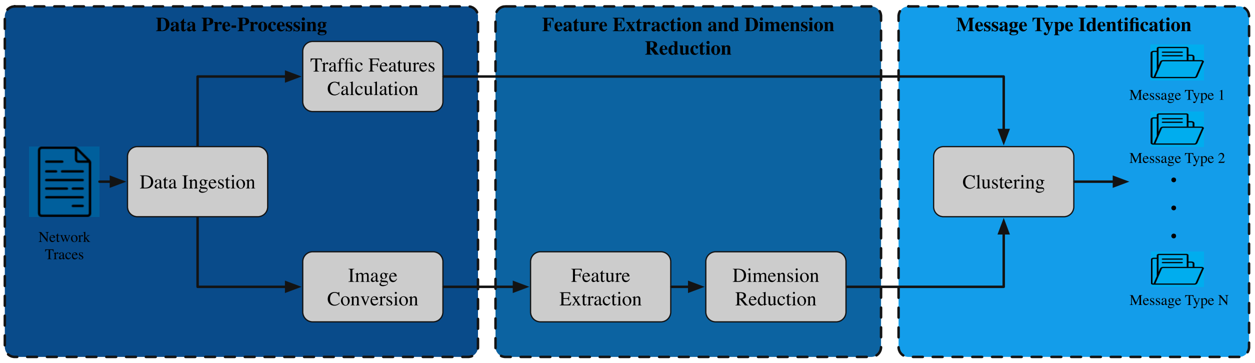

FIGURE 2. .Overview of the proposed PRE method (CNNPRE).

average of the four nearest pixels to the desired location in the input image. After this step, as the input size of the DenseNet-169 is RGB images, we used the OpenCV cvtColor function to convert the grayscale image of each message to an RGB image. It does this by iterating over each pixel in the input grayscale image and creating a new three-channel pixel value by replicating the grayscale intensity value three times, one for each color channel. The new pixel value is then assigned to the corresponding pixel in the output RGB image. This process is repeated for every pixel in the input image, resulting in a full-color RGB image that has the same spatial dimensions as the input grayscale image ($224 \times 224 \times 3$). The output of this step is an RGB image of each message which can be used as the input of the pre-trained DenseNet-169 model.

B. FEATURE EXTRACTION AND DIMENSION REDUCTION

The steps of the second phase are as follows.

1) FEATURE EXTRACTION

For this step, we used a pre-trained DenseNet-169 model with weights from ImageNet to extract the features from the packets. DenseNet-169 is a deep learning architecture for image classification tasks. It is a member of the DenseNet [23] family of models, which connect each layer to every other layer in a feed-forward fashion. DenseNet-169 has 169 layers and has layers in the four dense blocks. It is a widely used architecture for deep learning classification tasks. The output of the pre-trained DenseNet-169 model is a feature vector with a length of 1664 for each packet.

2) DIMENSION REDUCTION

In this step, since the length of the feature vectors is very long (1664 bits), for performance improvement and to avoid the curse of dimensionality, we used the Uniform Manifold Approximation and Projection (UMAP) [47] method to reduce the dimension of feature vectors to 10 bits. UMAP is a manifold learning technique for dimension reduction and is competitive with t-SNE for visualization quality, and arguably preserves more of the global structure and is more time efficient. As well, UMAP does not impose computational restrictions on the embedding dimension, which makes it suitable as a general-purpose technique for dimension reduction in machine learning.

C. MESSAGE TYPE IDENTIFICATION

The final phase in our proposed method is the message type identification process and its include clustering step.

1) CLUSTERING

The main challenge in this step is to automate the configuration of the parameters of the clustering method. The goal of our approach is to develop a method that does not require any prior knowledge of the protocol since, in the real world, we do not have any information regarding the target protocol. For this purpose, we found that HDBSCAN [48] is an appropriate tool. The HDBSCAN clustering algorithm extends DBSCAN by converting it into a hierarchical clustering algorithm and then applying a technique to extract a flat clustering based on cluster stability. Because HDBSCAN utilizes density-based analysis, there is no assumption regarding cluster shape. It does not require a target number of clusters as a parameter, and it deals with noise by not assigning samples to any cluster rather than selecting the wrong one. To identify message types by clustering, we used the 10-bit vectors of reduced extracted features of each message and their 6-bit traffic features as inputs to HDBSCAN.

D. GRAPHICAL ABSTRACT

The graphical abstract of the proposed method for a network message with l bits length from a dataset with a maximum message length of m is presented in Fig. 3. In this figure, D1 and D2 are direction bits and L1, L2, L3, and L4 are message length bits in one-hot encoding presentation.

Fig. 4 illustrates the 224×224 average images of each message type of the GSM protocol traffic images that have been processed via our proposed method before converting to RGB and feeding to DenseNet-169. As depicted in the figure, distinct message types differ from one another.

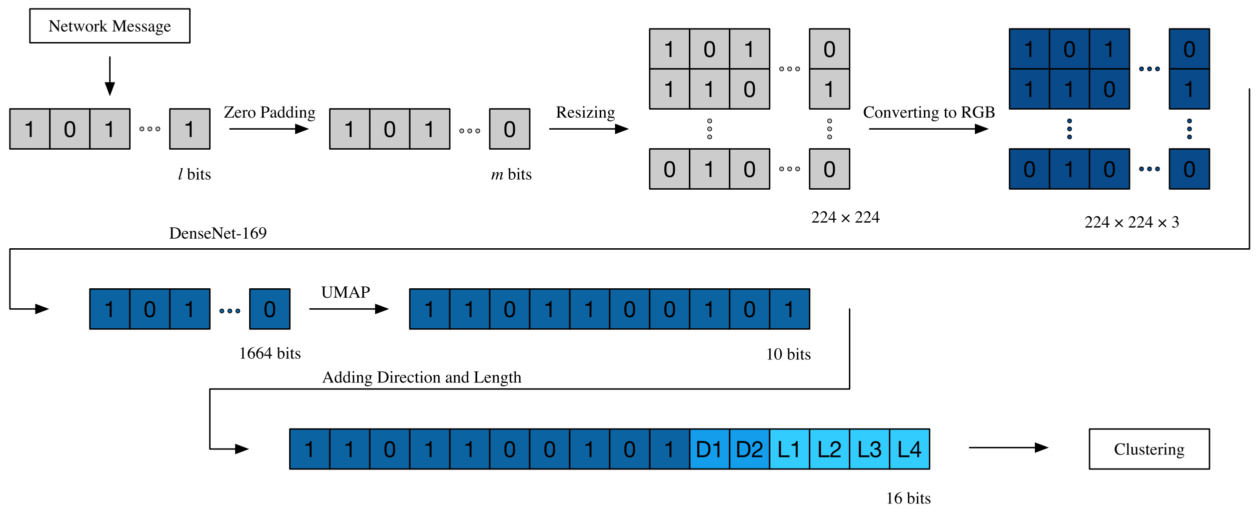

FIGURE 3. The graphical abstract of the CNNPRE method.

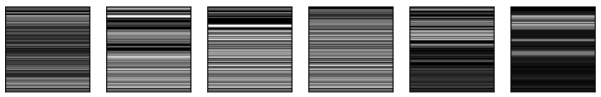

FIGURE 4. Traffic visualization of different message types of the GSM protocol (from left to right, BCCH, AGCH, PCH, SDCCH/4, SACCH, and CBCH).

IV. EXPERIMENTS

A description of the experimental environment we established for evaluating our method is presented in this section.

A. DATASET

This research endeavor aims to identify the diverse message types of unknown protocol packets by grouping packets that share similar formats. Hence, it is necessary to have a wide range of protocols and message formats to thoroughly assess the proposed method. As such, we obtained network traffic capture files in the PCAP format from the Western Regional Collegiate Cyber Defense Competition (WRCCDC) regional competition for 2021, which can be downloaded from the WRCCDC archive (available at: https://archive.wrccdc.org/pcaps/2021/regionals/feed1/, accessed on 15 October 2023). A PCAP file of MDSec is also available for download for GSM data, which encompasses packets from a 2.5G environment, including uplinks and downlinks, and two mobile stations (available at: https://github.com/JG91/CNNPRE, accessed on 15 October 2023). Subsequently, we developed a Python program that utilizes the Scapy library to filter and analyze the target computer network protocols in each PCAP file.

To ensure the quality and relevance of our annotations, we meticulously engaged the expertise of domain specialists and adopted a judicious approach to message type definition. In the quest for a comprehensive and precise dataset annotation, we enlisted the guidance and insights of experts from the fields of computer networks, telecommunications systems, and reverse engineering. The collective wisdom of these experts not only informed our annotation process but also provided us with a multidimensional perspective on the nuances of network communication. One of the pivotal aspects of dataset annotation is the definition of message types. This task involves categorizing network communication messages based on various attributes, including protocol fields and the underlying meaning of the message content. Our primary aim was to strike a balance between granularity and comprehensibility. We sought to avoid overly coarse or fine-grained type definitions, as both extremes would compromise the utility of the dataset.

To illustrate this point, consider the HTTP protocol. In a coarse labeling approach, one may classify messages into two broad categories: Request and Response. However, we recognized that within Request messages, there existed a multitude of subtypes, such as GET and POST, each with distinct characteristics and significance. This is also true for Response messages. The complexity of message type definition becomes even more apparent when dealing with protocols like GSM. For instance, both AGCH (Access Grant Channel) and PCH (Paging Channel) fall under the umbrella of CCCH (Common Control Channel), but their message meanings and contents diverge significantly. AGCH primarily serves the purpose of channel assignment, while PCH is utilized for paging.

Additionally, the message types and fields of each protocol message were determined by comparing them with their respective reference implementation documents. For instance, in the case of DHCP packets, the Python script filtered packets initially based on the UDP protocol and port numbers 67 and 68 and subsequently determined each packet's message type following the DHCP RFC document (available at: https://datatracker.ietf.org/doc/html/rfc2131, accessed on 15 October 2023). Based on the protocols documentations, for each protocol, we employed a protocol field or a combination of fields in each message to indicate the type of each packet. For example, in the case of the DHCP protocol, the ''Option'' field was employed to distinguish message types (such as Discover, Offer, Request, Ack, Inform, and NAK). For the Syslog protocol, the combination of ''Facility'' and ''Severity'' fields was used to identify the message type of each packet. The only exception to this process is the GSM protocol, the GSM has two layers, and the existence and type of the second layer depend on the values of some fields in the first layer. Therefore, we used all 23 bytes of the GSM layer of each packet as target protocol data and determined the message type of each packet with the help of a human expert. In our dataset, GSM packets are labeled based on the channel type of each message.

Given the evident disparities in the data structure of textual and binary protocols, we incorporated both into our dataset. The target protocols for this study encompassed ARP, DHCP, DNS, FTP, GSM, HTTP, ICMP, NBNS, NTP, POP, SMTP, and SYSLOG. The dataset description, which comprises protocol type, message types, packet count per message type, and total count, is presented in Table 1.

B. SOFTWARE AND HARDWARE MATERIAL

Our approach was instantiated in Python3, utilizing the TensorFlow and Scikit-learn modules. We leveraged a pre-trained DenseNet-169 model with ImageNet's weights, which we sourced from the Keras library of TensorFlow 2. Additionally, we employed UMAP and HDBSCAN from their official libraries. UMAP was invoked with the number of neighbors set to 20, and the minimum number of each cluster size in HDBSCAN was set to approximately 1% of each dataset size. Using this minimum cluster size will prevent over-clustering (dividing the dataset into too many small clusters). The execution of our proposed methodology was carried out on Google Colab. The Jupyter Notebook of CNNPRE and the processed datasets of 12 different protocols are accessible at https://github.com/JG91/CNNPRE

C. PERFORMANCE METRICS

To quantify the performance of clustering, it is necessary to take into account the type of output. As a result of clustering, each packet of data is grouped into a cluster. Due to the unsupervised nature of this problem, each cluster is not augmented with a class label. For instance, if 100 data packets (of which 50 are HTTP GET and 50 are HTTP POST) are run through the proposed PRE method and the output is 2 clusters (one of size 45 and another 55), the clusters can only be labeled '1'and '2'(cluster labels) but not GET or POST (class labels) directly. This lack of association is why traditional classification metrics like accuracy, precision, recall, and F-score cannot be used as metrics.

To address this challenge, a majority voting algorithm can be used after clustering to determine the label of the clusters. In this algorithm, the most common ground truth class label is used to determine each cluster's label. As an example, a cluster containing 100 packets, with 90 of them being HTTP GET packets and 10 being HTTP POST packets, would be classified as an HTTP GET cluster, with any HTTP POST packets considered to be miss-clustered with the HTTP GET

TABLE 1. Dataset description.

Protocol	Type	Type-Count	Count
ARP	Binary	1. Request - 744	1275
		2. Reply - 531	
DHCP	Binary	1. Request - 506	921
		2. Discover - 334	
		3. ACK - 53	
		4. Offer - 20	
		5. Inform - 6	
		6. NAK - 2	
DNS	Binary	1. Standard Query - 2425	4876
		2. Response can do recursive - 1747	
		3. Response no such name - 388	
		4. Response no such name (another type) - 214	
		5. Response no error - 73	
		6. Standard Query (another type) - 8	
		7. Dynamic Update - 5	
		8. Dynamic Update Response - 5	
		9. Response (another type) - 4	
		10. Response (another type) - 3	
		11. Response (server failure) - 3	
		12. Response (another type) - 1	
FTP	Text	1. Response, 200 - 270	1620
		2. Request, PASS - 135	
		3. Request, PORT - 135	
		4. Request, RETR - 135	
		5. Request, TYPE - 135	
		6. Request, USER - 135	
		7. Response, 150 - 135	
		8. Response, 220 - 135	
		9. Response, 226 - 135	
		10. Response, 230 - 135	
		11. Response, 331 - 135	
GSM	Binary	1. PCH - 2263	5000
		2. BCCH - 2069	
		3. CBCH - 303	
		4. SDCCH/4 - 157	
		5. SACCH - 143	
		6. AGCH - 65	
HTTP	Text	1. GET Request - 892	1963
		2. 200 Response - 581	
		3. 302 Response - 228	
		4. POST Request - 92	
		5. 404 Response - 92	
		6. 301 Response - 72	
		7. 304 Response - 6	
ICMP	Text	1. Echo Request - 3023	3231
		2. Destination Unreachable (Host) - 131	
		3. Destination Unreachable (Port) - 53	
		4. Echo Reply - 20	
		5. Fragmentation Needed - 4	
NBNS	Binary	1. Name Query (Recursive, Broadcast) - 1244	2642
		2. Registration (Recursive, Broadcast) - 762	
		3. Registration (Broadcast) - 372	
		4. Name Query (Broadcast) - 182	
		5. Name Query - 57	
		6. Registration Response - 21	
		7. Name Query Response - 4	
NTP	Binary	1. No warning, v4, Client - 499	1043
		2. No warning, v4, Server - 488	
		3. No warning, v3, Server mode - 27	
		4. Unknown, v3, Symmetric Active - 16	
		5. No warning, v3, Symmetric Active - 13	
POP	Text	1. OK - 630	1080
		2. QUIT - 180	
		3. USER - 90	
		4. PASS - 90	
		5. LIST - 90	
SMTP	Text	1. Mail Action OK (251) - 360	1172
		2. Service Ready (220) - 90	
		3. HELO - 90	
		4. MAIL - 90	
		5. RCPT - 90	
		6. DATA - 90	
		7. QUIT - 90	
		8. Service Closing Channel (221) - 90	
		9. Start Mail Input (354) - 90	
		10. DATA Fragment (1) - 90	
		11. DATA Fragment (2) - 2	
SYSLOG	Text	1. DAEMON.INFO -1300	1792
		2. AUTH.INFO - 242	
		3. AUTHPRIV.INFO - 90	
		4. AUTHPRIV.NOTICE - 50	
		5. AUTHPRIV.DEBUG - 50	
		6. AUTHPRIV.ERROR - 25	
		7. AUTHPRIV.WARNING - 25	
		8. CRON.INFO - 10	

packets. As a result, a confusion matrix can be generated, and standard classification metrics can be derived. Dataset bias is a weakness of this voting method. In [34], this approach was employed.

A better approach would be to use metrics that measure how well similar data points are grouped and how dissimilar data points are grouped separately without the need to label each cluster with a class label. Intuitive metrics are examples of this approach that can be used. The conditional entropy analysis can be used to define these intuitive metrics given the ground truth class assignments of the samples. In particular, Rosenberg and Hirschberg [49] define Homogeneity and Completeness as two desirable objectives for any cluster assignment:

- Homogeneity: every cluster contains members of a single class.
- Completeness: all members of the class are assigned to the same cluster.

After that, we can convert those concepts into scores such as the Homogeneity score and Completeness score. They're both bounded by 0.0 and 1.0 (higher is better). By using Formula (1), we can compute their harmonic mean, which is called the V-Measure.

$$\text{V-Measure} = \frac{(1+\beta) \times \text{Homogeneity} \times \text{Completeness}}{\beta \times \text{Homogeneity} + \text{Completeness}} \tag{1}$$

A higher V-Measure score indicates that the clustering is both pure and complete with respect to class labels. It represents a trade-off between Homogeneity and Completeness. By default, the β is set to 1.0, but a value less than 1.0 will give more weight to Homogeneity, and a value greater than 1.0 will give more weight to Completeness.

To underscore the practical implications of these metrics, let's consider the following examples: Suppose we have a protocol containing 200 messages, equally divided into 100 of type A and 100 of type B. We can apply these metrics to evaluate three distinct clustering results. In the first clustering result, we observe the presence of three clusters: 90 A, 90 B, and 10 A + 10 B. In this scenario, the Homogeneity score is 0.90, signifying that cluster 1 (comprising 90 A messages) and cluster 2 (comprising 90 B messages) are pure, containing messages from a single class. However, the Completeness score is 0.66, suggesting that some messages from both class A and class B are distributed across multiple clusters, specifically in cluster 3, which contains mixed messages. Further, the V-Measure, with $\beta = 0.1$, computes to 0.87, indicating a judicious balance between cluster purity and the comprehensive representation of class instances. The second clustering result consists of two clusters: 90 A + 10 B, and 90 B + 10 A. The Homogeneity score in this case is 0.53, signifying that neither of the two clusters is entirely pure, as they both contain mixed messages. Similarly, the Completeness score, also at 0.53, highlights that not all members of a class are confined to the same cluster, with messages from both class A and class B divided between the two clusters. Consequently, both the Homogeneity and Completeness scores are relatively low, resulting in a reduced V-Measure of 0.53. Lastly, in the third clustering result, we observe twenty clusters: 10 clusters with 10 A and 10 clusters with 10 B. In this instance, the Homogeneity score is a perfect 1.0, indicating that each cluster exclusively comprises members of a single class, achieving high purity. However, the Completeness score of 0.23 reveals that these clusters may not fully represent entire classes, as some messages are missing from specific clusters. The V-Measure, calculated at 0.77, suggests a balanced trade-off between cluster purity and class completeness. Comparing the first clustering and second clustering results, the metrics convincingly support the preference of human experts for the first clustering due to its ability to generate purer and more easily interpretable clusters. The Homogeneity and V-Measure scores reinforce this choice. In the context of the comparison between the first clustering and the third clustering, the analysis points out that while the third clustering result exhibits high Homogeneity, it would require a more substantial effort from human experts due to the greater number of clusters and smaller cluster sizes. In this scenario, the V-Measure metric emerges as more adept at highlighting the challenges posed by the third clustering, given its consideration of both Homogeneity and Completeness, thereby reaffirming the preference for the first clustering method by human experts.

Our view is that these metrics are appropriate in this particular field since the identification of message types is a key step of a multistep process, and the most important part of this step is grouping similar messages into the same clusters and putting different messages into different clusters. Thus, even though same-type messages may be grouped into multiple clusters, each cluster should contain only messages of the same type. Therefore, the most appropriate way to evaluate the clustering algorithm of this step is using Homogeneity or a V-Measure with a β close to 0. So in this work, to evaluate the clustering performance, we used V-Measure with β values of 0.1, 0.2, and 0.4 in addition to the Homogeneity score.

V. RESULTS AND ANALYSIS

In this section, we present the experiments performed to test the performance of the proposed method, CNNPRE. This section were divided into three parts: (A) comparison of the performance to other PRE tools or solutions; (B) ablation analysis, to understand the performance contribution of each components of the proposed method; and (C) practical applications of the proposed method.

A. COMPARISON ANALYSIS

As part of the evaluation process, we first compare the proposed method with other related methods. For this objective, we selected Discoverer and PREUNN because these methods are the most popular and most related to our method. Additionally, we compared our method with two other approaches, K-Means clustering, and Unweighted Pair Group Method with Arithmetic Mean (UPGMA) clustering for message type identification, which is used by some related works, including PI. In these approaches, we clustered the raw messages of the target protocol without adding the length and direction features. To find the optimal K for K-Means clustering, we used the Elbow method. The Elbow method runs K-Means clustering on the dataset for a range of values for K (from 1 to 10) and then computes an average distortion score for all clusters to find an optimal value for K. For UPGMA, we used Euclidian distance and three different values for the number of clusters, the values are 7, which is the average number of message types in our datasets (UPGMA-1), 12 which is the maximum number of message types in our datasets (UPGMA-2), and 24 which is the double of the maximum number of message types in our datasets (UPGMA-3). We also reported the average results of UPGMA clustering with these three values (UPGMA-AVG) in this work as the main representation for the UPGMA algorithm. Furthermore, in the next section, we compared the results of using these clustering algorithms instead of HDBSCAN in our proposed method.

Fig. 5 shows the Homogeneity score for the proposed method (CNNPRE), Discoverer, PREUNN, K-Means, and UPGMA across 12 datasets. It shows that our method outperforms others in 9 out of 12 datasets and has the best average for all datasets. Also, in the other three datasets of four datasets, the results of our method are very close to the best approach and stand in the second rank. Closer analysis shows that Discoverer performs better than PREUNN for both textual protocol and binary protocol datasets, and PREUNN has the lowest average results on all protocols, and on six datasets its Homogeneity score is about zero. We think the root cause of this poor result is the differences between our datasets and PREUNN datasets. Our results indicate that Discoverer is more effective for textual protocols than binary protocols and has the best results in two datasets and on average has the second rank. Furthermore, UPGMA-3 has good results (score>0.8) against six datasets. It is important to note that we did not normalize or balance our datasets in these experiments to avoid unintended biases.

Fig. 6 presents the V-Measure with $\beta = 0.1$ for these methods. In this figure, the CNNPRE method has the best performance in 8 of 12 datasets and has the highest V-Measure average on all datasets.

Table 2 shows the average Homogeneity score and V-Measure with three different β for all methods on all datasets.

As shown in Table 2 the proposed method (CNNPRE) has the highest average Homogeneity score and V-Measure for all datasets.

B. ABLATION ANALYSIS

Our second objective is to evaluate DenseNet-169, HDBSCAN, UMAP, and traffic features efficiency.

TABLE 2. Average Homogeneity score and V-Measure with $\beta = 0.1, 0.2, 0.4$ for all methods on all datasets.

Method	Homogeneity	V-M (0.1)	V-M (0.2)	V-M (0.4)
CNNPRE	0.87	0.75	0.70	0.62
Discoverer	0.59	0.54	0.52	0.52
PREUNN	0.19	0.20	0.21	0.24
K-Means	0.52	0.50	0.49	0.51
UPGMA-1	0.24	0.25	0.26	0.28
UPGMA-2	0.37	0.38	0.40	0.41
UPGMA-3	0.60	0.59	0.58	0.57

Table 3 presents the comparison of different well-known pre-trained models. According to the table, DenseNet-169 has the highest Homogeneity and V-Measure scores of all the models. It‘s worthwhile to note that ResNet-50 is also a suitable choice because it has the highest Homogeneity score for 5 protocols.

The results of comparisons between HDBSCAN and K-Means and UPGMA are presented in Fig. 7. In this experiment, we used K-Means with Elbow method and to gain the optimal results for UPGMA we tried three different values for the number of clusters, one of the values was the Elbow value for the K-Means method (UPGMA-E), the second value is the number of clusters from the HDBSCAN algorithms (UPGMA-H), and the third value is the average of these two values (UPGMA-A). In this figure, we reported the average score of these three values for UPGMA (UPGMA-AVG). It‘s worth mentioning that, in real-world scenarios, the analysts don‘t have any information about the target protocols, so they can‘t compare the results of different values and can‘t choose the best value for the number of clusters.

HDBSCAN has better results than the K-Means and UPGMA (Average) in the clustering step on all datasets, as shown in Fig. 7.

Table 4 presents the effects of using DenseNet-169, UMAP, and traffic features in our method.

In this table, the Binary row shows the results of using HDBSCAN on first 1024 bytes of packets (without any feature extraction), Binary-UDL shows the results of HDNSCAN algorithm on UMAP of Binary and after adding traffic features, DN169 shows the results of HDNSCAN algorithm on output of DenseNet-169 model, DN169-U illustrates the results of HDNSCAN algorithm on output of UMAP on DenseNet-169 results, DN169-UD illustrates the effects of adding direction to DN169-U, and DN169-UDL represents the result of the proposed method. This table also indicates that using UMAP improves the results by about 7.5% and adding traffic features (direction and packet length) on average can improve the Homogeneity score of message type identification by about 2.3%. In addition, using DenseNet-169 (conversion of packets into images and feature extraction) results in better Homogeneity score by approximately 4.8% over using raw packets. It’s important to note that these results shows that the overall performance of the proposed method is not biased by the traffic features, and CNNPRE has perfect generalization ability.

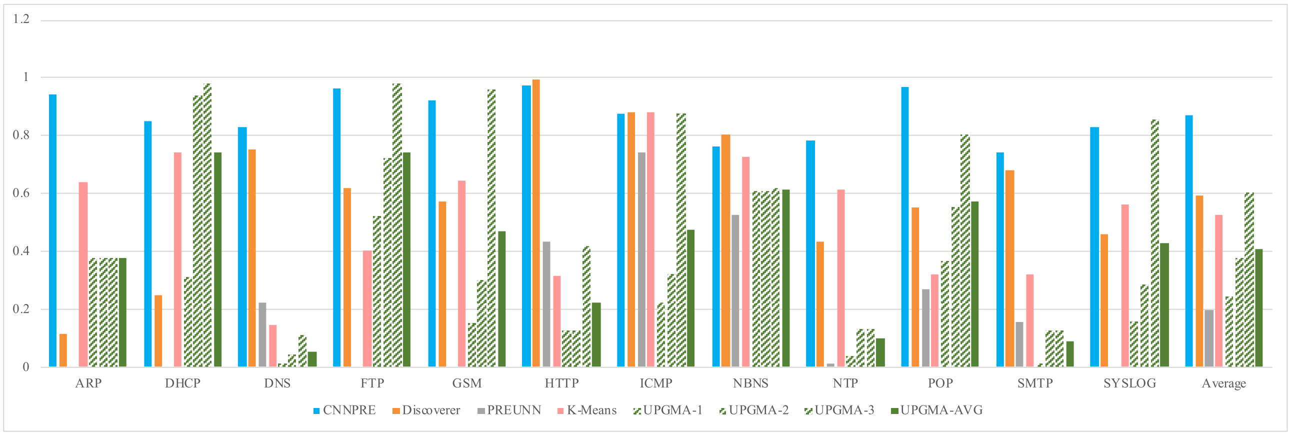

FIGURE 5. Homogeneity scores of the proposed method (CNNPRE), Discoverer, PREUNN, K-Means, and UPGMA on all datasets.

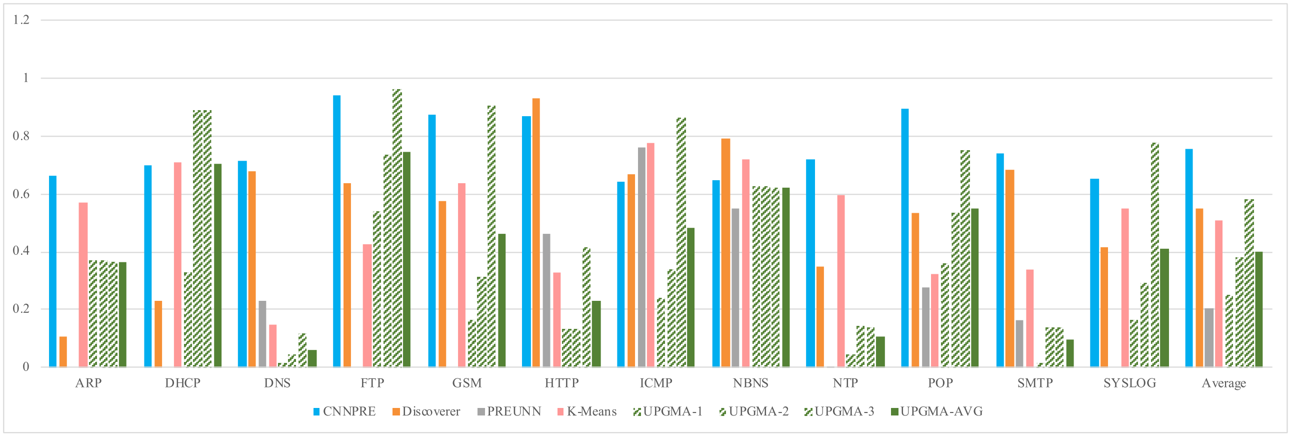

FIGURE 6. V-Measure of the proposed method (CNNPRE), Discoverer, PREUNN, K-Means, and UPGMA on all datasets.

TABLE 3. Homogeneity score and V-Measure ($\beta = 0.1$) of well-known pre-trained models.

	ResNet-50		VGG-16		VGG-19		DenseNet-121		DenseNet-169		DenseNet-201		Xception		Inception-V3	
	H	V-M (0.1)	H	V-M (0.1)	H	V-M (0.1)	H	V-M (0.1)	H	V-M (0.1)	H	V-M (0.1)	H	V-M (0.1)	H	V-M (0.1)
ARP	0.62	0.43	0.61	0.43	0.65	0.46	0.92	0.65	0.94	0.66	0.77	0.53	0.76	0.53	0.86	0.60
DHCP	0.89	0.72	0.79	0.65	0.81	0.66	0.87	0.70	0.85	0.70	0.83	0.68	0.86	0.70	0.85	0.69
DNS	0.86	0.73	0.85	0.73	0.83	0.72	0.84	0.72	0.83	0.72	0.84	0.72	0.83	0.71	0.85	0.73
FTP	0.97	0.94	0.98	0.95	0.97	0.95	0.93	0.91	0.96	0.94	0.97	0.95	0.97	0.94	0.95	0.93
HTTP	0.94	0.84	0.95	0.85	0.94	0.84	0.96	0.85	0.97	0.87	0.96	0.86	0.96	0.86	0.94	0.84
GSM	0.89	0.85	0.95	0.90	0.93	0.89	0.86	0.82	0.92	0.87	0.90	0.86	0.91	0.86	0.89	0.84
ICMP	0.93	0.77	0.84	0.62	0.91	0.75	0.93	0.70	0.85	0.64	0.93	0.77	0.93	0.77	0.93	0.77
NBNS	0.79	0.67	0.69	0.60	0.71	0.61	0.79	0.67	0.76	0.65	0.79	0.68	0.74	0.64	0.67	0.58
NTP	0.39	0.38	0.68	0.63	0.48	0.41	0.46	0.45	0.78	0.72	0.40	0.40	0.39	0.39	0.37	0.38
POP	0.92	0.85	0.91	0.84	0.91	0.84	0.98	0.90	0.97	0.89	0.96	0.88	0.92	0.84	0.88	0.82
SMTP	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.74	0.74	0.74	0.74	0.75	0.75	0.75	0.75
SYSLOG	0.87	0.68	0.89	0.70	0.84	0.66	0.80	0.63	0.83	0.65	0.83	0.66	0.91	0.71	0.84	0.66
Average	0.82	0.72	0.82	0.72	0.81	0.71	0.84	0.73	0.87	0.75	0.83	0.73	0.83	0.73	0.82	0.72

For the purpose of evaluating the performance of DenseNet-169 as the feature extractor in CNNPRE, we presented the results of using auto-encoders (AE-UDL) and n-grams (NG-UDL) instead of DenseNet-169. The parameters of PREUNN were used for feature extraction with AE, and 8 bits as grams were used for NG. As shown in Table 3, CNNPRE outperforms auto-encoder and n-gram by about 10%.

It's worth mentioning that in some datasets, we consider broadcast/non-broadcast as direction. Furthermore, using direction as a traffic feature for protocols with each packet having the same direction, like SYSLOG (sending data

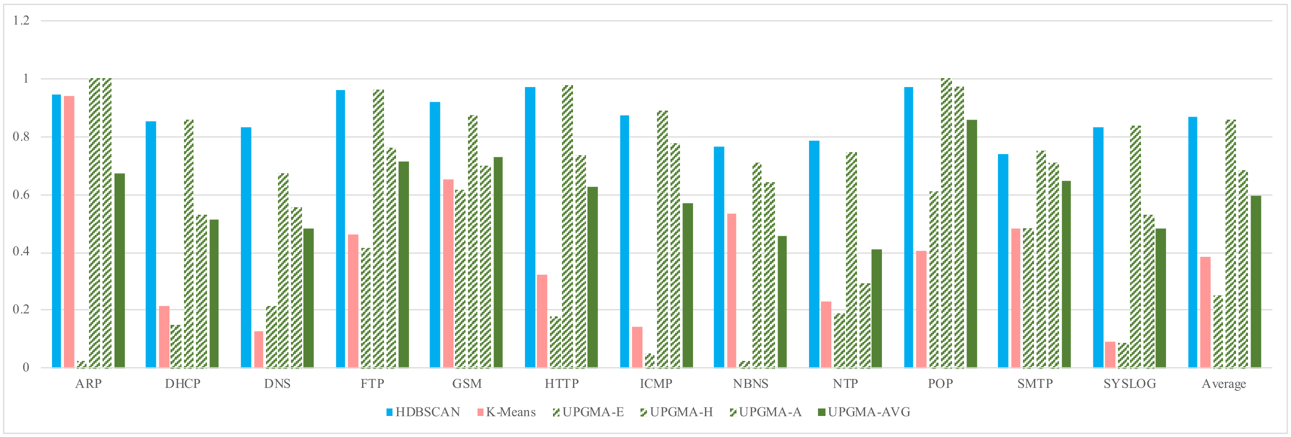

FIGURE 7. Comparison of Homogeneity score of HDBSCAN, K-Means, and UPGMA clustering methods on all datasets.

TABLE 4. Effects of CNN, UMAP, and traffic features on the proposed method.

	ARP	DHCP	DNS	FTP	GSM	HTTP	ICMP	NBNS	NTP	POP	SMTP	SYSLOG	Average
Binary	0.51	0.84	0.84	0.95	0.96	0.96	0.88	0.60	0.30	0.86	0.90	0.93	0.79
Binary-UDL	0.55	0.84	0.88	0.96	0.93	0.90	0.86	0.50	0.77	0.94	0.93	0.90	0.83
DN169	0.99	0.76	0.68	0.94	0.95	0.97	0.84	0.67	0.14	0.94	0.75	0.84	0.79
DN169-U	0.94	0.82	0.67	0.96	0.92	0.97	0.85	0.77	0.74	0.97	0.74	0.83	0.85
DN169-UD	0.94	0.82	0.83	0.96	0.92	0.97	0.85	0.77	0.74	0.97	0.74	0.83	0.86
DN169-UDL	0.94	0.85	0.83	0.96	0.92	0.97	0.85	0.76	0.78	0.97	0.74	0.83	0.87
AE-UDL	0.52	0.87	0.87	0.97	0.85	0.92	0.94	0.53	0.35	0.98	0.98	0.75	0.79
NG-UDL	0.12	0.58	0.90	0.99	0.88	0.96	0.87	0.68	0.67	0.98	1.00	0.97	0.80

from different nodes to one SYSLOG server), or protocols without a central server, like ICMP and NTP, is not effective. Additionally, in protocols with fixed lengths like GSM, packet length does not have any effect on the message type identification process.

C. PRACTICAL APPLICATION

This section provides two examples of CNNPRE's application for identifying unknown protocol message types.

In Figs. 8 and 9, CNNPRE results are shown for GSM and POP protocols, respectively. In these figures, the inner circles indicate the cluster labels and the outer circles indicate the true message types.

As shown in Fig. 8, in reverse engineering GSM protocol with CNNPRE, eight resultant clusters only contain one type of message, thus analysts can perform the next steps of the PRE process on these results with high accuracy and determine the types of BCCH, PCH, CBCH, SACCH, and SDCCH/4 messages.

Fig. 9 illustrates that when reverse engineering POP protocol with CNNPRE, nine resultant clusters contain only one type of message, allowing analysts to perform the next steps of PRE process with high accuracy and infer OK, QUIT, USER, PASS, and LIST messages.

The poorest Homogeneity score of our method was on NBNS and SMTP datasets, which were 76% and 74%, respectively. For the NBNS dataset, all methods have almost

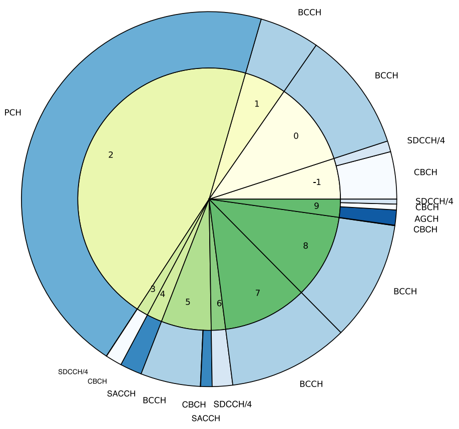

FIGURE 8. Visual representation of CNNPRE results in the GSM protocol.

the same performance. We believe that the cause of these results is the unbalanced distribution of message types in the NBNS protocol. For the SMTP protocol, we think the close similarity of message types is the main cause of our method's suboptimal performance. It is important to note that we did not perform any parameter tuning to improve the results of our method, and we used the same parameters across all datasets. This means that, in practical usage of our method, the analyst can adjust different parameters following the result. For example, suppose the number of resulting message types is very high. In that case,

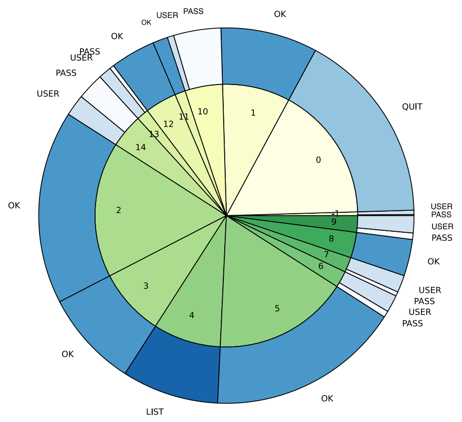

FIGURE 9. Visual representation of CNNPRE results in the POP protocol.

the analyst can increase the minimum cluster size in the clustering step to reduce the number of clusters created and vice-versa.

Finally, the extensive coverage of our 12 datasets across multiple OSI layers with diverse protocols and message types has demonstrated the robustness of our proposed approach to generalize for all kinds of unknown protocols, including binary and textual protocols, which is crucial for practical deployment and has not been adequately addressed in prior works. Our method also has the advantage of being independent of dataset size and functional for large and small datasets.

VI. CONSLUSION

In this study, we proposed a message type identification mechanism to group similar messages of both textual and binary protocols. Our proposed method CNNPRE, utilizing a combination of traffic payload and traffic features with pre-trained CNNs with weights drawn from the ImageNet dataset, performs the best on 9 out of 12 datasets and has the highest average Homogeneity score > 0.87 and V-Measure ($\beta = 0.1$) > 0.75 on all datasets, which is an excellent starting point for the PRE process to derive message format and protocol semantics. The results demonstrate the robustness and generalizability of the proposed approach.

Since our study did not focus primarily on the computational cost, more work can be done to optimize our code. It will enable us to make more detailed comparisons and analyses of the computational costs of our method and other existing methods in the future. Another future work could be a method for fine-tuning resultant message types, including merging similar types and dividing large clusters into smaller ones.

Further work could also include the use of more statistical features of traffic and other approaches for the extraction of payload characteristics. As a final note, our efforts have only touched the surface of PRE, and in the future, we hope to build upon our PRE framework to explore the application of advanced methods in other areas of PRE.

REFERENCES

[1] W. Cui, J. Kannan, and H. J. Wang, "Discoverer: Automatic protocol reverse engineering from network traces," in *Proc. USENIX Secur. Symp.*, 2007, pp. 1–14.

[2] J. Duchêne, C. L. Guernic, E. Alata, V. Nicomette, and M. Kaâniche, "State of the art of network protocol reverse engineering tools," *J. Comput. Virol. Hacking Techn.*, vol. 14, no. 1, pp. 53–68, Feb. 2018.

[3] J. Caballero, H. Yin, Z. Liang, and D. Song, "Polyglot: Automatic extraction of protocol message format using dynamic binary analysis," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Oct. 2007, pp. 317–329.

[4] C. Y. Cho, D. Babic, P. Poosankam, K. Z. Chen, E. X. Wu, and D. Song, "Mace: Model-inference-assisted concolic exploration for protocol and vulnerability discovery," in *Proc. USENIX Secur. Symp.*, vol. 139, 2011, pp. 1–16.

[5] P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol specification extraction," in *Proc. 30th IEEE Symp. Secur. Privacy*, May 2009, pp. 110–125.

[6] Y. Ye, Z. Zhang, F. Wang, X. Zhang, and D. Xu, "NetPlier: Probabilistic network protocol reverse engineering from message traces," in *Proc. NDSS*, 2021, pp. 1–18.

[7] T. Krueger, H. Gascon, N. Krämer, and K. Rieck, "Learning stateful models for network honeypots," in *Proc. 5th ACM Workshop Secur. Artif. Intell.*, Oct. 2012, pp. 37–48.

[8] J. Antunes, N. Neves, and P. Verissimo, "Reverse engineering of protocols from network traces," in *Proc. 18th Work. Conf. Reverse Eng.*, 2011, pp. 169–178.

[9] G. Bossert, F. Guihéry, and G. Hiet, "Towards automated protocol reverse engineering using semantic information," in *Proc. 9th ACM Symp. Inf., Comput. Commun. Secur.*, Jun. 2014, pp. 51–62.

[10] Y.-H. Goo, K.-S. Shim, M.-S. Lee, and M.-S. Kim, "Protocol specification extraction based on contiguous sequential pattern algorithm," *IEEE Access*, vol. 7, pp. 36057–36074, 2019.

[11] S. Kleber, L. Maile, and F. Kargl, "Survey of protocol reverse engineering algorithms: Decomposition of tools for static traffic analysis," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 526–561, 1st Quart., 2019.

[12] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, Mar. 1970.

[13] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *J. Comput. Biol.*, vol. 1, no. 4, pp. 337–348, 1994.

[14] R. R. Sokal, "A statiscal method for evaluating systematic relationships," *Univ. Kansas Sci. Bull.*, vol. 38, pp. 1409–1438, Mar. 1958.

[15] K. Millar, A. Cheng, H. G. Chew, and C.-C. Lim, "Using convolutional neural networks for classifying malicious network traffic," in *Deep Learning Applications for Cyber Security*. Cham, Switzerland: Springer, 2019, pp. 103–126.

[16] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.

[17] J. Krupski, W. Graniszewski, and M. Iwanowski, "Data transformation schemes for CNN-based network traffic analysis: A survey," *Electronics*, vol. 10, no. 16, p. 2042, Aug. 2021.

[18] L. Yang, S. Hanneke, and J. Carbonell, "A theory of transfer learning with applications to active learning," *Mach. Learn.*, vol. 90, no. 2, pp. 161–189, Feb. 2013.

[19] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning—ICANN 2018*. Rhodes, Greece: Springer, Oct. 2018, pp. 270–279.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[26] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[30] M. Shevertalov and S. Mancoridis, "A reverse engineering tool for extracting protocols of networked applications," in *Proc. 14th Work. Conf. Reverse Eng. (WCRE)*, Oct. 2007, pp. 229–238.
[31] Y. Wang, Z. Zhang, D. Yao, B. Qu, and L. Guo, "Inferring protocol state machine from network traces: A probabilistic approach," in *Applied Cryptography and Network Security*. Nerja, Spain: Springer, Jun. 2011, pp. 1–18.
[32] M. A. Beddoe, "Network protocol analysis using bioinformatics algorithms," *Toorcon*, vol. 26, no. 6, pp. 1095–1098, 2004.
[33] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.
[34] W. Cui, V. Paxson, N. Weaver, and R. H. Katz, "Protocol-independent adaptive replay of application dialog," in *Proc. NDSS*, 2006, pp. 1–15.
[35] Y. Wang, X. Yun, M. Z. Shafiq, L. Wang, A. X. Liu, Z. Zhang, D. Yao, Y. Zhang, and L. Guo, "A semantics aware approach to automated reverse engineering unknown protocols," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2012, pp. 1–10.
[36] P. Lin, Z. Hong, L. Wu, Y. Li, and Z. Zhou, "Protocol format extraction based on an improved CFSM algorithm," *China Commun.*, vol. 17, no. 11, pp. 156–180, Nov. 2020.
[37] C. Yang, C. Fu, Y. Qian, Y. Hong, G. Feng, and L. Han, "Deep learning-based reverse method of binary protocol," in *Security and Privacy in Digital Economy*. Quzhou, China: Springer, Oct./Nov. 2020, pp. 606–624.
[38] Y. Wang, B. Bai, X. Hei, L. Zhu, and W. Ji, "An unknown protocol syntax analysis method based on convolutional neural network," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 5, May 2021, Art. no. e3922.
[39] R. Zhao and Z. Liu, "Analysis of private industrial control protocol format based on LSTM-FCN model," in *Proc. Int. Conf. Aviation Saf. Inf. Technol.*, Oct. 2020, pp. 330–335.
[40] B. Ning, X. Zong, K. He, and L. Lian, "PREIUD: An industrial control protocols reverse engineering tool based on unsupervised learning and deep neural network methods," *Symmetry*, vol. 15, no. 3, p. 706, Mar. 2023.
[41] V. Kiechle, M. Börsig, S. Nitzsche, I. Baumgart, and J. Becker, "PREUNN: Protocol reverse engineering using neural networks," in *Proc. 8th Int. Conf. Inf. Syst. Secur. Privacy*, 2022, pp. 345–356.
[42] C. Leita, K. Mermoud, and M. Dacier, "ScriptGen: An automated script generation tool for honeyd," in *Proc. 21st Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2005, p. 214.
[43] I. Bermudez, A. Tongaonkar, M. Iliofotou, M. Mellia, and M. M. Munafò, "Towards automatic protocol field inference," *Comput. Commun.*, vol. 84, pp. 40–51, Jun. 2016.
[44] Z. Lin, X. Jiang, D. Xu, and X. Zhang, "Automatic protocol format reverse engineering through context-aware monitored execution," in *Proc. NDSS*, vol. 8, 2008, pp. 1–15.
[45] W. Cui, M. Peinado, K. Chen, H. J. Wang, and L. Irun-Briz, "Tupni: Automatic reverse engineering of input formats," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, Oct. 2008, pp. 391–402.
[46] Z. Wang, X. Jiang, W. Cui, X. Wang, and M. Grace, "Reformat: Automatic reverse engineering of encrypted messages," in *Computer Security—ESORICS 2009*. Saint-Malo, France: Springer, Sep. 2009, pp. 200–215.
[47] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018, *arXiv:1802.03426*.
[48] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *J. Open Source Softw.*, vol. 2, no. 11, p. 205, Mar. 2017.
[49] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn. (EMNLP-CoNLL)*, 2007, pp. 410–420.

JAVAD GARSHASBI (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree in computer science with the University of Tehran, Iran. He is also a Research Assistant with the Information Theory and Coding Laboratory, University of Tehran. His research interests include protocol reverse engineering, information theory and coding, cyber security, and the applications of artificial intelligence in these fields.

MEHDI TEIMOURI (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of Tehran, Iran, in 2003, 2005, and 2009, respectively. He is currently an Associate Professor with the Department of Data Science and Technology, University of Tehran, where he is also the Head of the Information Theory and Coding Laboratory. His current research interests are in the area of information theory and coding, with an emphasis on applications of artificial intelligence in this field.