IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 September 2022, accepted 9 October 2022, date of publication 17 October 2022, date of current version 24 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3215132

RESEARCH ARTICLE

Detecting and Locating Storage-Based Covert Channels in Internet Protocol Version 6

ARTI DUA[1], VINITA JINDAL[2], AND PUNAM BEDI[3], (Senior Member, IEEE)
[1]Bhaskaracharya College of Applied Sciences, University of Delhi, New Delhi, Delhi 110075, India
[2]Keshav Mahavidyalaya, University of Delhi, New Delhi, Delhi 110034, India
[3]Department of Computer Science, University of Delhi, New Delhi, Delhi 110007, India

Corresponding author: Arti Dua (arti.batra@bcas.du.ac.in)

ABSTRACT Increased usage of the Internet has risen the demand for more IP addresses across the globe resulting in replacement of IPv4 by IPv6 protocol. Hence, security of IPv6 has become a vital area of research. One of the serious threats to Internet security is the presence of Network Covert Channels (NCCs) that provide substantial aid for performing covered communications like exchanging secret data and/or exfiltrating secret information from the organizations. To detect such malicious activities, there is an urgent requirement to develop and deploy efficient detection mechanisms in real-time networks. Further, to decode the hidden communications, there is an additional need to identify the location of covert data. Thus, this paper proposes a system for detecting and locating storage-based NCC(s) in IPv6 using Deep Neural Network (DNN) and One-vs-Rest (OvR) technique with Support Vector Machine (SVM). The proposed system is a two-layered system. Layer 1 detects an IPv6 packet as a normal/covert packet. Layer 2 locates the storage area of secret data in the covert packets detected at Layer 1. For experimentation, a dataset of normal and covert IPv6 packets was created using CAIDA's dataset and pcapStego tool. Experiments were conducted to select the appropriate classifiers at both layers of the proposed system. With DNN and OvR SVM selected as the classifiers at Layer 1 and Layer 2 respectively, the proposed system locates covert data in IPv6 packets with an Accuracy of 99.7% and an average prediction time of 0.0719 seconds per covert sample, making it suitable for real-time deployment.

INDEX TERMS Cybersecurity, deep neural network, internet protocol version 6, machine learning, network covert channel detection, one-vs-rest, support vector machine.

I. INTRODUCTION

With rapidly evolving technology, the Internet has become an essential part of our daily lives. With the worldwide lockdowns imposed as a result of Covid-19, a majority of organizations shifted to the work-from-home option for their employees thereby increasing the dependence on the Internet tremendously. Along with this, the various cyber-attacks like online fraud, scams, intrusions, and security breaches have also increased excessively [1]. The use of stegomalware, which concerns the transfer of malware through Information Hiding techniques has also risen significantly in the last decade [2], [3], [4]. The stegomalware exploit an innocent-looking cover medium to carry malicious data. Examples of such cover media include images, documents, videos, audios, network traffic flows, etc. The use of network traffic flows for hiding secret information is implemented with the help of Network Covert Channels (NCCs). Based on the technique used to hide secret data in traffic flows, the NCCs can be classified into two sub-categories [5]:

1. Storage-Based Network Covert Channels: These NCCs utilize the reserved or unutilized storage area of the network protocol header or the payload part of a network protocol [6].
2. Timing-Based Network Covert Channels: The timing-based NCCs utilize the inter-packet timing relations or delays for encoding and transmitting secret information [7].

The three characteristics that measure the efficiency of an NCC are the capacity of the NCC, the undetectability of

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

the hidden data, and the robustness of the technique used to develop the covert channel [8]. In terms of the capacity of a network covert channel, the covert channels developed using more frequently used network protocols generally provide higher covert bandwidth per unit time because of the higher availability of the respective protocol packets over the network(s). With Internet Protocol version 6 (IPv6) rapidly replacing Internet Protocol version 4 (IPv4), the use of IPv6 over the Internet is increasing every day [9]. With the increase in the availability of IPv6 packets over the Internet, this network layer protocol can become a good candidate for the development of covert channels. Some covert channels targeting IPv6 protocol have already been proposed by researchers in [10], [11], and [12]. Further, in [12], the authors also investigated and showed that some well-known IDS tools like Zeek (bro) and Suricata are not effective to detect covert channel threats in IPv6. As countermeasures, few methods have been proposed by researchers to detect the existence of storage-based NCCs in IPv6. The detection approaches in [13] and [14] use statistical analysis of the IPv6 packet's header field using eBPF (extended Berkley Packet Filter) for IPv6-based covert channel detection. In these techniques, authors make use of changing number of bins (evaluated for a fixed time in IPv6 network traffic) as a metric to detect the existence of storage-based NCCs in IPv6. The limitation of such approaches as discussed by the authors in their work [13], [14] are as follows.

Firstly, these approaches are not suitable for detecting short-length covert communication which is due to the small size of covert message transfer. Consequently, the change in the number of bins is also very small which might not be detected by this approach. Secondly, with eBPF, which is a lightweight framework, more granular values of the number of bins give better results but need higher memory resources in the node running the eBPF filter.

Such limitations offered a gap for future research in covert channel detection in IPv6. Thus, the first motivation for this work is *the need to overcome the limitations which are encountered due to the use of eBPF for covert channel detection*.

The use of existing traditional techniques faces some challenges in the network traffic analysis [15]. The major concerns include accuracy and effective processing of real-time big data. Additionally, the real-time network traffic shows complex behavior that happens because of various factors like network diversity etc. Nowadays Machine Learning (ML) and Deep Learning (DL) are being used to solve many crucial problems by analyzing and identifying the patterns hidden in the data. These techniques are being used for efficient analysis of big data systems to recognize hidden and complex patterns. This has motivated the researchers working in the field of networking to apply DL and ML techniques for Network applications like Traffic classification and Prediction. Chourib [16] suggested the use of DL and ML techniques like SVM, DNN, and KNN to identify covert channels in selected header fields of IPv4, ICMP, TCP, UDP, and DNS protocols. Recently, [17] and [18] proposed the detection of storage-based NCCs in IPv6 using Deep Neural Network (DNN) and Convolutional Neural Network (CNN) respectively. To the best of our knowledge, none of the existing research works target the locating of the storage area of secret data in the IPv6 protocol header which is an important area of research. For scenarios like that of Intelligence Agencies who wish to detect as well as decode the covert messages that are being transferred over a medium, identifying the location of covert data holds extreme importance. Thus, to overcome this gap, the second motivation for this work is *the need to design an efficient system that not only detects the presence of storage-based NCCs but also locates the storage area of secret data in the IPv6 header*.

With these motivations, we propose a system for detecting and locating storage-based covert channels in IPv6 using the Deep Neural Network and One-vs-Rest (OvR) technique with Support Vector Machine. The proposed system is implemented in two layers: Layer 1 and Layer 2. Layer 1 detects an IPv6 packet as a covert or a normal IPv6 packet. If a particular IPv6 packet is detected as a covert IPv6 packet at Layer 1, Layer 2 locates the storage area of secret data in the header of that packet.

Following are the key contributions of this paper:

1. This paper proposes a two-layered system for detecting and locating storage-based covert channels in IPv6.
2. For ensuring the accurate, efficient detection and locating of the storage area of storage-based covert channels in IPv6, extensive experimentation was done for the selection of classifiers for the proposed system.
3. For training and testing the classifiers used in the proposed system, the dataset consisting of normal and covert IPv6 packets was created using a benchmark dataset and an already existing IPv6 covert packet generation tool.

The structural organization of the rest of this paper is as follows. Section II briefly describes the Internet Protocol version 6, Deep Neural Network, One-vs-Rest technique, and Support Vector Machine used in the development of the proposed system, followed by the threat model considered in this work. Section III discusses the related research work in the area of development and detection of IPv6-based NCCs. Section IV describes the working of the proposed system. This is followed by Section V which explains the development, training, and testing phases along with the experiments conducted on the proposed system. Section VI presents the detailed results of experiments conducted with the proposed system. This is followed by Section VII which gives the conclusion.

II. BACKGROUND

This section provides a concise overview of Internet Protocol version 6, Deep Neural Network, One-vs-Rest technique, and the Support Vector Machine which are used in this paper followed by the threat model considered in this work.

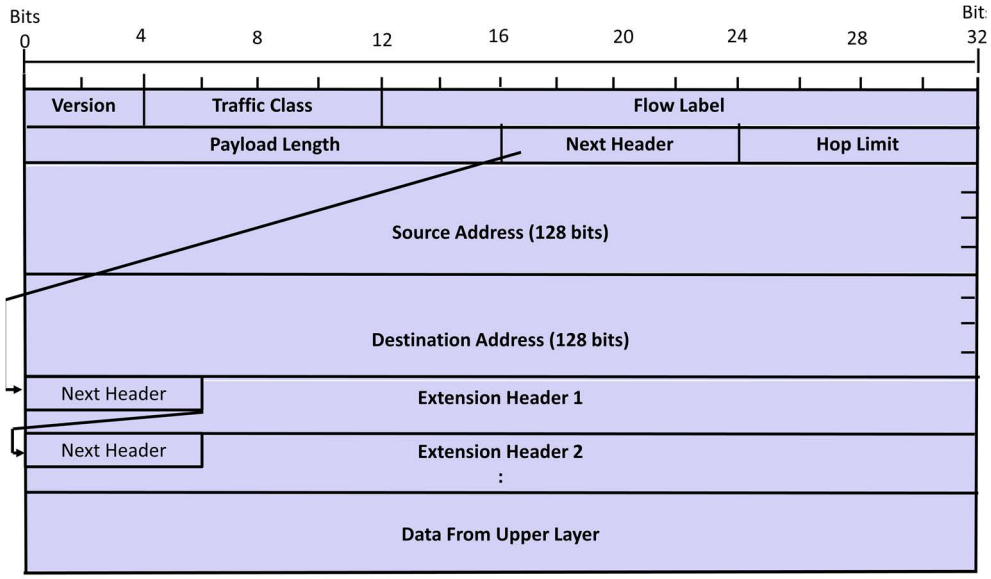

FIGURE 1. Structure of IPv6 header.

A. INTERNET PROTOCOL VERSION 6

The IPv6 which operates at the network layer, is rapidly replacing Internet Protocol version 4. To overcome the problem of depletion of 32 bits long IPv4 addresses, IPv6 was engineered by Internet Engineering Task Force in 1998. According to Google statistics, the IPv6 adoption rate has reached a value of 39.88% as of 31 July 2022 [9]. RFC 8200 [19] provides a detailed description of this protocol. The structure of the header part of this next-generation protocol is given in Fig. 1.

The first field of the Internet Protocol version 6 header is the Version field. It denotes the version of Internet Protocol being used. Its value is set to 6 for the IPv6 header. The subsequent field is the Traffic Class (TC) field which is 8 bits in size. Its value marks the priority or class of a packet. This value can be easily replaced with 8 bits of secret data. The following field is the Flow Label (FL) field which is 20 bits long. This field identifies the IPv6 packets belonging to a single flow and is used to mark the packets that require special handling from intermediate IPv6 routers. The details regarding the usage of this field are given in RFC 6437 [20]. Due to the use of pseudo-random values in this field, it stands as a good candidate for covert communication and provides a covert capacity of 20 bits per IPv6 packet. The subsequent field is the Payload Length (PL) field. The size of this field is 16 bits and it contains the total length of the payload (including the extension headers (if any) and the transport layer headers) carried by an IPv6 packet. The next field in the sequence is the Next Header (NH) field which is 8 bits long and identifies the next extension header or transport header attached to the base IPv6 header. This Next Header could be an extension header or in absence of an extension header, it could be a transport layer header such as a Transmission Control Protocol (TCP) header. The subsequent field is the Hop Limit (HL) field which is also 8 bits long. It gives the number of nodes that an IPv6 packet can pass through over the Internet before getting discarded. This field helps in avoiding unnecessary forwarding of IPv6 packets over the Internet thereby reducing congestion over the Internet. The association and use of any two specific values in this field can be used to denote a binary zero and one for covert communication providing a covert bandwidth of 1 bit per IPv6 packet. Following that are 128 bits long Source and Destination Address fields. Each IPv6 address is divided into eight blocks carrying sixteen bits each to identify the endpoints of an IPv6 packet. All the above-described fields compose an IPv6 base header.

Further, this base header may be linked to one or more Extension Headers (EHs). These extension headers contain extra information that may be needed by network devices such as intermediate routers, to decide how to forward or process the particular IPv6 packet. The commonly known EHs include headers like Hop-by-Hop EH, Destination Options EH, Routing EH, Authentication EH, Fragment EH, and Encapsulating Security Payload header. The transmission and processing details of these EHs are given in RFC 7045 [21].

It can be inferred that the Flow Label field, the Traffic Class field, and the Hop Limit field provide a covert bandwidth of 20 bits, 8 bits, and 1 bit per IPv6 packet and hence are most suitable for storage-based covert communication. In this paper, covert packets were created by injecting covert data in these three header fields of normal IPv6 packets.

B. DEEP NEURAL NETWORK

A Deep Neural network (DNN) is a special type of Artificial Neural Network that has more than one hidden layer in the network architecture. These networks are also called feedforward networks as the output received from the current

layer is fed as input to the subsequent layer for further processing. The architecture of a DNN comprises of three types of layers: a single input layer, some hidden layers (two or more in number), and a single output layer. The DNN has the capability to handle unstructured and non-linear data. The aim of a DNN is to approximate some function V. In a DNN classifier, $y = V(z)$ maps an input z to a class y. A DNN defines a mapping,

$$y = V(z; \theta) \tag{1}$$

and finds the value of the parameter 'θ' that results in the best function approximation [22]. The neurons in a DNN have a hierarchical organization similar to the human brain. The neurons at one layer pass the signal to the neurons at the next layer. If the signal received is greater than the threshold, the output is forwarded else it is ignored. The signal finally reaches the output layer where it provides the prediction based on the calculated probability values. Each layer is denoted by a function $V^{(i)}$, where i stands for the layer number. All these different functions combine together to give a chain of functions that denote a neural network function V as below:

$$V(z) = V^{(i)}(\ldots.(V^{(2)}(V^{(1)}(z)))\ldots.) \tag{2}$$

Here, z denotes the input, $V^{(1)}$ denotes the first layer, $V^{(2)}$ denotes the second layer and $V^{(i)}$ denotes the last layer or the output layer. The length of the chain denotes the number of layers used in the classifier and is also called the depth of the classifier.

Every layer in a DNN comprises several neurons, where each neuron has a function called the Activation Function. It is a kind of doorway that passes the signal to the next neuron connected to the current neuron in the forward direction. Each connection between any two neurons is assigned a weight. The weight values are initially random, but as the network gets trained iteratively, the weights are optimized to make the network produce correct predictions. During the training phase, the input data is provided to the network, and based on that the output prediction is made. On the basis of the predicted output and the actual output, the feedback is sent back to the DNN and the weights between the layers are adjusted. This process is called Backpropagation [22].

C. ONE-VS-REST TECHNIQUE

A binary classification model classifies a sample into one of the available two classes on which the model is trained whereas a multiclass classification model classifies a given sample into one of the 'n' numbers of classes on which a model is trained. All classification algorithms do not support multiclass classification directly such as Support Vector Machine, Logistic Regression (LR), etc. Thus, one way to use such classification algorithms for multiclass classification is One-vs-Rest (OvR) technique. With it, the multiclass training dataset is transformed into multiple binary datasets and the binary classification model is fit on each of the datasets, and predictions are made using the model that predicts with

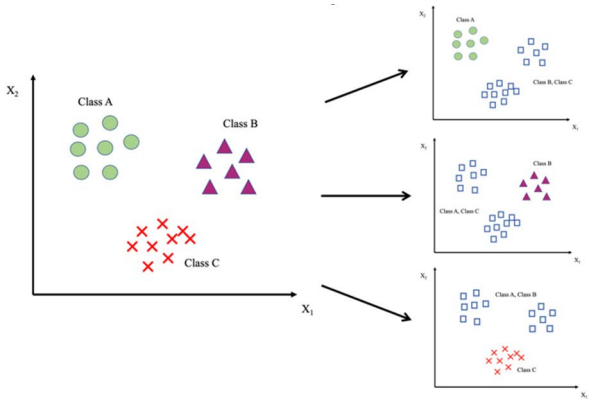

FIGURE 2. One-vs-rest technique.

the highest probability. For example, there is a classification problem to classify a sample as belonging to class 'A', 'B', or 'C'. This problem can be solved by dividing the training dataset into 3 binary datasets (as shown in Fig. 2), to train 3 classification models as follows:

Classifier 1: First binary classification model is trained on the binary dataset 'A' vs ['B', 'C']. Here a positive label is given to all the samples belonging to class 'A' and a negative label is given to samples belonging to class 'B' or class 'C'. Let this classifier be denoted by $C_\theta^{(1)}(X)$, i.e. classifier 1 trained on data X parameterized by θ.

Classifier 2: Second binary classification model is trained on the binary dataset 'B' vs ['A', 'C']. Here a positive label is given to all the samples belonging to class 'B' and a negative label is given to samples belonging to class 'A' or class 'C'. Let this classifier be denoted by $C_\theta^{(2)}(X)$.

Classifier 3: Third binary classification model is trained on the binary dataset 'C' vs ['A', 'B']. Here a positive label is given to all the samples belonging to class 'C' and a negative label is given to samples belonging to class 'A' or class 'B'. Let this classifier be denoted by $C_\theta^{(3)}(X)$.

The One-vs-Rest technique is a combination of all these classifiers which can be defined as:

$$C_\theta^{(i)}(X) = \text{Prob}(y = i|X; \theta); \text{ for } i = 1, 2, 3.$$

For the final prediction, each model predicts a probability score that denotes the class membership probability. The class which has the maximum probability is predicted as the final class denoted by

$$\max(C_\theta^{(i)}(X)); \text{ for } i = 1, 2, 3.$$

In this work, Classifier 1, Classifier 2, and Classifier 3 are implemented using the Support Vector Machine (SVM) algorithm. A brief description of SVM is given in the next subsection.

D. SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a type of supervised learning algorithm used majorly for solving classification problems. The main objective of this ML algorithm is to find

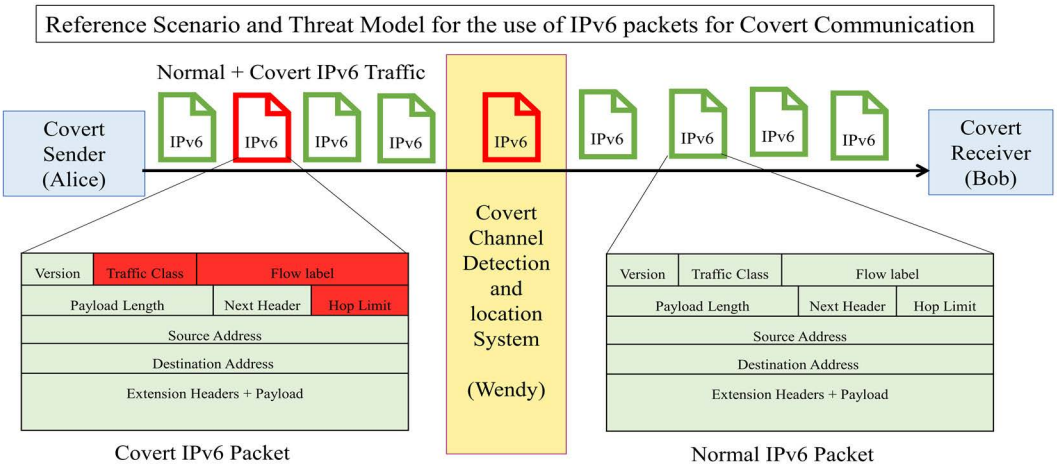

FIGURE 3. Reference scenario and threat model for covert communications.

the best hyperplane or decision boundary that can segregate n-dimensional space into label classes so that the new data samples can be classified into true categories. This algorithm selects the extreme vectors that help in finding the decision boundary. These extremes are called support vectors, and thus, this algorithm is called a Support Vector Machine.

For a linear SVM problem, let the training samples be denoted by (X_i, y_i) pairs, where X_i denotes the weighted feature vector corresponding to sample i and y_i denotes the class label of sample i, where $y_i \in [1,-1]$. In the SVM algorithm, the objective is to maximize the margin M $(= 2/||w||)$ between the data points and the hyperplane. i.e.,

$$\begin{aligned} &Maximize\ M = 2/||w||, \\ &\text{s.t. } y_i(wx_i + b) \geq 1, \text{ for all i} \end{aligned}$$

where w_i is a vector perpendicular to the hyperplane and is used to define the hyperplane's orientation and b denotes the hyperplane's position.

After solving this optimization problem for w and b, the score for test samples is calculated using the decision function:

$$y = wx + b \tag{3}$$

if $y \geq 1$, the input sample is predicted as a positive sample, and if $y \leq -1$, the input sample is predicted as a negative sample. Some of the advantages that SVM offers are: firstly, it is efficient in high-dimensional use cases. Secondly, it is efficient in terms of memory usage as it uses a subset of training samples in the decision function called support vectors. Thirdly, different kernel functions can be used as decision functions and it also allows to define the custom kernels.

E. THE THREAT MODEL

The existence of NCCs poses a serious threat to the security of cyberspace. The reference scenario for covert communications is the Prisoner's Problem as described by Simmons [23]. In the Prisoner's Problem, the two prisoners named Alice and Bob intend to escape the prison and thus, need to communicate secretly. All the messages that are being shared between Alice and Bob are being monitored by the warden Wendy. Hence, Alice and Bob need to communicate covertly in a way that Wendy misses to notice.

The Reference Scenario and the threat model for IPv6-based covert communications are given in Fig. 3, where Alice and Bob wish to communicate secretly by hiding secret data in any one of the three header fields (the Flow Label field, the Traffic Class field, and the Hop Limit field) of an IPv6 packet. Thus, in this paper, we propose a system that the warden Wendy can use to detect covert communication and fetch the hidden data as well. In the next section, the related research work done in the area of development and detection of NCCs is discussed.

III. LITERATURE REVIEW

Information hiding has been a favorite research topic for the past few decades. It can be implemented in computer networks using either Network Steganography or Network Covert Channels [24]. Network steganography is defined as a form of information hiding technique that utilizes network protocols as enablers of hidden communication [25]. ''Covert Channels are the channels not intended for information transfer at all'', as defined by Lampson [26]. Covert channels implemented using network traffic flows are termed as Network Covert Channels. Padlipsky et al. [27] first introduced the idea of the development of covert channels over network communication protocol in the year 1978. Handel et al. [28] further explored the possibility of the development of network covert channels in various network protocols operating at different layers of the OSI model.

In recent years, many popular protocols such as HTTP, RTP, SCTP, TCP, UDP, ICMP, IPv4, ARP, etc. were explored for vulnerabilities and the development of covert channels. Aniello et al. [29] used HTTP protocol for secure cross-layer

collaborative information transfer in mobile edge computing environments using the HTTP protocol. Saenger et al. [30] used fake silence RTP packets to inject secret data during the silence period for covert communication. Fraczek et al. [31] utilized the SCTP for the development of covert channels. They discussed nineteen different covert channels that can be developed using intra-protocol or inter-protocol methods. Giffin et al. [32] used the timestamp values in TCP packets to encode secret messages. Sabeti et al. [33] implemented two methods utilizing the lengths of consecutive UDP packets to communicate covertly. Ahsan et al. [34] used the Identification field of the IPv4 packet for communicating covertly. Another technique was developed by Bedi et al. [35] to hide data in the IPv4 protocol. In this, the authors proposed the development of a covert channel in an optional field named timestamp in an Internet Protocol version 4 packet. In this optional field, authors utilized the overflow field to carry secret data. Ray et al. [36] presented the use of the OS fingerprinting area inside ICMP echo request messages' payloads to communicate covertly. The last five bytes of the initial 8 bytes used for OS fingerprinting were used for covert data transfer. Bedi et al. [37] presented the use of the Address Resolution Protocol (ARP) for information hiding. In this, the authors utilized pre-decided seed value and ARP request messages to unused local IPv4 addresses to facilitate covert communication. Another covert channel using ARP was developed by Dua et al. [38] in which the secret data characters are transformed using the ASCII code values of that character and a random number generated using a pre-decided seed value between the covert message sender and the covert message receiver. They hid these transformed secret data character bits in the last octet of ARP broadcast Request messages.

Further, covert channel generation techniques using the IPv6 protocol have also been proposed by researchers. Bedi et al. [11] utilized the existence or absence of an IPv6 extension header in a fixed sequence to communicate a secret message over a LAN. The use of extension headers over the Internet is minimal, hence its use to carry covert data can easily be identified as an anomaly.

Lucena et al. [10] discussed the possibility of 22 covert channels that may be developed using the IPv6 protocol header. They proposed the use of fields such as Flow Label, Payload Length, Traffic Class, Next Header, Hop Limit, Source Address, and the use of already known extension headers for injecting covert data directly. The limitation of this study was that all these 22 covert channels were theoretically proposed by the authors without any experimental evaluation.

Mazurczyk et al. [12] conducted various experiments to check the actual feasibility of the channels proposed by Lucena et al. This was done to understand which IPv6 header fields can be utilized to transmit covert data over the wild Internet. These experiments downsized the covert channel capacity of an IPv6 header to a few header fields like the Flow Label field, the Traffic Class field, and the Hop Limit field.

With the possible existence of covert channels over IPv6, few researchers have worked on the detection of IPv6-based NCCs. Luca et al. [13] presented the use of code augmentation in eBPF inside the Linux kernel to collect the statistics of IPv6 header fields like Flow Label. Repetto et al. [14] used the BCC tool for running eBPF programs to obtain statistics about IPv6 header fields viz. the Flow Label field, the Traffic Class field, and the Hop Limit field. The underlying concept in both [13] and [14] was to share a common technique for analyzing packets' headers and gathering data for hidden data analysis. The authors inferred that abnormal changes in the statistical values of these header fields can raise an alarm about the existence of a covert channel. The limitation of the eBPF-based NCC detection mechanism is that it gives good accuracy with more granular values of the number of bins which consumes a large amount of resources. In addition, eBPF-based techniques cannot detect short covert communications. The system proposed in this paper aims to handle these limitations and can easily be scaled for other protocols in the future.

In recent years, Machine Learning as well as Deep Learning-based techniques have shown remarkable results in a variety of applications [39]. The same has been applied to detect covert channels in different network protocols. Salih et al. [40] used a Modified naïve Bayes classifier to detect ICMPv6 and IPv6-based covert channels with an accuracy percentage of 94.47%. They proposed a framework that uses Intelligent Heuristic Algorithm and modified C4.5 Decision Trees to create training data to detect hidden channels in the IPv6 network. Though the technique attained high accuracy, the creation of their dataset using the Intelligent Heuristic Algorithm is ambiguous. A. Senaid [41] applied a CNN-based approach to identify covert channels created in the code field of ICMPv6 protocol.

Zhao et al. [18] used a CNN to detect covert channels developed using Hop Limit and Source Address fields in the Internet Protocol version 6 header. They performed a deep packet inspection by transforming header fields of a network packet into a matrix containing the number of fields as rows of the matrix and the length of the longest field extracted as the number of columns of the matrix. Their technique reported an accuracy percentage of 100%. The gap in this work was that firstly the authors focused only on the detection of NCCs and secondly the detection technique was developed for covert channels developed with Hop Limit and Source Address fields only. Further, for the Source Address field, the possibility of covert channels was already ruled out in [12] due to the widespread protection against spoofing.

Dua et al. [17] proposed the detection of IPv6-based covert channels in Traffic Class and Flow Label fields using a Deep Neural Network architecture. They selected the DNN classifier for detecting covert channels in IPv6 after a rigorous comparison with other ML/DL techniques like CNN, LSTM, and SVM. They reported an accuracy percentage of 99.59% on a dataset size of a total of 16091 normal and covert

IPv6 packets. Their technique focused only on the detection of covert communication using two header fields of IPv6 namely Traffic Class and Flow Label. As their approach was giving good results for accuracy, the DNN classifier was selected for Layer 1 of the proposed system.

Further, to the best of our knowledge, none of the research works focuses on finding the location of the storage area of covert data in the IPv6 protocol header which is an important area of research. For an instance, identifying the location of covert data holds extreme importance for Intelligence Agencies who wish to detect as well as decode the covert messages that are being transferred over a medium. Thus, to this aim, a two-layered system that aims to detect and locate IPv6-based covert channels is proposed in this paper that uses ML/DL algorithms at both layers. Since a majority of the recent literature targets the detection of covert data in header fields like Traffic Class, Flow Label, and Hop limit [13], [14], [17] (which is due to its practical feasibility as covert channel carriers over the wild Internet [12]), these header fields are specifically considered for detection and locating of covert data in this work. The detailed working of the proposed system has been discussed in the next section.

IV. THE PROPOSED SYSTEM

A novel system for detecting and locating storage-based covert channels in IPv6 is proposed in this paper. Locating the covert storage area of covert data is equally important as the detection of covert IPv6 packets. This is because the knowledge of the exact location of the hidden data is needed to perform further processing like decoding of the hidden message. The proposed system has a two-layered architecture that detects and locates storage-based covert channels in IPv6 using DNN and One-vs-Rest technique with SVM. Layer 1 segregates covert IPv6 packets and normal IPv6 packets using the DNN classifier. The IPv6 packets detected as covert packets at Layer 1 are forwarded to Layer 2 for further classification to identify the location of the covert data.

Layer 2 performs multiclass classification to categorize the covert IPv6 packets into their respective covert storage area classes using the One-vs-Rest technique with SVM classifiers.

The following assumptions were made for this work:

1. Due to the unavailability of a benchmark dataset, a dataset was created, containing normal and covert packets. The normal packets were obtained from CAIDA's dataset (Anonymized Internet Traces 2019) [42] containing original IPv6 traces and covert IPv6 packets were generated using the pcapStego tool [43].
2. In this work, it is assumed that storage-based covert channels in IPv6 exist only in single header fields of IPv6, i.e. at any given time, for a covert IPv6 packet the covert data is stored in any one of the 3 fields viz. Flow Label, Traffic Class, or Hop Limit.

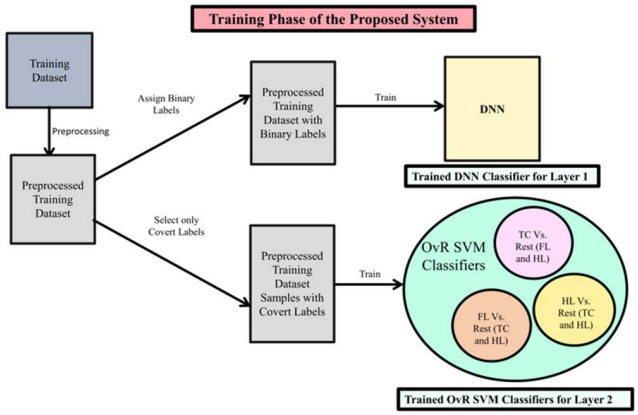

FIGURE 4. Training phase of the proposed system.

The training and testing of the proposed system were performed with the created dataset. Fig. 4 shows the training phase of the proposed system.

Before the training phase, the training dataset is preprocessed to quantize and standardize the data. Quantization transforms the categorical data values into numerical data values. This is done by giving a unique value to each category of a feature. After that, Standardization on the dataset is performed using the *standardscalar()* function of *sklearn* library [44].

The preprocessed training dataset is then used to train the DNN and OvR SVM classifiers used by the proposed system at the two layers. At Layer 1, the DNN binary classifier is trained on the preprocessed training dataset after assigning binary class labels (0 for normal IPv6 packet and 1 for covert IPv6 packet) to all the samples present in the training dataset. At Layer 2, three OvR classifiers with SVM are trained on only the covert samples of the preprocessed training dataset to perform multiclass classification. This locates the storage area of covert data in the IPv6 header of the packets filtered at Layer 1. Fig. 5 shows the testing phase of the proposed system.

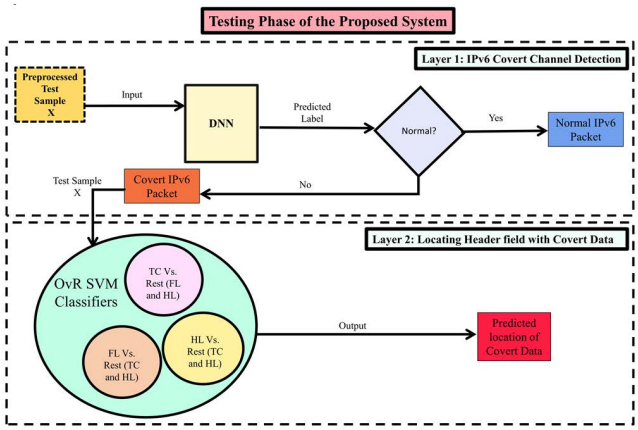

FIGURE 5. Testing phase of the proposed system.

Input: Test Sample 'X'
Output: Predicted Class Label for test sample 'X'

#Binary Classification at Layer 1:
1. Perform Binary classification on sample 'X' using DNN.
2. Assign the class label predicted by DNN as O_x.
3. If O_x denotes Normal class:
   a. O_x is classified as Normal IPv6 packet.
   else if O_x denotes Covert class :
   #Pass sample 'X' to Layer 2 for Multiclass Classification.
   a. Pass Sample 'X' to OvR SVM classifiers to predict the storage area of covert data in IPv6 packet.
   b. Assign the class label predicted by OvR SVM as O_x.
4. O_x is the final predicted label for sample 'X'.
5. End.

FIGURE 6. Algorithm for testing phase of the proposed system.

During the testing phase of the proposed system, each test sample X is initially passed through Layer 1 of the proposed system. If sample X is classified as a normal IPv6 packet by Layer 1 then this packet sample does not undergo any further processing. However, if Layer 1 predicts X as a covert IPv6 packet, X is further passed to Layer 2 of the proposed system for identifying the storage area of covert data. In Layer 2, X is classified using the OvR technique with SVM classifiers to identify the covert storage area in the covert IPv6 packet. Fig. 6 describes the algorithm used for testing the proposed system.

The DNN classifier was selected for performing binary classification at Layer 1. The hyperparameters used for training the DNN are as follows. It consisted of 3 hidden layers having 24-12-6 neurons at the respective layers followed by an additional dropout layer with a dropout rate of 0.2 to reduce overfitting. The learning rate of 0.01 was fixed with a batch size of 32. The DNN with this configuration was trained for 30 epochs. For implementing the One-vs-Rest technique at Layer 2, SVM classifiers using Radial Basis Function (RBF) kernel were used.

The next section presents the details of experiments conducted for the development of the proposed system.

V. EXPERIMENTAL STUDY

The proposed system was developed on a Windows 10 OS with a 2.20 GHz Intel Core i7 processor and 8 GB RAM. Python version 3.9.12 was used for the implementation of the proposed system. The development of the proposed system consisted of the creation of a dataset, preprocessing of the dataset, training, and testing of the classifiers at Layer 1 and Layer 2. The following sub-section describes the process of creation of the training and testing dataset.

A. DATASET

The dataset used for training and testing of the proposed system contained normal IPv6 packets obtained from CAIDA's dataset (Anonymized Internet Traces 2019) [42] and covert IPv6 packets generated using the pcapStego tool [43]. A set of 150000 packets were randomly selected as normal IPv6 packets from the CAIDA's (Anonymized Internet Traces 2019) dataset in the form of .pcap file.

For creating covert IPv6 packets, the pcapStego tool was used. This tool injects secret data in either of Traffic Class, Flow Label, or Hop Limit field of IPv6 packets. It takes as input a.pcap file that contains normal IPv6 flows (a flow contains one or more network packets that are uniquely identified by {Source IPv6 Address, Destination IPv6 Address, Transport Layer Protocol, Source Port Number, Destination Port Number} tuple). The output of this tool is a pcap file containing covert data packets.

For generating covert IPv6 packets, 24 large flows were randomly selected from CAIDA's (Anonymized Internet Traces 2019) dataset. The pcapStego tool used these flows to inject secret data in the Traffic Class field of IPv6 packets making each packet carry 8 bits of secret data. Similarly, pcapStego tool was used to inject secret data separately in Flow Label and Hop Limit field of IPv6 packets where each packet carried 20 bits of secret data and 1 bit of secret data per packet respectively. Finally, a total of 150000 covert IPv6 packets were obtained, out of which 50000 packets contained secret data in the Traffic Class field, 50000 packets contained secret data in the Hop Limit field and 50000 packets contained secret data in the Flow Label field.

Each IPv6 packet sample was labeled with one of the four classes: Normal, Traffic Class-based covert packet, Flow Label-based covert packet, and Hop Limit-based covert packet. The final combined IPv6 packets dataset was then created by combining the normal IPv6 packets and IPv6 packets carrying secret data in the Traffic Class field, the Hop Limit field, and the Flow Label field. The combined IPv6 packets dataset contained 300000 IPv6 packets in all. The process of the creation of the dataset is shown in Fig. 7.

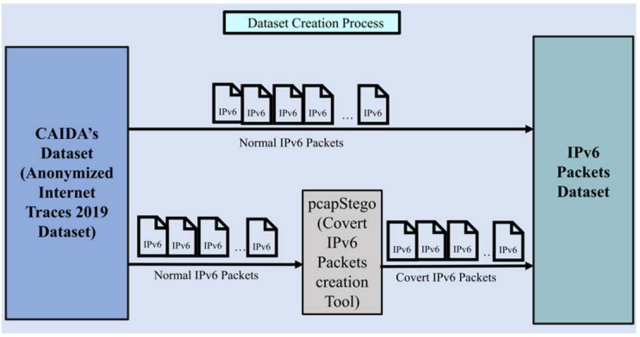

FIGURE 7. Dataset creation process.

Next, the relevant header fields of these IPv6 packets were extracted from the obtained pcap files (containing normal

and covert IPv6 packets) using Wireshark [45] which was further converted into a corresponding csv file. The resulting csv file consisted of the following header field attributes: Source IPv6 Address, Destination IPv6 Address, Flow Label, Payload Length, Traffic Class, Hop Limit, Next Header, Source Port Number, Destination Port Number, and Transport Layer Protocol.

The complete dataset of 300000 packets was divided into two parts, the training_and_validation dataset, and the testing dataset. This was done using the *train_test_split* function of *sklearn* library which splits a dataset into random training and testing subsets. The training_and_validation dataset consisted of 240000 packets and the testing dataset consisted of 60000 packets. With random splitting that was done using *train_test_split*, the training_and_validation dataset contained 119997 normal IPv6 packets and 120003 covert IPv6 packets. Out of 120003 covert packets, 39881 packets contained secret data in the Traffic Class field, 40100 contained secret data in the Flow Label field, and 40022 packets contained secret data in the Hop Limit field.

The testing dataset contained 60000 packets in all, out of which 30003 were normal IPv6 packets and 29997 were covert IPv6 packets. Out of 29997 covert IPv6 packets, 10119 packets carried secret data in the Traffic Class field, 9900 packets carried secret data in the Flow Label field and 9978 packets carried secret data in the Hop Limit field.

TABLE 1. Description of dataset.

	Training Samples	Testing Samples
Normal	119997	30003
Covert – Traffic Class	39881	10119
Covert – Flow Label	40100	9900
Covert – Hop Limit	40022	9978
Total	240000	60000

The first dataset named the training_and_validation dataset was used to train and validate the classifiers of the proposed system. The second dataset named the testing dataset was used to test the generalization ability of the two-layered system trained with the first dataset. The tabular description of the dataset is shown in Table 1.

The preprocessing of both datasets was done independently using the same Python program to quantize and standardize the values. The next sub-section describes the preprocessing applied to both of the datasets separately.

B. DATASET PREPROCESSING

Dataset preprocessing is a technique for converting raw data into a form suitable for the training and testing of ML or DL classifiers. In this paper, the following steps were done using a Python program to preprocess the datasets. Firstly, a single attribute corresponding to the Source IPv6 Address field having colon-separated 8 octets was broken into 8 different attributes. Similarly, the Destination IPv6 address was broken down into 8 different attributes. The dataset contained two categorical attributes: Protocol and Next Header. Quantization was used to convert these categorical values into a unique number corresponding to different values of each attribute. Input attributes have different scales and hence, there is a need for scaling or standardization in ML algorithms. In this work, *standardscalar()* of *sklearn* library from Python was used to scale all the data. Before starting with the training phase, the preprocessing of the training_and_validation dataset and the testing dataset was done separately. At the beginning of the training phase, the preprocessed training_and_validation dataset was divided into two parts: the training dataset and the validation dataset. The next sub-section describes the training stage and testing stage of the proposed system.

C. TRAINING AND TESTING OF THE PROPOSED SYSTEM

In the training phase, the DNN selected at Layer 1 of the proposed system was trained using binary labels. For this, all labels of the training_and_validation dataset were converted to binary labels viz. 0 for normal and 1 for covert. During the training phase of any Machine Learning or Deep Learning algorithm, certain hyperparameter values need to be tuned. These hyperparameters include the number of hidden layers used in a classifier, the number of neurons used in the hidden layers of the classifier, the batch size, and the number of epochs used to train the classifier. The hyperparameters chosen for the DNN classifier at Layer 1 were as mentioned in Section IV. The Activation Function used at the hidden layers of the DNN was ReLU and at the output layer, the Sigmoid activation function was used. The optimizer used was the Adam optimizer.

Further, to find the location of covert data in an IPv6 packet header One-vs-Rest technique with SVM classifiers was used at Layer 2 for multiclass classification. It was implemented with *OneVsRestClassifier* function of *sklearn* library with SVM classifiers using RBF kernel.

Experiments were conducted on the proposed system to develop an efficient system in terms of both accuracy and prediction time.

D. EXPERIMENTS

Several experiments were conducted for choosing the classifier at Layer 2. The following combinations were experimented.

1. *DNN − OvR LR*: DNN binary classifier at Layer 1 and One-vs-Rest with Logistic Regression multiclass classifier at Layer 2.

2. *DNN − OvR SVM*: DNN binary classifier at Layer 1 and One-vs-Rest with Support Vector Machine multiclass classifier at Layer 2.
3. *DNN − Naïve Bayes*: DNN binary classifier at Layer 1 and Naïve Bayes multiclass classifier at Layer 2.
4. *DNN − DNN*: DNN binary classifier at Layer 1 and DNN multiclass classifier at Layer 2.
5. *DNN − CNN*: DNN binary classifier at Layer 1 and CNN multiclass classifier at Layer 2.
6. *DNN − LSTM*: DNN binary classifier at Layer 1 and LSTM multiclass classifier at Layer 2.

The classifier at Layer 2 for different combinations was trained using only the covert samples of the training_and_validation dataset. Further, the performance of these combinations was evaluated on the testing dataset. All these combinations were compared with respect to training time, prediction time, and accuracy percentage. The evaluation metrics used in this work for comparing various ML/DL algorithms are discussed in the next subsection.

E. EVALUATION METRICS

To measure the effectiveness of the proposed system, various metrics like precision, recall, F1-score, and accuracy percentage were calculated using the testing dataset on the proposed system. These metrics are computed with the help of a confusion matrix that evaluates the performance of a classifier on the testing dataset with true labels known in prior. A confusion matrix is a square matrix with elements $C_{i,j}$, such that $C_{i,j}$ specifies the count of samples truly belonging to be in class i and predicted by the classifier to be in class j.

Accuracy describes the total number of samples correctly classified by the classifier. Equation (4) is used to calculate the Accuracy of a classifier.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

where, TP stands for True Positives, specifying the number of samples that a classifier classified correctly. FP stands for False Positives denoting the number of samples that a classifier has erroneously classified as being positive. TN stands for True Negatives which denotes the number of samples that the classifier accurately classified as being negative. FN stands for False Negatives, which is the number of samples that the classifier erroneously classified as being negative.

Precision is described as the ratio of correctly predicted positive samples to the total number of samples predicted as positive by a classifier. Equation (5) gives the formula for calculating the precision value of a classifier.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

Recall is stated as the ratio of number of correctly predicted positive samples to the number of all the actual positive samples. Equation (6) calculates the recall value of a classifier.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

F1-Score is computed as the harmonic mean of the Recall value and the Precision value. Equation (7) calculates the F1-score of a classifier.

$$\text{F1} - \text{score} = \frac{2\ \text{XPrecisionXRecall}}{\text{Precision} + \text{Recall}} \tag{7}$$

For comparison purposes, the performances of different combinations of classifiers were evaluated in terms of the above-mentioned metrics along with the training time and average prediction time for locating covert samples only. The average prediction time for normal samples was not considered because the same DNN classifier was used at Layer 1 for all combinations. However, due to the use of different classifiers at Layer 2 for the combinations considered for comparison, the difference will only be in the prediction time of locating covert data in a covert IPv6 packet. The average prediction time for covert samples was calculated by selecting ten random covert test samples. The time taken by the system to predict the location of covert data in each of these samples was noted and an average value was calculated for the same. This is how the average prediction time for covert samples was calculated throughout this work. The training time of the combinations of classifiers was calculated as the sum of the time taken to train each classifier at both layers. The outcomes of all the aforementioned experiments are discussed in the next section.

VI. RESULTS

This section presents and discusses the results of experiments discussed in the previous section. Further, more experiments were conducted to compare the performance of the proposed system with various ML and DL algorithms used for multiclass classifications. Evaluation metrics described in section V were used to evaluate the performance of the proposed system and other counterparts considered for comparison in this paper.

A. EXPERIMENT TO CHOOSE SUITABLE CLASSIFIERS FOR THE PROPOSED SYSTEM

For deciding the classifiers at layer 2, the groupings mentioned in subsection D of section V were experimented with to find a combination that provides high accuracy percentage with an acceptable prediction time for locating secret data in covert IPv6 packets.

The results of these experiments in terms of accuracy percentage, training time, and average prediction time per covert sample after trying different combinations of classifiers are shown in Fig. 8, Fig. 9, and Fig. 10 respectively. To differentiate the performances of different classifiers, accuracy was calculated in terms of percentage. It was found that the combinations of DNN – DNN, DNN – CNN, and DNN – OvR SVM outperformed the rest by achieving the highest accuracy percentage in detecting and locating storage-based covert channels in IPv6. However, it was observed that out of these three combinations, DNN – OvR SVM took a much lesser average prediction time per covert sample with the second largest

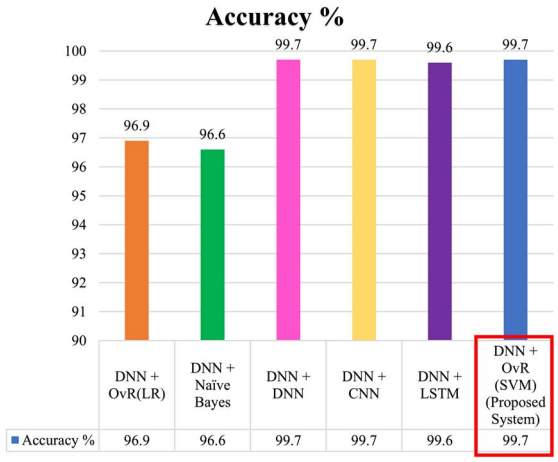

FIGURE 8. Comparison of accuracy percentage for different classifiers at Layer 2.

FIGURE 9. Comparison of training times with different classifiers at layer 2.

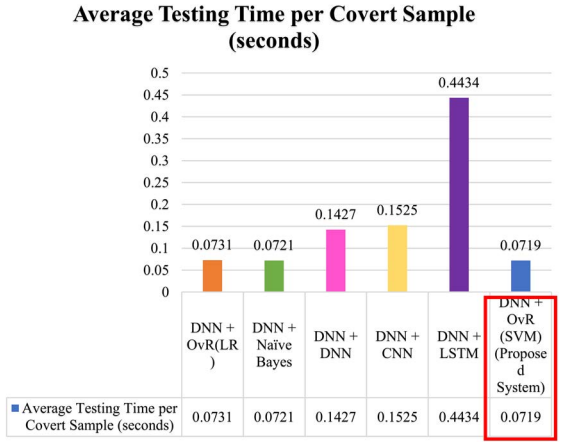

FIGURE 10. Comparison of average prediction times per covert sample with different classifiers at layer 2.

training time. Both training time and average prediction time are important parameters when developing systems that work on real-time networks but when compared to each other, the average prediction time metric is more significant. This is due to the fact that training a system is a single-time effort and is done only once when a system is deployed whereas average prediction time is a runtime metric that affects the efficiency of a system. Further, even a minute improvement in the average prediction time of a single sample can have a huge effect on the efficiency of the complete system, thereby increasing the significance of this metric. Thus, giving due importance to both metrics named accuracy percentage and average prediction time per sample, the combination with DNN at Layer 1 and OvR SVM at Layer 2 was finalized as the proposed system.

The confusion matrix generated while experimenting with the proposed system on the testing dataset is shown in Fig. 11. The next experiment compared the proposed system with the various state-of-the-art ML and DL classifiers used for multiclass classification.

Actual Class \ Predicted Class	Normal	Covert-Traffic Class	Covert-Flow Label	Covert – Hop Limit
Normal	29999	0	4	0
Covert - Traffic Class	0	10098	21	0
Covert – Flow Label	171	0	9729	0
Covert – Hop Limit	0	0	0	9978

FIGURE 11. Confusion matrix generated after experimenting the proposed system with the testing dataset.

B. COMPARISON WITH STATE-OF-THE-ART ML AND DL CLASSIFIERS

The proposed system was further compared with various state-of-the-art ML and DL classifiers for multiclass classifications. These algorithms included CNN, LSTM, OvR SVM, OvR LR, Naïve Bayes, and XGBoost classifiers. Fig. 12, Fig. 13, Fig. 14, and, Fig. 15 depict the comparison of accuracy percentage, precision, recall, and F1-score values of various ML and DL algorithms in consideration in predicting all four classes. The results showed that the proposed system outperformed all its counterparts in terms of accuracy percentage, precision, recall, and F1-score in detecting and locating the storage area of covert data in covert IPv6 packets. It was observed that the value for Covert-Hop Limit class is perfect 1 of precision, recall and F1-score for most of the classifiers, the reason for the same is the use of only two fixed values for injecting a 0 bit and 1 bit in the Hop Limit field by pcapStego tool. This enables the classifiers to easily distinguish normal IPv6 packets from covert IPv6 packets carrying secret data in the Hop Limit field.

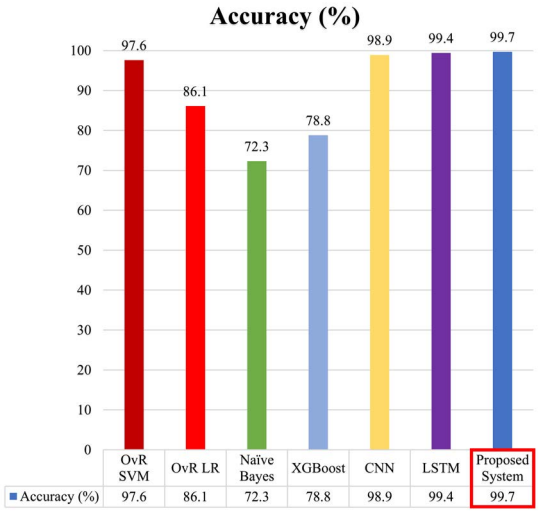

FIGURE 12. Comparison of accuracy of the proposed system with state-of-the-art ML/DL classifiers.

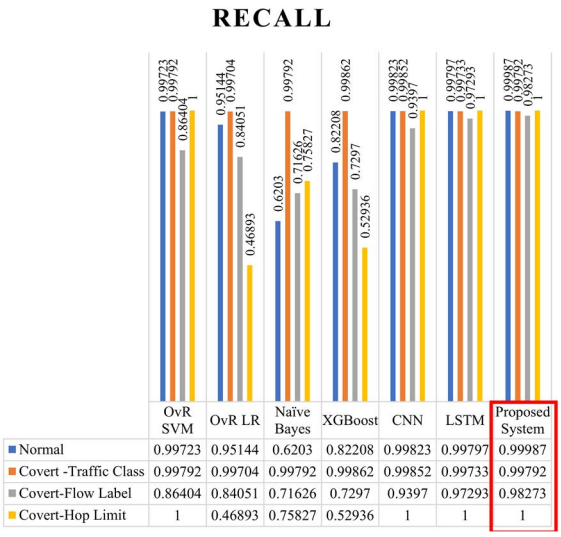

FIGURE 14. Comparison of recall values of the proposed system with state-of-the-art ML/DL classifiers.

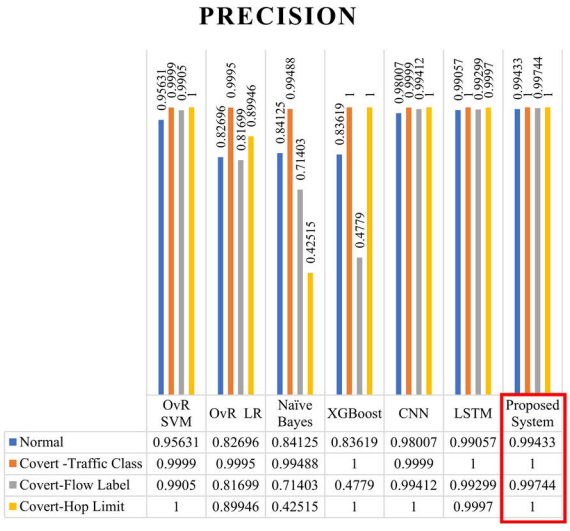

FIGURE 13. Comparison of precision values of the proposed system with state-of-the-art ML/DL classifiers.

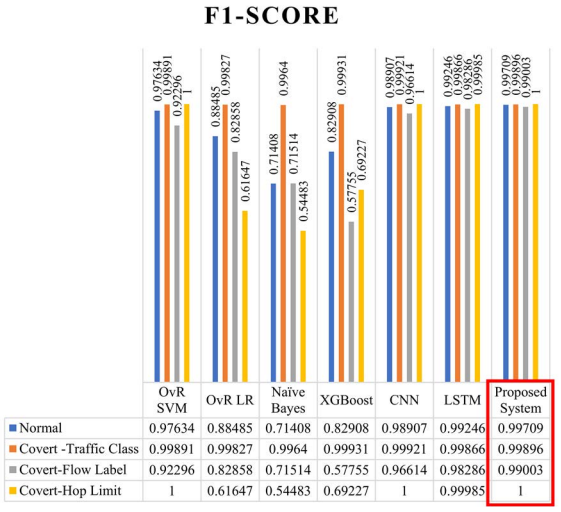

FIGURE 15. Comparison of F1-Score values of the proposed system with state-of-the-art ML/DL classifiers.

C. COMPARISON OF THE PROPOSED SYSTEM WITH OTHER RELATED WORKS

The proposed system was further compared with recent state-of-the-art works in the literature that emphasize the detection of covert channels in IPv6 protocol. Some of these works made use of DL/ML techniques to detect covert communications using IPv6. A comparison with such techniques is shown in Table 1. Rest of the techniques made use of packet filters present inside the kernel area to collect statistics for detecting covert channels. Luca et al. [13] made the use of code augmentation in eBPF within the Linux kernel to collect the statistics of the Flow Label field of the IPv6 header. Repetto et al. [14] proposed a tool bccstego that made use of the BCC tool which helps in obtaining statistics about IPv6 header fields like Flow Label, Traffic Class, and Hop Limit to detect anomalous behavior. The underlying concept in both of these works is to share a methodology for analyzing packets' headers and gathering data for hidden data analysis. The authors in their work inferred that abnormal changes in the statistical values of these header fields can raise an alarm about the existence of an NCC. The advantages that the proposed system offers over these eBPF-based techniques are many folds. Firstly, the proposed system performs the task of detecting as well as locating the storage area of covert channels in Flow Label field, Traffic Class field, and Hop Limit field of IPv6 packets whereas both eBPF-based techniques only detect the presence of covert channels. Secondly, the limitation of the eBPF-based covert channel detection mechanism gives good accuracy with more granular values of the number of bins which consumes a

large amount of resources in the nodes running eBPF whereas there is no such dependency in the proposed system. Thirdly, short-length covert communications cannot be detected using the methodology proposed in these eBPF-based techniques, whereas the proposed technique is suitable for detecting and locating any length of covert communication. Fourth, the authors discussed the limitation of the bccstego tool for showing less visibility of Hop Limit-based covert channels in IPv6, whereas the proposed system accurately identified the presence of covert data in the Hop Limit field.

Further, as shown in Table 2, the proposed system is performing the task of both detection and finding the location of the storage area whereas other systems/techniques only perform the task of detection of covert channels. Locating the storage area of covert data is equally important as its detection because it helps the Intelligence Agencies to detect as well as decode the existing covert message in the network packets. Thus, the proposed system outperforms its counterparts in terms of functionality with comparable accuracy percentage and prediction time.

TABLE 2. Comparison of proposed system with other related works.

Technique /System	Functionality	IPv6 Header Fields considered for detecting Covert Communication	Accuracy %	Prediction time per covert sample(s)
BNS-CNN [15]	Detection only	Hop-Limit, Source Address	100%	0.1710
DICCh-D [17]	Detection only	Flow Label, Traffic Class	99.5%	0.0710
Proposed System	Detection and location	Flow Label, Traffic Class, Hop-Limit	99.7%	0.0719

In addition, two new systems that perform detection and locating of the storage area of storage-based NCCs in IPv6 were developed using only DNN and only OvR SVM. Their performance was compared with that of the proposed system and the results have been presented in Fig.16. It was observed that the proposed system achieved maximum accuracy percentage in comparison to other systems in consideration.

Next, we experimented with different sizes of the dataset to analyze the effect of dataset size on the accuracy percentage of the proposed system by taking a different number of packets each time. The results for the same are shown in Fig. 17. It was observed that the proposed system obtained the highest accuracy percentage with the dataset size of 300000 IPv6 packets thereby justifying the use of a dataset with 300000 IPv6 packets for training and testing of the proposed system.

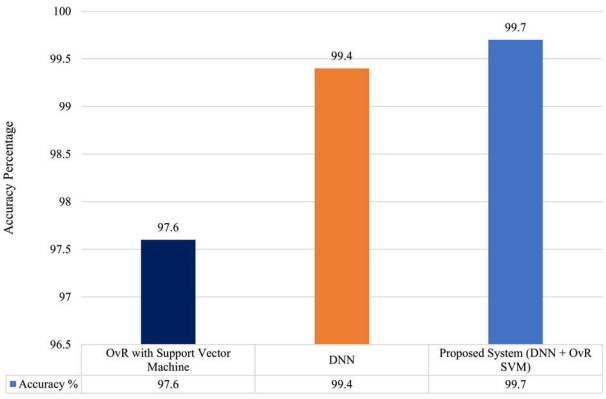

FIGURE 16. Comparison of accuracy of the proposed system with only DNN and only OvR SVM.

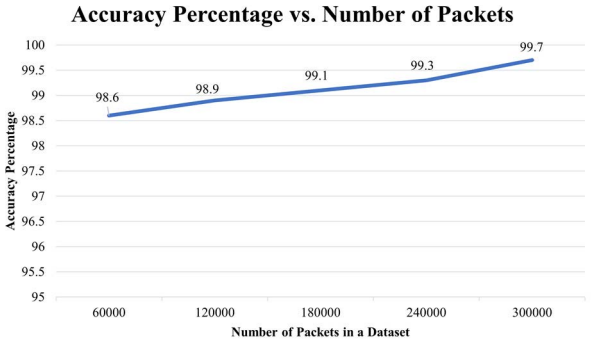

FIGURE 17. Effect of change in the size of the dataset on accuracy percentage of the proposed system.

To summarize, the proposed system performed the detection and locating of the storage area of storage-based covert channels in IPv6 with an accuracy percentage of 99.7average prediction time of 0.0691 seconds per normal test sample and 0.0719 seconds per covert test sample. The results from all the above-described experiments summarize that the proposed system not only performs accurate covert channel detection and locating of storage area in IPv6 but also does it in an efficient time.

VII. CONCLUSION

In the last decade, the transmission of malware through Information Hiding techniques like covert channels, also known as stegomalware has risen significantly. Thus, to capture such malicious hidden communications over IPv6, this paper proposes a system that not only detects the presence of a storage-based network covert channel developed over IPv6 protocol but also locates the storage area of the covert data in the IPv6 header.

Due to the unavailability of any benchmark dataset, a dataset containing normal and covert IPv6 packets was created to train and validate the proposed system. The normal IPv6 packets were taken from CAIDA's Anonymized Internet

Traces 2019 dataset and the covert IPv6 packets were generated using the pcapStego tool. For ensuring the accurate and efficient locating of storage-based covert channels in IPv6, extensive experimentation was done. Various state-of-the-art DL and ML algorithms were experimented with and compared, for choosing the best classifiers in terms of accuracy and average prediction time of the proposed system.

Further, to check the generalizability of the proposed system, a generalization test dataset was generated and was kept aside before the system training and validation phase. With the generalization test dataset, the proposed system obtained an accuracy percentage of 99.7% for detecting and locating secret data in covert IPv6 packets. Also, the proposed system gave efficient results in terms of average prediction time per test sample. It took only 0.0691 seconds on an average to predict a normal test sample and 0.0719 seconds on an average to locate the storage area of a covert test sample which was much lesser than its counterparts in consideration.

Overall the proposed system delivered an efficient performance in terms of the combination of the highest accuracy percentage and least average prediction time per covert sample in comparison to its counterparts in consideration for this work. Hence, the proposed system stands as a good candidate for deployment in real-world network scenarios. For future work, this work can be extended to explore the detection and locating of the storage area of secret data in simultaneous multiple header field locations of an IPv6 packet.

REFERENCES

[1] R. De, N. Pandey, and A. Pal, "Impact of digital surge during COVID-19 pandemic: A viewpoint on research and practice," *Int. J. Inf. Manag.*, vol. 55, Dec. 2020, Art. no. 102171.

[2] W. Mazurczyk and L. Caviglione, "Information hiding as a challenge for malware detection," *IEEE Security Privacy*, vol. 13, no. 2, pp. 89–93, Mar./Apr. 2015.

[3] K. Cabaj, L. Caviglione, W. Mazurczyk, S. Wendzel, A. Woodward, and S. Zander, "The new threats of information hiding: The road ahead," *IT Prof.*, vol. 20, no. 3, pp. 31–39, 2018.

[4] A. Velinov, A. Mileva, S. Wendzel, and W. Mazurczyk, "Covert channels in the MQTT-based Internet of Things," *IEEE Access*, vol. 7, pp. 161899–161915, 2019.

[5] M. A. Elsadig and A. Gafar, "Covert channel detection: Machine learning approaches," *IEEE Access*, vol. 10, pp. 38391–38405, 2022.

[6] W. Mazurczyk, S. Wendzel, M. Chourib, and J. Keller, "Countering adaptive network covert communication with dynamic wardens," *Future Gener. Comput. Syst.*, vol. 94, pp. 712–725, May 2019.

[7] O. Darwish, A. Al-Fuqaha, G. B. Brahim, I. Jenhani, and A. Vasilakos, "Using hierarchical statistical analysis and deep neural networks to detect covert timing channels," *Appl. Soft Comput.*, vol. 82, Sep. 2019, Art. no. 105546.

[8] K. Cabaj, P. Zorawski, P. Nowakowski, M. Purski, and W. Mazurczyk, "Efficient distributed network covert channels for Internet of Things environments," *J. Cybersecur.*, vol. 6, no. 1, pp. 1–18, Jan. 2020.

[9] Google. (Jun. 2022). *Google IPv6*. Accessed: Jun. 27, 2022. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption

[10] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert channels in IPv6," in *Proc. Int. Workshop Privacy Enhancing Technol.*, Berlin, Germany, 2005, pp. 147–166.

[11] P. Bedi and A. Dua, "Network steganography using extension headers in IPv6," in *Proc. 5th Int. Conf. Inf., Commun. Comput. Technol. (ICICCT)*, Delhi, India, 2020, pp. 100–110.

[12] W. Mazurczyk, K. Powójski, and L. Caviglione, "IPv6 covert channels in the wild," in *Proc. 3rd Central Eur. Cybersecur. Conf.*, Munich, Germany, Nov. 2019, pp. 1–6.

[13] L. Caviglione, W. Mazurczyk, M. Repetto, A. Schaffhauser, and M. Zuppelli, "Kernel-level tracing for detecting stegomalware and covert channels in Linux environments," *Comput. Netw.*, vol. 191, May 2021, Art. no. 108010.

[14] M. Repetto, L. Caviglione, and M. Zuppelli, "Bccstego: A framework for investigating network covert channels," in *Proc. 16th Int. Conf. Availability, Rel. Secur.*, Aug. 2021, pp. 1–7.

[15] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): A survey," *Comput. Commun.*, vol. 170, pp. 19–41, Feb. 2021.

[16] M. Chourib, "Detecting selected network covert channels using machine learning," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2019, pp. 582–588.

[17] A. Dua, V. Jindal, and P. Bedi, "DICCh-D: Detection of IPv6 based covert channels using DNN," *Presented at 7th Int. Conf. Inf., Commun. Comput. Technol. (ICICCT)*, Delhi, India, 2022.

[18] D. Zhao and K. Wang, "BNS-CNN: A blind network steganalysis model based on convolutional neural network in IPv6 network," in *Proc. Int. Workshop Digital Watermarking*, 2019, pp. 365–373.

[19] S. Deering and R. Hinden. (2017). *Internet Protocol, Version 6 (Specifications)*. Internet Engineering Task Force. Accessed: Jul. 16, 2020. [Online]. Available: https://tools.ietf.org/html/rfc8200#section-6

[20] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme. (Nov. 2011). *IPv6 Flow Label Specification*. Accessed: Jun. 10, 2020. [Online]. Available: https://tools.ietf.org/html/rfc6437

[21] B. Carpenter and S. Jiang. (Dec. 2013). *RFC 7045: Transmission and Processing of IPv6 Extension Headers*. IETF. Accessed: Feb. 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7045

[22] I. Goodfellow, Y. Bengio, and A. Courvil, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[23] S. J. Gustavus, "The prisoners problem and the subliminal channel," in *Advances in Cryptology*. Boston, MA, USA: Springer, 1984, pp. 51–67.

[24] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansader, and K. Szczypiorski, *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Hoboken, NJ, USA: Wiley, 2016.

[25] J. Lubacz, W. Mazurczyk, and K. Szczypiorski, "Principles and overview of network steganography," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 225–229, May 2014.

[26] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[27] M. A. Padlipsky, D. W. Snow, and P. A. Karger, "Limitations of end-to-end encryptions in secure computer networks," MITRE Corp., Bedford, MA, USA, Tech. Rep. ESD-TR-78-158, 1978.

[28] T. G. Handel and M. T. Sandford, "Hiding data in the OSI network model," in *Proc. Int. Workshop Inf. Hiding*, Berlin, Germany:, 1996, pp. 23–38.

[29] A. Castiglione, M. Nappi, F. Narducci, and C. Pero, "Fostering secure cross-layer collaborative communications by means of covert channels in MEC environments," *Comput. Commun.*, vol. 169, pp. 211–219, Mar. 2021.

[30] J. Saenger, W. Mazurczyk, J. Keller, and L. Caviglione, "VoIP network covert channels to enhance privacy and information sharing," *Future Gener. Comput. Syst.*, vol. 111, pp. 96–106, Oct. 2020.

[31] W. Frączek, W. Mazurczyk, and K. Szczypiorski, "Hiding information in a stream control transmission protocol," *Comput. Commun.*, vol. 35, no. 2, pp. 159–169, 2012.

[32] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibetts, "Covert messaging through TCP timestamps," in *Proc. Int. Workshop Privacy Enhancing Technol.*, Berlin, Germany, 2002, pp. 194–208.

[33] V. Sabeti and M. Shoaei, "New high secure network steganography method based on packet length," *ISC Int. J. Inf. Secur.*, vol. 12, no. 1, pp. 24–44, 2020.

[34] K. Ahsan and D. Kundur, "Practical data hiding in TCP/IP," in *Proc. Workshop Multimedia Secur. ACM Multimedia*, New York, NY USA, 2002, pp. 1–8.

[35] P. Bedi and A. Dua, "Network steganography using the overflow field of timestamp option in an IPv4 packet," *Proc. Comput. Sci.*, vol. 171, pp. 1810–1818, Jan. 2020.

[36] B. Ray and S. Mishra, "Secure and reliable covert channel," in *Proc. 4th Annu. Workshop Cyber Secur. Inf. Intell. Res., Developing Strategies Meet Secur. Inf. Intell. Challenges Ahead*, New York, NY, USA, 2008, pp. 1–3.

[37] P. Bedi and A. Dua, "ARPNetSteg: Network steganography using address resolution protocol," *Int. J. Electron. Telecommun.*, vol. 66, no. 4, pp. 671–677, 2020.
[38] A. Dua, V. Jindal, and P. Bedi, "Covert communication using address resolution protocol broadcast request messages," in *Proc. 9th Int. Conf. Rel., Infocom Technol. Optim. (Trends Future Directions) (ICRITO)*, Delhi, India, Sep. 2021, pp. 1–6.
[39] K. Aggarwal, M. M. Mijwil, A. H. Al-Mistarehi, S. Alomari, M. Gök, A. M. Z. Alaabdin, and S. H. Abdulrhman, "Has the future started? The current growth of artificial intelligence, machine learning, and deep learning," *Iraqi J. Comput. Sci. Math.*, vol. 3, no. 1, pp. 115–123, 2022.
[40] S. Abdulrahman, M. Xiaoqi, and E. Peytchev, "Detection and classification of covert channels in IPv6 using enhanced machine learning," in *Proc. Int. Conf. Comput. Technol. Inf. Syst.*, 2015, pp. 1–7.
[41] A. Senaid and F. Rashid, "A deep learning based approach to detect covert channels attacks and anomaly in new generation internet protocol IPv6," M.S. thesis, Dept. Comput. Sci. Eng., Qatar Univ., Doha, Qatar, 2020.
[42] (2021). *The CAIDA UCSD Anonymized Internet Traces Dataset-[20, Jan. 2019, 21, Jan. 2019, 22, Jan. 2019, 23, Jan. 2019], Center for Applied Internet Data Analysis*. Accessed: Jul. 2021. [Online]. Available: https://www.caida.org/data/passive/passive_dataset
[43] M. Zuppelli and L. Caviglione, "PcapStego: A tool for generating traffic traces for experimenting with network covert channels," in *Proc. 16th Int. Conf. Availability, Rel. Secur.*, Aug. 2021, pp. 1–8.
[44] F. Pedregosa, G. Veraquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cornapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[45] (2022). *Wireshark, Wireshark Foundation*. Accessed: Jun. 25, 2022. [Online]. Available: https://www.wireshark.org

ARTI DUA received the B.Sc. degree (Hons.) in computer science from Keshav Mahavidyalaya, University of Delhi, in 2003, and the M.C.A. degree from Guru Gobind Singh Indraprastha University, in 2009. She is currently pursuing the Ph.D. degree in computer science from the University of Delhi.

She was a Software Engineer with Altran (formerly known as Aricent Technologies (Holdings)), from June 2009 to July 2010. After that she worked as an Assistant Professor at Keshav Mahavidhyalaya, University of Delhi, from August 2010 to January 2011. She was the Head of Department of Computer Science, Bhaskaracharya College of Applied Sciences, University of Delhi, from April 2013 to December 2014. She has been an Assistant Professor with the Department of Computer Science, Bhaskaracharya College of Applied Sciences, University of Delhi, since February 2011. Her research interests include network steganography and its detection, cybersecurity, information hiding, network covert channels, and their detection.

VINITA JINDAL received the bachelor's degree in mathematics from the University of Delhi, in 1997, the M.C.A. degree from IGNOU, in 2000, the M.Phil. degree in computer science from Madurai Kamaraj University, in 2007, and the doctorate degree in computer science from the University of Delhi, in 2018.

She worked as the Manager/Senior Faculty Member at PCTI Ltd., from July 1999 to July 2001. She was the Head of Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, from June 2017 to May 2019. She has been a Professor with the Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, since November 2021. She is mainly working in the area of artificial intelligence and networks. Her research interests include covert channels and their detection, cybersecurity, intrusion detection systems, dark web, deep learning, recommender systems, and vehicular adhoc networks.

PUNAM BEDI (Senior Member, IEEE) received the M.Sc. degree in mathematics and the M.Tech. degree in computer science from IIT Delhi, in 1984 and 1986, respectively, received the doctorate degree in computer science from the University of Delhi, in 1999.

She was a Lecturer/Reader at Deshbandhu College, University of Delhi, from January 1987 to January 2002. She was the Head of Department of Computer Science, University of Delhi, from October 2005 to October 2008. She was the Acting Director of the Delhi University Computer Centre, from June 2009 to October 2009. She was the Officiating Director, Delhi University Computer Centre, from October 2017 to April 2018. She has been a Senior Professor with the Department of Computer Science, University of Delhi, since July 2018. Her research interests include steganography, steganalysis, cybersecurity, intrusion detection systems, recommender systems, deep learning, artificial intelligence for healthcare, and artificial intelligence for agriculture.