IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 August 2023, accepted 14 September 2023, date of publication 22 September 2023,
date of current version 27 September 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3318172*

RESEARCH ARTICLE

# SPYIPv6: Locating Covert Data in One or a Combination of IPv6 Header Field(s)

**PUNAM BEDI[1], (Senior Member, IEEE), VINITA JINDAL[2], AND ARTI DUA[1,3]**
[1]Department of Computer Science, University of Delhi, Delhi 110007, India
[2]Keshav Mahavidyalaya, University of Delhi, Delhi 110035, India
[3]Bhaskaracharya College of Applied Sciences, University of Delhi, Delhi 110075, India

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

**ABSTRACT** Advancement in the utilization of IPv6 protocol has led to an increase in research related to its security. In recent times, researchers proposed the possibility of the existence of covert channels over networks termed Network Covert Channels (NCCs) which may exploit IPv6. NCC is a serious threat that provides a hidden avenue for the transfer of information from one end to another. Hence, to detect and locate such threats that use IPv6 packets as cover, SPYIPv6 is proposed that detects the existence of hidden information in IPv6 packets and further identifies its location in one or a combination of IPv6 header field(s). The proposed SPYIPv6 comprises two layers. The first layer detects the covert IPv6 packets in the network traffic using a binary K-Nearest-Neighbour (b-KNN) classifier. These packets are further passed to the second layer that locates the header field(s) carrying covert data using a multiclass K-Nearest-Neighbour (m-KNN) classifier. The experimentation dataset was generated from normal and covert IPv6 packet samples. Normal packets were obtained from the Center for Applied Internet Data Analysis (CAIDA), whereas covert packets were obtained using an NCC generation tool (*pcapStego*) and Python scripts. Experimentation results show that SPYIPv6 attains an accuracy of 99.85% in detecting and identifying the location of hidden information in the IPv6 header. Further, when compared with other counterparts, SPYIPv6 provides higher accuracy in lesser testing time justifying its suitability for the detection and location of covert information present in one or a combination of the header field(s) of an IPv6 packet.

**INDEX TERMS** Cybersecurity, detection of covert channels, IPv6, K-nearest-neighbour (KNN), label powerset, network security.

## I. INTRODUCTION

With rapid innovations and the growth of technology, the dependence on the Internet has been rising. The use of new technologies like IoT, Cloud Computing, etc. for developing smart technologies for a better future has increased the load on the Internet. Consequently, there is an increase in the need for Internet addresses needed for communication of smart devices over the Internet. The IPv4 addresses are limited and are close to exhaustion. The next-generation IPv6 protocol is catering to this problem by offering a much larger address space of $2^{128}$ IP addresses. With IPv6 logical addressing, smart devices can have unique IP addresses without any need for performing traditional procedures like Network Address Translation and handling firewall issues. With the growing load and dependence on IPv6, ensuring the security of this protocol is a major area of concern for researchers. A lot of work is being done on the security aspect of IPv6 [1], [2]. In real-world management problems, minority events like cyber-attacks carry much more importance and useful knowledge than common events, despite their rareness due to their impact [3]. As this is only after studying these, one can be prepared to handle such unwanted events. The possibility of threats like data exfiltration, DDoS attacks, and stegomalware have also been explored by various researchers. Stegomalware is a special category of malware that use information hiding techniques to hinder its detection [4]. The medium for Information hiding that is used to conceal secret data is termed a cover. This cover may comprise an image, audio, video, text, or a network traffic flow. Network Covert

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

Channels (NCCs) use the network traffic flow(s) as cover to carry secret data [5]. NCCs may be categorized into two types based on the method of injecting secret information into the network traffic flow(s) [6].

1. Storage-based NCCs: These are those types of NCCs that use the storage area of a network protocol packet such as unused or reserved places in a protocol header to hide secret data [7]. This type of NCCs may carry secret data in single or multiple locations in a protocol header. In this paper, we worked on the detection and location of covert data present within the field(s) of an IPv6 header.
2. Timing-based NCCs: These are also the sub-category of NCCs that use inter-packet delays or the timing information related to a network protocol packet to hide the secret data [8].

Three major characteristics govern the effectiveness of an NCC. These comprise the robustness to external attacks, the undetectability of hidden information, and the hiding capacity. The robustness of an NCC is defined as the resilience of a covert channel to external changes due to noise or other factors over the networks. Undetectability refers to the inability to find the presence of a secret message inside a cover. Regarding the hiding capacity of an NCC, it is defined as the number of secret data bits that it can carry in a cover protocol packet. Further, the more the usage of a network protocol over the networks, the more its number of packets over the networks which results in higher covert capacity per unit of time. For example, the usage of TCP and UDP is more than SCTP over the Internet, hence TCP and UDP can provide more cover packets to carry secret data compared to the SCTP protocol. The usage of IPv6 has been on a continuous rise over the Internet. As of 6$^{th}$ May 2023, the adoption percentage as per Google statistics was 43.29% [9]. With its increased usage, the use of this protocol as a cover for information hiding has been investigated by various researchers [10], [11], [12].

### A. MOTIVATION AND CONTRIBUTIONS

With the ongoing research exploring the possible covert channels over IPv6, there is an equal need for research and development of detection techniques for handling such covert channels. References [13], [14], and [15] try to find a pattern for scrutinizing IPv6 packets and gathering statistical indicators. The unusual variations in the statistical values related to the considered header fields can generate an alert regarding the presence of some hidden communication. Zhao and Wang [16] presented an approach that aimed at detecting covert channels in HL and SA fields of IPv6 using CNN. Dua et al. [17] presented the use of a Deep Neural Network classifier for the detection of IPv6-based covert channels. Most of the papers in the literature discuss only the storage-based covert communication detection in IPv6. Further, Dua et al. [18] presented a system that detects and finds the location of the hidden data in covert communications that utilize either field of an IPv6 header amongst Hop Limit, Flow Label, and Traffic Class. However, we could not find any work that detects and identifies the location of the hidden data present in a combination of header fields of an IPv6 packet. The existence of hidden data in multiple header fields is a possibility in real-world scenarios of covert communications that can provide high covert capacity. An attacker who wants to send more covert information in a single packet may use multiple header fields at the same time to carry covert data. Hence, identifying the location of covert information in multiple fields of a covert packet's header is a vital research area as it will help in performing next-level processing like interpreting the hidden information. Thus, with this motivation in this paper, SPYIPv6, a novel system that detects and identifies the location of hidden data present in one or a combination of the header field(s) of an IPv6 packet is proposed. The proposed SPYIPv6 is a system that comprises two layers. These layers are connected in a sequential manner where the output of the first layer is fed as input to the second layer. The first layer separates IPv6 packets into covert/normal packets. Further, for covert packets, the second layer finds the location of covert information present in one or a combination of the header field(s) of each covert IPv6 packet.

To summarize, the key contributions made by this paper are:

1. The paper proposes SPYIPv6, a system comprising two layers for detecting as well as finding the location of covert data present in one or a combination of the header field(s) in an IPv6 packet. The layers in this system are connected sequentially. The first layer of the proposed SPYIPv6 uses a binary KNN classifier that separates covert and normal IPv6 packets. The packets identified as covert are further passed to the second layer to find the location of hidden information in one or a combination of the header field(s) using a multiclass KNN classifier.
2. To find the best classifiers at both the layers of the proposed SPYIPv6 (that provide high accuracy and take less time) for the detection and location of storage-based NCCs, extensive experimentation was done with several ML and DL classifiers. The various metrics like precision, recall, accuracy, F1-score, average testing time per sample, and training time are used for the performance evaluation of SPYIPv6.
3. A dataset containing 420000 IPv6 packets (Covert + Normal) was created because of the non-availability of a standard dataset. The normal packets were gathered from *UCSD Anonymized Internet Traces Dataset [January 2019]* received from CAIDA. The covert packets comprised single-field covert packets (obtained with the help of a tool named *pcapStego*), two-field covert packets (obtained using Python Scripts), and three-field covert packets (obtained using Python Scripts).
4. The proposed two-layered system provides a consolidated solution for two types of applications: one, which is only interested in detecting if a given IPv6 packet belongs to a normal/covert category, and second, which further provides the location of hidden data present in

one or a combination of header fields of an IPv6 packet for the packets identified as belonging to the covert category.

### B. STRUCTURAL ORGANIZATION

The rest of the paper is structured as follows. Section II tabulates the header structure of IPv6, followed by a brief description of the K-Nearest-Neighbour algorithm, used in the development of the proposed SPYIPv6. Next, it elaborates on the threat model referred to in this paper. The literature review which is presented in Section III discusses various works proposed for the creation or detection of storage-based NCCs over various protocols followed by the working of SPYIPv6 in Section IV. Section V presents various experiments performed for the development, training, and testing of the proposed SPYIPv6 followed by the obtained results. Finally, Section VI presents the conclusion.

## II. BACKGROUND STUDY

This section briefly describes IPv6 and the K-Nearest-Neighbour algorithm. Next, the paper discusses the threat model and reference scenario considered in this paper.

### A. INTERNET PROTOCOL VERSION 6

Internet Protocol version 6 is a recent network layer protocol that provides logical addresses to the devices communicating over the Internet. The IPv4 protocol, which is the previous version of this protocol provided 32 bits long logical addresses thereby providing approximately 4.3 billion addresses. With the growing number of devices over the Internet worldwide, the IETF (Internet Engineering Task Force) proposed and developed IPv6 which supported 128 bits long IP addresses offering a much larger address space. The shifting of technologies from IPv4 to IPv6 over the Internet is still in progress. According to IPv6 statistics provided by Google, the usage of IPv6 is increasing worldwide every year [9]. RFC 8200 gives the specification document of the IPv6 [19].

The IPv6 header contains eight fields in its base header and extension headers (if any). All the header fields serve a special purpose. Table 1 discusses the size and purpose of each of these header fields briefly. The next subsection describes the K-Nearest-Neighbour algorithm.

**TABLE 1. Description of header fields of IPv6 protocol.**

| Header Field | Size | Purpose |
|---|---|---|
| Version | 4 bits | Denotes the Internet Protocol's version being used. |
| Traffic Class (TC) | 8 bits | Used to assign priority of a packet. |
| Flow Label (FL) | 20 bits | Used to point the packets that need special management by midway routers. |
| Payload Length (PL) | 16 bits | Contains the total payload length carried by packet. |
| Next Header (NH) | 8 bits | Stores a number corresponding to the next protocol header attached to the IPv6 base header. |
| Hop Limit (HL) | 8 bits | Denotes the number of hops a packet can pass through before being discarded. |
| Source Address (SA) | 128 bits | 128 bits long address of the source node |
| Destination Address (DA) | 128 bits | 128 bits long address of the destination node |

### B. K-NEAREST-NEIGHBOUR

*K-Nearest-Neighbour* (KNN) [20], [21] is a non-parametric ML algorithm that is based on supervised learning. It works on the idea that similar items exist in close proximity. It is a lazy algorithm that makes no assumptions from the training data. Instead of creating a model, it stores the entire dataset. KNN makes classification on a test sample by finding its $K$ most similar samples in the training dataset. To find similar samples, the distances between the test sample and all training samples are calculated and based on these distances, the samples are sorted. The first $K$ samples in the order are then fetched with their labels as the most similar samples. At last, the mode of the fetched labels is calculated and returned as the final classification.

The two significant hyperparameters used in the KNN algorithm are the $K$ value which denotes the number of nearest neighbours (samples) to be considered, and the distance calculation algorithm used. The hyperparameter $K$ selects the number of similar neighbours to be found each time a test sample is presented for prediction. For example, to make a classification using 3 nearest neighbours the value of $K$ is selected as 3 and the samples in the innermost circle are chosen for classification as shown in Fig. 1.

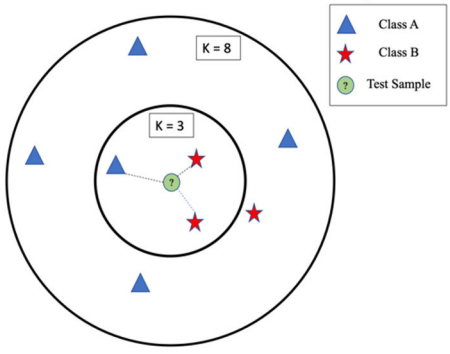



**FIGURE 1. K-nearest-neighbour classification.**

The second parameter is the distance algorithm that is used to calculate the similarity index of instances. The distance algorithms used with KNN are Minkowski Distance, Euclidean Distance, and Manhattan Distance which are calculated with the help of equations (1), (2),

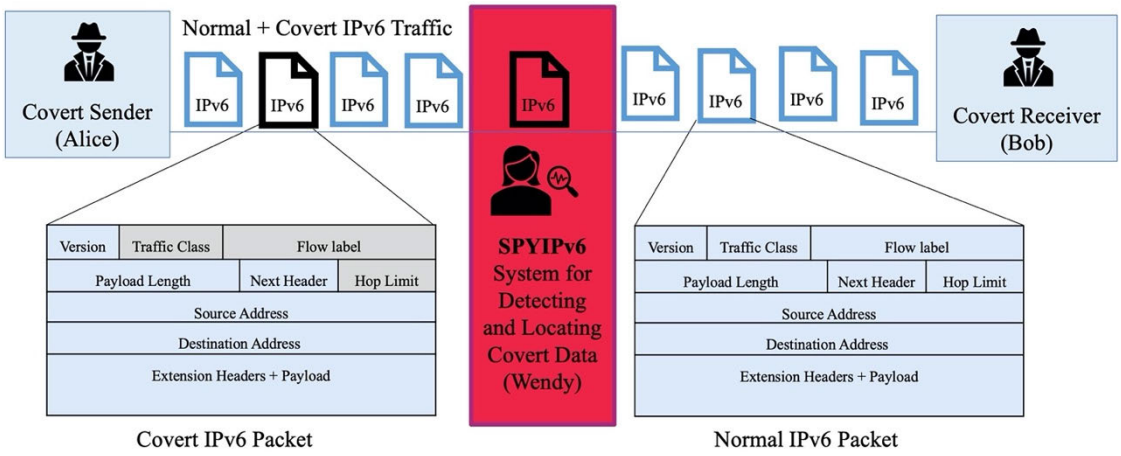



**FIGURE 2.** Reference model for covert communications over IPv6.

and (3) respectively.

$$\sqrt[q]{\sum_{i=1}^{k} (\mathrm{xi} - \mathrm{yi})^{\mathrm{q}}} \tag{1}$$

$$\sqrt{\sum_{i=1}^{k} (\mathrm{xi} - \mathrm{yi})^{2}} \tag{2}$$

$$\sum_{i=1}^{k} \parallel (xi - yi) \parallel \tag{3}$$

When the value of q = 1, in the Minkowski distance, it is equivalent to the Manhattan distance, and when the value of q = 2, it is equivalent to the Euclidean distance. The next subsection describes the threat model used as a reference model to understand the framework of covert communications.

### C. THE THREAT MODEL

The covert communication framework is best explained using Prisoner's Problem [22] as a de-facto model for hidden communication. In the Prisoner's Problem, Alice and Bob are the two prisoners who want to escape prison. For that, they require to communicate clandestinely. However, warden [23] Wendy is watching over all the communications taking place between Alice and Bob. Hence, Alice and Bob must interact stealthily. The threat model for the same is given in Fig. 2. Here Alice and Bob exchange messages clandestinely by hiding secret information in one or a combination of header field(s) (TC, FL, and HL) of an IPv6 packet. Thus to expose such communications over IPv6, a system for the warden Wendy is proposed in this paper that not only helps Wendy to detect the presence of covert communication but also finds the location of hidden data for further processing. The next section presents a brief discussion on NCC development and detection techniques proposed in the recent literature.

## III. LITERATURE REVIEW

Studying and analyzing covert channels has always been a favorite area of researchers. Many researchers have recently proposed the possibility of covert channels in new or upcoming technologies or protocols. These latest technologies include blockchain [24], [25], [26], [27] etc. The latest protocols include protocols like IPv6 [28] etc. Network Steganography (NS) and Network Covert Channels are the two techniques that implement Information Hiding in network traffic flows [29]. NS is a young subclass of steganography techniques that utilize network packets as cover to carry out hidden communications [30]. It covers the covert communication process by injecting secret messages into the legitimate overt network traffic without altering the carrier significantly [31], [32]. The idea of a covert channel was first proposed by Lampson [33] in the year 1973 concerning monolithic systems in which "a process at a higher level of security may leak information to a process at a lower level of security which would have been inaccessible otherwise". In the field of computer networks, the idea of network covert channels was first introduced in 1978 by Padlipsky et al. [34].

Further, in 1996, Handel et al. [35] proposed the possible misuse of various network protocols in the Open System Interconnections (OSI) reference model for the development of covert channels. Since then various techniques for the development of NCCs have been proposed, a few of which are summarized in [36], [37], [38], [39], [40], [41], [42], and [43] over different protocols like IPv4, ARP, TCP, UDP, MQTT etc. used over the Internet.

IPv6 is a younger protocol. Lucena et al. [10] proposed 22 theoretical covert channels over IPv6 using various header fields and extension headers. The limitation of their work was that all these NCCs were proposed theoretically without validating them experimentally. Next, to check the practical feasibility of these covert channels over the Internet, Mazurczyk et al. [11] performed various experiments by embedding data in various header fields of an IPv6 packet. They also showed that these covert channels can easily bypass popular IDS like Suricata and bro-Zeek. These experiments reduced the possible number of header fields that may be utilized as

covert channels in an IPv6 header to fewer fields like Traffic Class (TC), Hop Limit (HL), and Flow Label (FL).

Further, Bedi and Dua [12] proposed a technique in which the occurrence or absenteeism of IPv6 extension headers in a specific order encodes a covert message. This technique has limited usage due to the limited use of extension headers in the IPv6 packets observed over the Internet.

With the development of many NCC creation techniques, the research community has also proposed various methods for the detection of NCCs. Xuan et al. [44] experimented with different Deep Learning techniques that use backpropagation for training [45] such as LSTM, Multilayer Perceptron, and CNN to evaluate the most effective technique to detect Network Steganography in selective header fields of IPv4, IPv6, TCP, UDP, ICMP, MQTT, Ethernet, and Frame. They obtained the best accuracy with LSTM but that recorded a higher testing time. The limitation of their study is that the header fields used for covert data storage in the IPv6 header viz. source address, destination address, and next header can be detected easily and don't require advanced detection mechanisms. The source address field if used for covert communication can easily be detected with widespread protection against spoofing. The destination address field cannot be used to carry covert data because in this case there will be no place to store the destination address of an IPv6 packet which is required to make a packet reach its intended destination. Lastly, the values of the next header field used over the internet are limited due to the restricted use of extension headers for security reasons, thus if these eight bits corresponding to the next header are used for covert communication then they can easily be detected.

Chourib et al. [46] presented the use of KNN to detect storage-based NCCs in selective header fields of various protocols viz. IPv4, ICMP, IGMP, TCP, and UDP. They created the dataset by taking normal packets from a benchmark dataset and created covert packets with the help of various covert channel generation tools for protocols under consideration. They also compared the results by training and testing various classifiers like DNN and SVM. This work did not consider IPv6-based covert channels and further aimed at only detecting the existence of specific covert channels. In our paper, we propose the detection as well as identification of the location of secret information in one or a combination of the header field(s) of covert IPv6 packets.

Cho et al. [47] proposed the use of the Random Forest algorithm to detect the presence of secret data in various protocols viz. IPv4, TCP, and ICMP. They also compared their results with the performance of SVM and Naïve Bayes classifiers on the same datasets.

Luca et al. [13] suggested using code augmentation technique in the Linux kernel to record the statistics with the help of extended Berkeley Packet Filter (eBPF) corresponding to some IPv6 header fields. Utilization of BCC (BPF Compiler Collection) framework for executing eBPF programs that gather statistical information about header fields like TC, FL, and HL in the IPv6 header for detecting hidden communications was suggested by Repetto et al. [14]. In the extension work, Zuppelli et al. [15] suggested the use of code-layering schemes over eBPF to detect storage-based NCCs in IPv6 and other timing-based channels. The fundamental concept in [13], [14], and [15] is to find a pattern for scrutinizing IPv6 packets and gathering statistical indicators. The unusual variations in the statistical values related to the considered header fields can generate an alert notifying the network administrator regarding the presence of some hidden communication. As stated by the authors, the detection techniques that leverage eBPF have limitations. Firstly, the indicator viz. number of bins needs to have more granulated values for providing good accuracy which would use a large amount of memory for the node executing eBPF. Second, these techniques are not suitable for identifying short-length covert communications.

Further, Salih et al. [48] proposed a framework using a modified C4.5 technique with IHA to generate primary training data for detecting covert channels over IPv6. Although their method reported an accuracy value of 94.47%, the creation of their dataset is unclear as the authors did not specify the criteria applied for marking the values in the Hop Limit field as high, low, or moderate. Similarly, they did not specify the criteria for choosing the values as increased, decreased, and low for different Payload Length field values.

Zhao et al. [16] presented an approach that aimed at detecting covert channels in HL and SA fields of IPv6 using CNN. Here header fields' values of captured IPv6 packets are transformed into a matrix. This matrix contains the number of IPv6 header fields as the number of rows and the size (in bits) of the longest header field as the column number. They recorded 100% accuracy. The limitation of this work is that the authors aimed at only detecting storage-based covert communications. Additionally, the detection technique was developed for covert channels utilizing two individual header fields named HL and SA only. Further, the possibility of covert channels in the SA Field is rare as this choice was already eliminated in [11] because of the extensive use of spoofing detection techniques.

Dua et al. [17] presented the use of a Deep Neural Network classifier for the detection of IPv6-based covert channels. They achieved an accuracy of 99.59% in detecting IPv6-based NCCs targeting only the TC and FL fields individually.

As per the best of our knowledge, state-of-the-art work presented to date mostly discusses only NCCs detection in IPv6 header fields. Recently, Dua et al. [18] proposed a system that detects and locates the storage area of IPv6-based NCCs in any one of the header fields namely TC, HL, and FL. However, we could not find any work that detects and identifies the location of the hidden data present in multiple header fields of an IPv6 header simultaneously. The existence of covert data in multiple header fields is a possibility in real-world scenarios for covert communications providing high covert capacity. An attacker who wants to send more covert information in a single packet may use multiple header fields at the same time to carry covert data. Thus, identifying

the placement of covert data in multiple fields of an IPv6 header is a vital research area as it will help in performing next-level processing like interpreting the hidden data. Hence in this work, SPYIPv6 a system for identifying the location of covert data in multiple header fields of an IPv6 header is proposed. Further, due to the practical feasibility of header fields like TC, FL, and HL as covert data carriers over the real-time networks [11], most of the recent literature work aims at covert communication detection that utilizes these three header fields [13], [14], [15], [17]. For the same reason, these header fields are particularly investigated (individually and together in all combinations) for detection and finding the location of hidden data in this work. The next section explains the complete working of the proposed SPYIPv6.

## IV. PROPOSED SPYIPv6

The proposed SPYIPv6 detects and finds the location of covert data present in one or a combination of header fields of an IPv6 packet using two layers that are connected sequentially. The first layer performs binary classification for segregating covert and normal IPv6 packets using a binary KNN (b-KNN) classifier. Those packets that are identified as covert are then passed to the second layer. The second layer uses a multiclass KNN (m-KNN) classifier to find the location of hidden data present in one or a combination of the header field(s) of the covert IPv6 packet filtered by the first layer of SPYIPv6. Finding the location of hidden data in a storage-based NCCs over IPv6 is a significant task, as it is only after identifying the location of secret data, the next-level processing like interpreting a covert message or traffic normalization in the respective location(s) of the header area can be performed.

### A. ASSUMPTIONS

For the development of the proposed SPYIPv6, an assumption was made:

1. Because of the non-availability of any standard dataset comprising covert and normal IPv6 packets, a dataset was generated. The normal packets were gathered from the *UCSD Anonymized Internet Traces Dataset [January 2019]* [49] requested and obtained from the Centre for Applied Internet Data Analytics (CAIDA). The covert IPv6 packets were obtained for three categories viz. single field-based covert packets, two fields-based covert packets, and three fields-based covert packets.

The development of the proposed SPYIPv6 was done in four phases out of which the first three phases contribute towards the dataset creation and the last phase performs training, validation, and testing of the proposed SPYIPv6. Phase 1 consisted of gathering packet capture (.pcap) files containing covert/normal IPv6 packets. Phase 2 performed the extraction of various header fields from the packet capture files containing normal IPv6 packets and packet capture files carrying covert IPv6 packets (created in Phase 1). This resulted in a dataset containing normal packets, single-field covert packets, two-field covert packets, and three-field covert packets. Phase 3 transformed the multilabel dataset created in Phase 2 into a multiclass dataset. Phase 4 consisted of training, validation, and testing of SPYIPv6. The complete development framework of SPYIPv6 is shown in Fig. 3. Each of the phases is explained in detail in the consequent subsections.

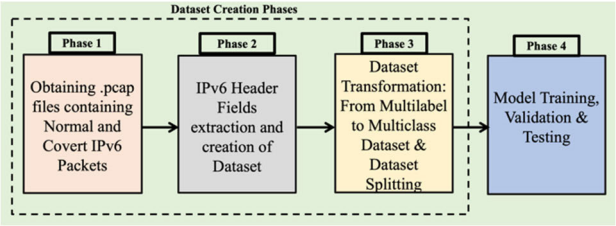



**FIGURE 3.** Dataset creation and development of SPYIPv6.

#### 1) PHASE 1: OBTAINING .pcap FILES CONTAINING NORMAL AND COVERT IPv6 PACKETS

In view of the fact of the nonavailability of any standard dataset comprising normal and covert storage-based IPv6 packets, a dataset is generated for the training and testing of SPYIPv6. *CAIDA's Dataset* [49] was used to obtain normal IPv6 packets. Regarding the covert packets, the single-field covert packets are obtained by utilizing the *pcapStego* tool [50]. For two-field and three-field covert channels in IPv6 packets, Python scripts using *Scapy* [51] are written to inject data into normal IPv6 packets. For two-field covert channels data is injected in three combinations: {TC – FL}, {TC – HL}, and, {FL – HL} fields. For three-field covert channels, data is injected in {TC – FL – HL} fields altogether. Fig. 4 depicts the creation of packet capture (.pcap) files carrying normal and secret data in IPv6 headers. The details of number of packets obtained for each class are given in Section V.

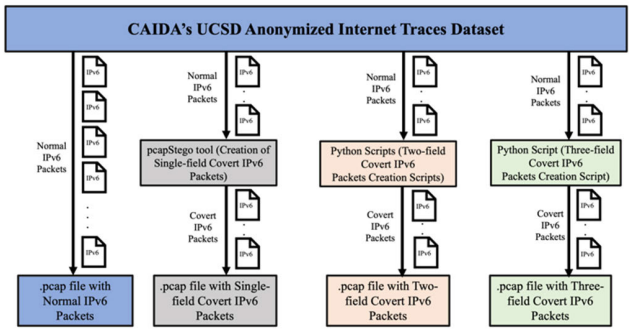



**FIGURE 4.** Obtaining packet capture files carrying normal and covert IPv6 packets.

#### 2) PHASE 2: IPv6 HEADER FIELDS EXTRACTION AND CREATION OF DATASET

In Phase 2, the header fields of all the IPv6 packets captured in the .pcap files obtained in Phase 1, are extracted. IPv6 Header

fields viz. SA, DA, PL, TC, NH, HL, FL, and Transport layer protocol's header fields viz. Transport Layer Protocol, Source Port, and Destination Port are extracted as features from each packet in the .pcap files. Next, the corresponding label attributes are added to create a dataset. The following four labels are added for each extracted sample: Normal, Covert Traffic Class, Covert Flow Label, and Covert Hop Limit. For further reference, Covert Traffic Class is referred to as Covert_TC, Covert Flow Label is referred to as Covert_FL and Covert Hop Limit is referred to as Covert_HL. For illustrating the addition of label values, consider the following two scenarios. First, if a packet is a normal IPv6 packet, the value for the Normal label is set to 1, and the rest three labels are set to 0 in the corresponding row of the dataset. Similarly, if a packet carried secret data in both Flow Label and Traffic Class fields, then the labels viz. Covert_FL and Covert_TC are set to 1 and the rest two labels (Normal and Covert_HL) are set to 0 for that sample in the dataset. This is how the dataset carrying normal and covert IPv6 packets is created. Fig. 5 depicts the extraction of header fields from IPv6 packets and the creation of a dataset containing the extracted header fields.

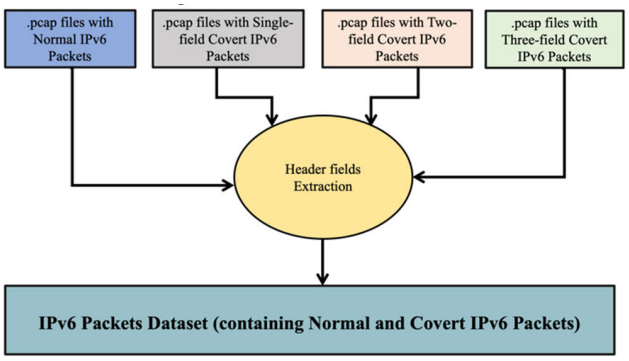



**FIGURE 5.** Header fields extraction and dataset creation.

### 3) PHASE 3: DATASET TRANSFORMATION FROM MULTILABEL TO MULTICLASS DATASET AND DATASET SPLITTING

The dataset obtained in Phase 2 consisted of 4 labels corresponding to each sample: Normal, Covert_TC, Covert_FL, and Covert_HL. The commonly used technique for handling a multilabel problem is the use of any Problem Transformation methods. The Problem transformation methods address the multilabel classification task by transforming the multilabel dataset into one or more binary/multiclass datasets. Further, the transformed dataset(s) are used to train existing binary/multiclass classification methods to build one or more single-class models. The commonly used approaches for Problem transformation are Binary Relevance (BR) and Label Powerset (LP). Both techniques use simple classifiers for problem-solving. In Binary Relevance, $n_L$ binary classifiers are used, where $n_L$ is the number of labels in the original multilabel problem whereas in the Label Powerset method, a single multiclass classifier is used to address the problem. The advantage of using the LP method over BR is that a single multiclass model is learned instead of multiple binary-class models. Also, the LP method preserves the dependency amongst the labels whereas BR ignores it. For more details on Problem Transformation methods, [52] can be referred to. Due to the advantages of the LP method, the same is used for Problem Transformation in this paper. Thus, in the third phase, the original multilabel dataset is transformed into a multiclass dataset using the Label Powerset method. The transformation applied to the dataset is shown in Fig. 6, where X stands for input features viz. Transport layer protocol, Source's port number, Destination's port number, TC, SA, DA, FL, PL, HL, and NH. Further, this transformed dataset is split into: the *Train_and_validate dataset*(eighty percent of the total) and the *Test dataset (twenty percent of the total)*. The *Train_and_validate dataset* is utilized to perform training and validation of the proposed SPYIPv6. The *Test dataset* is utilized to assess its generalization ability.

Input Features | Output With Multiple labels

| X | Normal | Covert-Traffic Class | Covert-Flow Label | Covert - Hop Limit |
|---|---|---|---|---|
| $x_1$ | 1 | 0 | 0 | 0 |
| $x_2$ | 0 | 1 | 0 | 0 |
| $x_3$ | 0 | 0 | 1 | 0 |
| $x_4$ | 0 | 0 | 0 | 1 |
| $x_5$ | 0 | 1 | 1 | 0 |
| $x_6$ | 0 | 1 | 0 | 1 |
| $x_7$ | 0 | 0 | 1 | 1 |
| $x_8$ | 0 | 1 | 1 | 1 |
| $x_9$ | 0 | 1 | 1 | 0 |

Dataset Transformation using Label Powerset Method

Input Features | Output Classes

| X | Transformed Class |
|---|---|
| $x_1$ | 0 |
| $x_2$ | 1 |
| $x_3$ | 2 |
| $x_4$ | 3 |
| $x_5$ | 4 |
| $x_6$ | 5 |
| $x_7$ | 6 |
| $x_8$ | 7 |
| $x_9$ | 4 |

**FIGURE 6.** Dataset transformation from multilabel dataset to multiclass dataset using label powerset method.

### 4) PHASE 4: TRAINING, VALIDATION, AND TESTING OF SPYIPv6

After transforming the multilabel dataset into a multiclass dataset and splitting it, the next phase comprises the training, validation, and testing of SPYIPv6.

Before training the SPYIPv6, pre-processing of the training dataset is done which comprises quantization and standardization of the data. To transform all category-based values into numeral-based values, quantization is applied, whereas standardization scales the data to fit the standard normal distribution. After applying this pre-processing, the training dataset is used for training. At the first layer, the binary KNN (b-KNN) is trained on the pre-processed training dataset after transforming the multiclass values into binary class values. This is done by assigning a class value of 0 to a normal IPv6 packet and a class value of 1 to any type of covert IPv6 packet. For the second layer, the multiclass KNN (m-KNN) classifier is trained to perform multiclass classification to find the header field(s) carrying covert data in

the packets classified as covert by the first layer. For training the classifier at this layer, only covert samples are fetched to create a covert sample dataset. The training phase of the b-KNN classifier for layer 1 and the m-KNN classifier for layer 2 is shown in Fig. 7.

**TABLE 2. Hyperparameters for b-KNN and m-KNN classifiers.**

| Hyperparameters | b-KNN | m-KNN |
|---|---|---|
| K | 7 | 3 |
| Distance Algorithm | Euclidean | Euclidean |

Table 2 describes the hyperparameters used for b-KNN and m-KNN classifiers, which are chosen after rigorous experimentation.

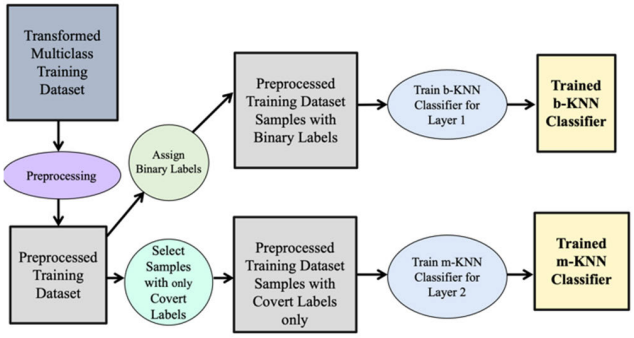



**FIGURE 7. Training phase of the classifiers for the first and second layer of SPYIPv6.**

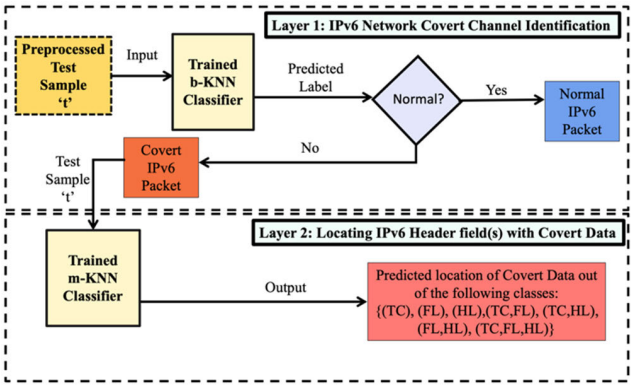



**FIGURE 8. Testing phase of SPYIPv6.**

The testing phase of the proposed SPYIPv6 is shown in Fig. 8. Each test sample 't' is given as input to the first layer of the proposed system. If the b-KNN classifier used in layer 1 classifies sample 't' as a normal packet then it is marked as a normal packet. Whereas, if b-KNN classifies the sample 't' as a covert packet, then sample 't' is forwarded to the second layer which consists of an m-KNN classifier that finds the location of covert data in the covert packet. Fig. 9 presents the algorithm used for testing SPYIPv6.

Input: Test Sample 't'
Output: $L_x$ (Class Predicted for test sample 't')

#Binary Classification at Level 1:
1. Pass the sample 't' to b-KNN for binary classification.
2. Denote the class predicted by b-KNN as $P_x$.
3. If $P_x$ indicates Normal class then:
   a. 't' is recorded as a Normal packet with the class $L_x$ as Normal.
   else if $P_x$ indicates a Covert class then:
   # Multiclass Classification at Level 2.
   a. Sample 't' is passed to m-KNN classifier to locate hidden data in the IPv6 packet's header.
   b. The class predicted by the m-KNN classifier is marked as the final label $L_x$ for test sample 't'.
4. $L_x$ is the final predicted class for the test sample 't'.
5. End.

**FIGURE 9. Algorithm used for testing SPYIPv6.**

## V. EXPERIMENTS AND RESULTS

This system was developed using a 2.1 GHz Intel Core i7-12700 processor with 16 GB RAM on Windows 11 pro operating system. For implementation, Python version 3.9.12 was used. Version 1.0.1 of the sklearn library was used to implement and experiment with various classifiers like SVM, KNN, RF, Naïve Bayes (NB), etc. For the development of the proposed SPYIPv6, the following steps were carried out: Dataset creation, Dataset preprocessing, Training, and Validation of classifiers for both layers, and Testing of the hybrid classifier system with the *Test dataset*. Experiments were conducted with different Deep Learning and Machine Learning-based classifiers like DNN, CNN, LSTM, KNN, SVM, NB, Logistic Regression (LR), and RF algorithms to find the classifiers that give high accuracy in minimum testing time. The next subsection describes the first step which is the process of dataset creation.

### A. DATASET

The proposed SPYIPv6 was trained and tested on a generated dataset comprising normal as well as covert IPv6 packets. Firstly, the *UCSD Anonymized Internet Traces Dataset [January 2019]* [49] which was supplied by the Centre for Applied Internet Data Analysis was used to gather normal packets. This dataset consists of packet capture files. From these packet capture files, several flows were selected randomly which contained a total of 327192 real-time normal IPv6 packets. In the second step, 210000 packets were selected randomly as normal IPv6 packets out of these 327192 packets. Thirdly, for the covert packets, the following categories were needed: single-field covert packets (packets containing covert data in TC, FL, and HL fields individually), two-field covert packets (packets containing covert data in two-field combinations viz. {TC – FL}, {TC - HL} and {FL – HL}) and three-field covert packets (packets containing covert data in three-field combination viz. {TC– FL– HL}). The single-field covert IPv6 packets were obtained by utilizing a tool named *pcapStego* [50]. It takes two files as input, first, a text file containing the secret message and header location where the secret message is to be

injected, and second a packet capture file that comprises IPv6 packets, into which the secret data is to be injected. These packet capture file containing IPv6 packets were obtained from the same *UCSD Anonymized Internet Traces Dataset* which consisted of 37 large flows. *pcapStego* injects secret data in individual fields namely TC, FL, and HL fields of an IPv6 packet keeping all other fields and values as they were originally during the real-time capture. The output of *pcapStego* is a new packet capture file that contains IPv6 packets with manipulated headers. In the fourth step, a total of 51485 covert IPv6 packets were obtained individually for TC, FL and HL fields. 30000 covert packets were randomly selected from each category and added to the final dataset. In the fifth step, the two-field and three-field covert packets were generated with Python scripts using the same technique as used in *pcapStego* tool. The Python scripts made use of the *Scapy* library [51] to inject data into the above-said combinations of header fields in IPv6 packets. The Python scripts took as input a .pcap file containing IPv6 packets (in which secret data is to be injected) which were obtained from the same *UCSD Anonymized Internet Traces Dataset* and output a new packet capture file containing IPv6 packets with covert data in the respective header field(s). These Python scripts were also verified on a simulation over a LAN to test the successful transfer of covert packets by injecting covert messages in the above-said IPv6 header fields. A total of 51485 covert IPv6 packets belonging to each of the four categories ({TC – FL}, {TC - HL}, {FL – HL} and {TC– FL– HL}) were obtained using these scripts. 30000 covert packets were again randomly selected from each category and were added to the final dataset. Finally, the dataset consisted of 420000 IPv6 packets out of which 210000 were normal packets and 210000 were covert IPv6 packets. In the sixth step, the header fields viz. SA, DA, TC, PL, HL, NH, FL, Transport Layer Protocol, Source Port number, and Destination Port number were fetched as features from all the packet capture files obtained/created in the previous step. The fields other than the IPv6 header like Transport Layer Protocol, Source Port number, and Destination Port number were selected as features, as they are important in identifying the flow to which an IPv6 packet belongs. They were needed for making the model learn vital associations such as the FL value remains the same throughout for all the packets that belong to a flow and any deviation to this should be taken as an anomaly. Further, the following four labels were added to each sample: Normal, Covert_TC, Covert_FL, and Covert_HL. Altogether, the normal packets and all types of covert packets resulted in a complete dataset.

In the seventh step, this complete multilabel dataset was converted to a single multiclass dataset using the Label Powerset method as discussed in Section IV. Lastly, this complete dataset was further split into: the *Train_and_validate dataset* (eighty percent of the entire dataset) and the *Test dataset* (twenty percent of the entire dataset). The dataset used for the experimentation has been described in Table 3.

**TABLE 3. Description of the dataset.**

| Label | *Train_and_validate Dataset* | *Test Dataset* | Total |
|---|---|---|---|
| Normal | 168023 | 41977 | 210000 |
| Covert-Traffic Class | 23974 | 6026 | 30000 |
| Covert-Flow Label | 23985 | 6015 | 30000 |
| Covert-Hop Limit | 23912 | 6088 | 30000 |
| Covert-Traffic Class-Flow Label | 24055 | 5945 | 30000 |
| Covert-Traffic Class- Hop Limit | 23932 | 6068 | 30000 |
| Covert-Flow Label- Hop Limit | 23974 | 6026 | 30000 |
| Covert-Traffic Class-Flow Label-Hop Limit | 24145 | 5855 | 30000 |
| Total | 336000 | 84000 | 420000 |

### B. PREPROCESSING

Preprocessing is an important task to be done before applying any ML/DL algorithm to a dataset. It helps in feature scaling and thereby provides better results. In this paper, the *Train_and_validate dataset* and the *Test dataset* were preprocessed independently. The following preprocessing steps were applied to the created datasets. To begin with, the single attribute fields corresponding to the SA and DA, each having 8 octets separated by colons were broken into 8 different attributes corresponding to each field. The two categorical attributes viz. Transport Layer Protocol and NH were converted to numerical values using quantization. Whenever the dataset has a different range of values, then to give equal importance to all the attributes, scaling or standardization is done. In this paper, *standardscalar()* function from *sklearn's* [53] version 1.0.1 was used for scaling of data. Prior to training, the pre-processed *Train_and_validate dataset* was split into: the *Train dataset* which was used for training the classifier used in both layers and the *Validate dataset* which was used to validate the performance of the classifier.

The classifiers for both layers of the proposed SPYIPv6 were chosen after extensive experimentation. The following experiments were done to develop an efficient system that provides high accuracy in less testing time.

#### 1) EXPERIMENT 1: SELECTING CLASSIFIER FOR THE FIRST LAYER OF SPYIPv6

In the proposed system, the first layer performs binary classification to segregate normal and covert IPv6 packets. For performing binary classification, a dataset with binary classes was required. Hence the *Train_and_validate* multiclass dataset was transformed into a binary class dataset by keeping the class of normal samples as 0 and transforming all covert classes to 1. After this dataset transformation,

various ML and DL classifiers were experimented with and evaluated to select the classifier that gives high accuracy in efficient training and testing time. The following algorithms were experimented with: DNN, CNN, LSTM, KNN, RF, SVM, LR, and NB for binary classification. After a thorough comparison which is shown in the *Results* subsection, KNN was chosen for the binary classification task performed in layer 1.

#### 2) EXPERIMENT 2: SELECTING CLASSIFIER FOR THE SECOND LAYER OF SPYIPv6

The second layer of the proposed SPYIPv6 finds the header fields carrying covert data in covert IPv6 packets. For locating covert data, a dataset with only covert-class samples was required. Hence the multiclass *Train_and_validate dataset* (comprising both normal and covert packets) was converted to the only covert class dataset by removing all the normal samples and keeping only the covert samples. After this dataset transformation, various ML and DL algorithms were experimented and evaluated, to select a classifier that gives high accuracy in less training and testing time. As KNN was finalized as the classifier for layer 1, thus for selecting the classifier at the second layer, these combinations of classifiers viz. KNN-DNN, KNN-CNN, KNN-LSTM, KNN-KNN, KNN-SVM, KNN-LR, KNN-RF, and KNN-NB were examined.

The accuracy percentage, time taken for training, and average testing time per sample for all the above-said combinations were evaluated and are discussed in the *Results* subsection. After comparing the classifiers on these parameters, the KNN-KNN combination was selected as this outperformed its counterparts with high accuracy and comparable testing time.

### C. EVALUATION METRICS

The proposed SPYIPv6 was evaluated with the help of various performance metrics viz. accuracy, recall, precision, and F1-score using the *Test dataset*.

Accuracy is defined as the number of test samples that are classified correctly, out of all the test samples that were passed through a classifier. Equation (4) presents the formula used to compute the accuracy of a system.

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

where TN denotes True Negatives, TP denotes True Positives, FN denotes False Negatives and, FP denotes False Positives.

Recall is computed as the ratio of the number of true positives to the sum of true positives and false negatives. Equation (5) is used to compute the recall value.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

Precision is described as the ratio of the number of true positives to the sum of false positives and true positives. Equation (6) is used to compute the precision value.

$$Precision = \frac{TP}{FP + TP} \tag{6}$$

The harmonic mean of the Precision and Recall values is computed to give the F1-score. F1-score is calculated using equation (7).

$$F1 - Score = \frac{(2XPrecisionXRecall)}{Precision + Recall} \tag{7}$$

False Positive Rate (FPR) is the fraction of negative samples predicted as positive samples. It is calculated using equation (8).

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

False Negative Rate (FNR) is the fraction of positive samples predicted as negative samples. It is calculated using equation (9).

$$FNR = \frac{FN}{FN + TP} \tag{9}$$

The evaluation of the experiments was done based on the above-stated parameters and the results are discussed in the following subsection.

### D. RESULTS

This subsection discusses the results obtained after conducting various experiments described above. The first experiment was conducted to select a suitable classifier for the first layer that provided high accuracy with less testing time for detecting covert IPv6 packet. For this purpose, various ML and DL algorithms were experimented with to find the best classifier. The results of their performance with respect to accuracy percentage, time taken for training, and average testing time per sample are shown below in Fig. 10, Fig. 11, and Fig. 12. To calculate testing time, ten samples of each covert and normal class from the *zTest dataset* were selected randomly. Each of these samples was passed through the proposed system individually. An average of the testing times taken by selected test samples was noted as the average testing time per sample. After comparing all the classifiers on the above-said parameters, it was observed that for the first layer, the KNN classifier gave the best accuracy with the least training time and the third least testing time as shown in Fig. 10, Fig. 11 and Fig. 12. Thus, KNN was finalized as the binary classifier to be used at the first layer of SPYIPv6.

The results of the second experiment which was conducted to select a suitable classifier for the second layer with respect to accuracy percentage, time taken for training, and average testing time per sample are shown below in Fig. 13, Fig. 14, and Fig. 15. The average testing time per sample was evaluated for both normal and covert samples separately in the same way as done for classifiers experimented for the first layer. After comparing all the classifiers on these parameters, it was observed that the combination of KNN at layer 1 and KNN at layer 2 delivered the best accuracy percentage in

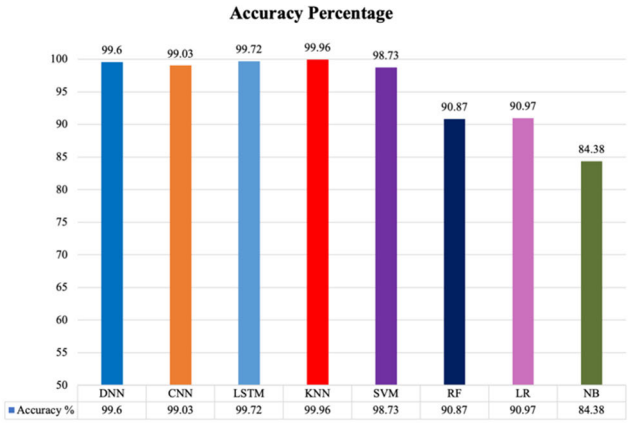



**FIGURE 10.** Accuracy percentage of different classifiers experimented for first layer.


**FIGURE 11.** Training time of different classifiers experimented for first layer.

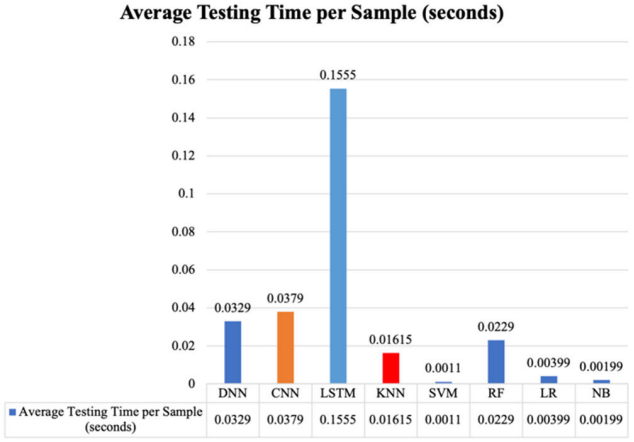



**FIGURE 12.** Average testing time per test sample of different classifiers experimented for first layer.

comparison to other combinations in consideration. Further, it took the least training time and least testing time among the classifiers for which the accuracy percentage was greater than 97. Thus, the combination of KNN at the first layer and KNN at the second layer was selected for the proposed SPYIPv6.

Further, FPR is an important metric while evaluating a classifier identifying and locating covert packets. As in real-world scenarios, covert packets are expected to be very rare. At the same time, many covert packets are usually needed to send a message, so it might suffice to detect just some of them to detect the covert communication in a flow. Therefore, the classifier needs to have an extremely low FPR while FNR can be higher.

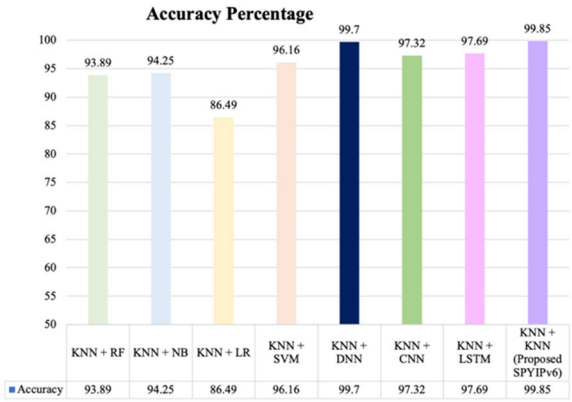



**FIGURE 13.** Accuracy percentage of different classifiers combinations experimented for the Proposed SPYIPv6.

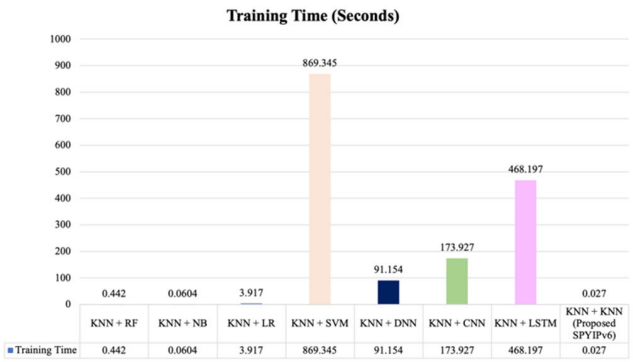



**FIGURE 14.** Training time of different classifier combinations experimented for the proposed SPYIPv6.

The FPR and FNR for all the classifiers considered at level 1 were evaluated and a comparison of the same is given in Fig. 16. It was found that KNN recorded an extremely least FPR of 0.000309 and FNR of 0.000310 further strengthening the decision to choose KNN as the level 1 classifier.

Additionally, the FPR and FNR for all the classifiers considered for level 2 were also evaluated. A comparison of the same is given in Fig. 17 and Fig. 18. It was found that KNN and DNN recorded extremely least FPR for identifying various covert classes. Between KNN and DNN, the KNN further recorded the lowest FPR for six out of eight classes, hence justifying the selection of KNN as a level 2 classifier.

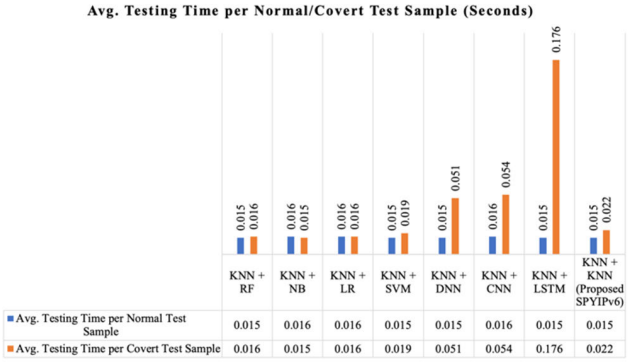



**FIGURE 15.** Average testing time per normal and covert test sample for different combinations of classifiers.

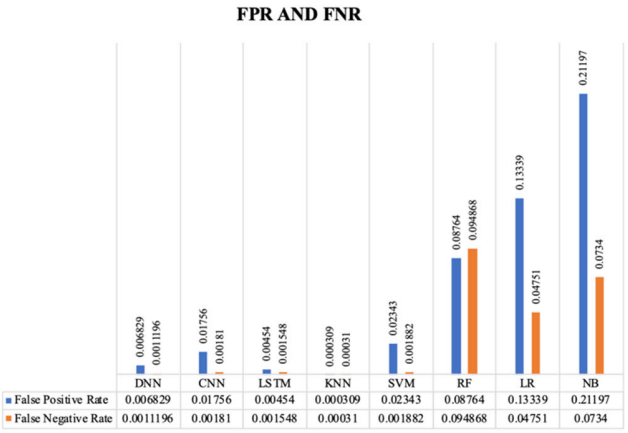



**FIGURE 16.** FPR and FNR of different classifiers experimented for the first layer.

KNN works as a suitable classifier at both the layers as it operates by finding the distances between a test sample and all the training samples in the data, selecting the K closest training samples to the test sample, then votes for the most frequent class.

## COMPARISON WITH STATE-OF-THE-ART TECHNIQUES

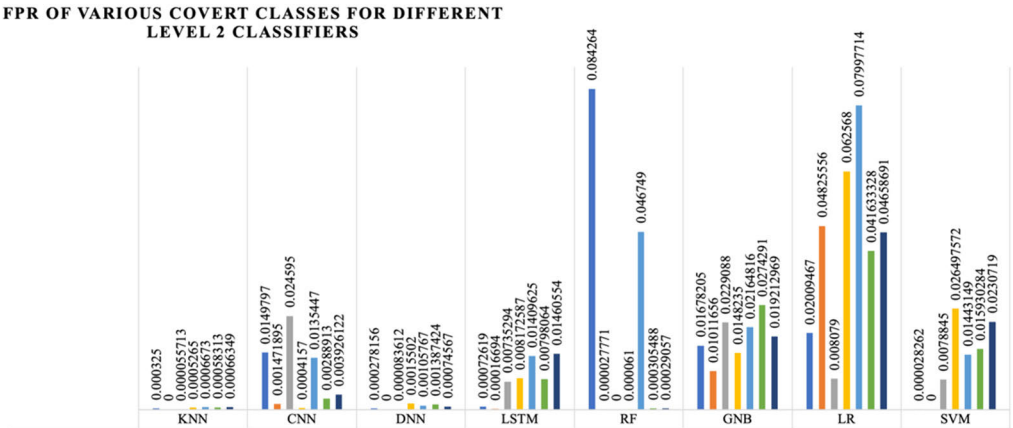
This subsection discusses the comparison of the proposed system with other related works. Some related works utilized DL or ML classifiers to detect covert communications over IPv6. Table 4 shows a concise comparison of these works with the proposed SPYIPv6. It is evident from Table 4 that the proposed SPYIPv6 obtains comparable accuracy in the least testing time among all the previous works considered for comparison in this paper. Moreover, the works done in [16] and [17] only detect covert IPv6 packets, and work done in [18] detects and locates covert data in individual fields only. Thus, the proposed SPYIPv6 outperforms all the considered previous works in terms of average testing time per covert sample as well as functionality with comparable or higher accuracy.

Other techniques in related works utilized packet filters available inside the kernel to gather statistical information that helps in detecting covert channels. Reference [13] suggested using code augmentation within the Linux kernel with the help of extended Berkley Packet Filter to obtain the statistical information about the values used in the FL field. Repetto et al. [14] presented bccstego that utilized the BPF Compiler Collection tool for gathering statistical information about TC, FL, and HL fields to detect unusual activity. The baseline concept in both [13] and [14] is to find a pattern for scrutinizing header fields of IPv6 packets and gathering statistical indicators that can predict the presence of hidden data. The advantages that our proposed technique offers over both [13] and [14] are as follows.

**TABLE 4.** Comparison of previous related works with proposed SPYIPv6.

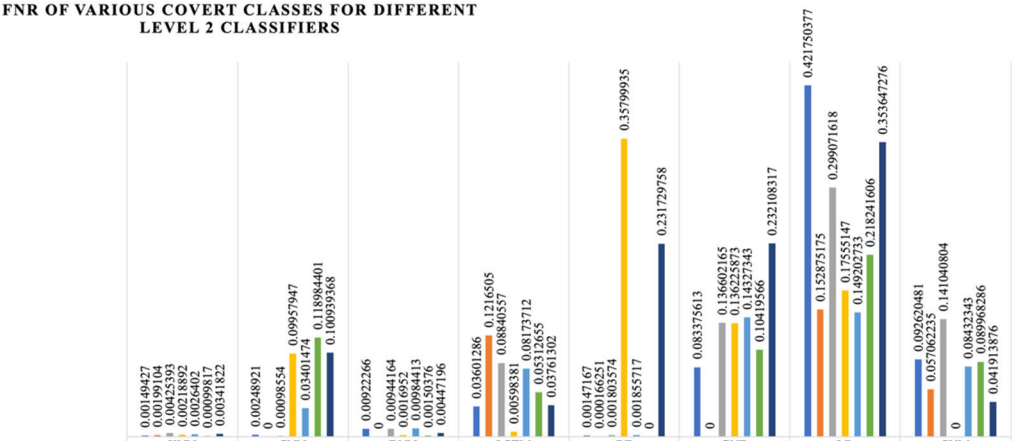
| | Accuracy Percentage | Testing time per covert sample (seconds) | Functionality (Detection/ Location Identification of Covert Data) | Types of storage-based covert channels detected/located |
|---|---|---|---|---|
| BNS_CNN [16] | 100% | 0.171 | Detection Only | Source Address based, Hop Limit based. |
| DICCh-D [17] | 99.59% | 0.0498 | Detection Only | Flow Label based, Traffic Class based. |
| Dua *et al.* [18] | 99.70% | 0.0523 | Detection and Location Identification of hidden data in single fields of IPv6 header | Traffic Class based, Hop Limit based, Flow Label based. |
| Proposed SPYIPv6 | 99.85% | 0.0222 | Detection and Location Identification of hidden data in one or a combination of header fields | Traffic Class based, Hop Limit based, Flow Label based, {Flow Label + Traffic Class} based, {Hop Limit + Traffic Class} based, { Flow Label + Hop Limit} based, {Traffic Class + Hop Limit + Flow Label} based |

Firstly, our technique proposes the detection and identification of the location of the covert data present in one or a combination of header fields viz. TC, FL, and HL fields of IPv6 packets whereas both [13] and [14] only detect the presence of a covert channel in a single field. Secondly, the eBPF-based detection mechanism used in both [13] and [14], has a limitation that the indicator viz. number of bins needs to have more granulated values for providing good accuracy which would use a large amount of memory for the node executing eBPF. Whereas no such limitation is there in the proposed SPYIPv6.




| | KNN | CNN | DNN | LSTM | RF | GNB | LR | SVM |
|---|---|---|---|---|---|---|---|---|
| Covert_TC | 0.000325 | 0.0149797 | 0.000278156 | 0.00072619 | 0.084264 | 0.01678205 | 0.02009467 | 0.000028262 |
| Covert_FL | 0 | 0.001471895 | 0 | 0.00016694 | 0.000027771 | 0.01011656 | 0.04825556 | 0 |
| Covert_HL | 0.000055713 | 0.024595 | 0.000083612 | 0.00735294 | 0 | 0.0229088 | 0.008079 | 0.0078845 |
| Covert_TC_FL | 0.0005265 | 0.0004157 | 0.0015502 | 0.008172587 | 0.000061 | 0.0148235 | 0.062568 | 0.026497572 |
| Covert_TC_HL | 0.0006673 | 0.0135447 | 0.00105767 | 0.01409625 | 0.046749 | 0.02164816 | 0.07997714 | 0.01443149 |
| Covert_FL_HL | 0.00058313 | 0.00288913 | 0.001387424 | 0.00798064 | 0.000305488 | 0.0274291 | 0.041633328 | 0.015930284 |
| Covert_TC_FL_HL | 0.00066349 | 0.003926122 | 0.00074567 | 0.01460554 | 0.00029057 | 0.019212969 | 0.04658691 | 0.0230719 |

**FIGURE 17.** FPR of different covert classes for different second-level classifiers.




| | KNN | CNN | DNN | LSTM | RF | GNB | LR | SVM |
|---|---|---|---|---|---|---|---|---|
| Covert_TC | 0.00149427 | 0.00248921 | 0.00922266 | 0.03601286 | 0.00147167 | 0.083375613 | 0.421750377 | 0.092620481 |
| Covert_FL | 0.00199104 | 0 | 0 | 0.1216505 | 0.000166251 | 0 | 0.152875175 | 0.057062235 |
| Covert_HL | 0.00425393 | 0.00098554 | 0.00944164 | 0.08840557 | 0.001803574 | 0.136602165 | 0.299071618 | 0.141040804 |
| Covert_TC_FL | 0.00218892 | 0.09957947 | 0.0016952 | 0.00598381 | 0.35799935 | 0.136225873 | 0.17555147 | 0 |
| Covert_TC_HL | 0.0026402 | 0.03401474 | 0.00984413 | 0.08173712 | 0.001855717 | 0.14327343 | 0.149202733 | 0.08432343 |
| Covert_FL_HL | 0.00099817 | 0.118984401 | 0.00150376 | 0.05312655 | 0 | 0.10419566 | 0.218241606 | 0.089968286 |
| Covert_TC_FL_HL | 0.00341822 | 0.100939368 | 0.00447196 | 0.03761302 | 0.231729758 | 0.232108317 | 0.353647276 | 0.041913876 |

**FIGURE 18.** FNR of different covert classes for different second-level classifiers.

Thirdly, the proposed technique can be used for detecting and identifying the location of any length of covert messages whereas using the methodology proposed in both [13] and [14] cannot detect small-length covert communications. Lastly, the authors pointed out a limitation in [14] that it is less effective for detecting covert channels developed using the HL field of the IPv6 header, whereas the proposed SPYIPv6 detected the existence of hidden data in the same field with high accuracy.

In addition, the proposed SPYIPv6 was also compared with the IPv6-based NCCs detection and location system developed using only a single multiclass KNN classifier. The comparison was done on the basis of precision, F1-score, and recall values. It was observed that the proposed SPYIPv6 outperformed the single multiclass KNN classifier with respect to precision, F1-score, and recall values for mostly all the classes. The results of the same are shown in Fig. 19, Fig. 20, and Fig. 21.

From all the results discussed above, it is clear that the proposed SPYIPv6 performs the detection of covert data along with the locating of covert data present in one or a combination of header fields of storage-based NCCs over IPv6 with high accuracy in an efficient time. Moreover, the applications which only need to identify whether an IPv6

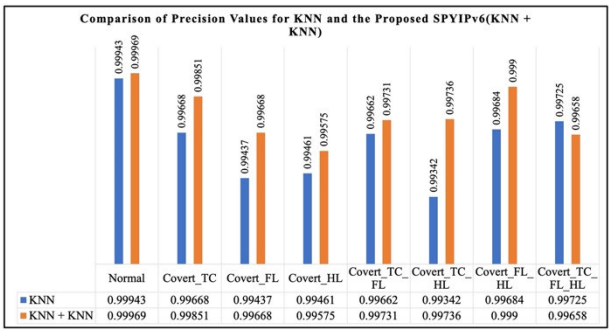



**FIGURE 19.** Precision values-based comparison for KNN and the proposed SPYIPv6 corresponding to each class.

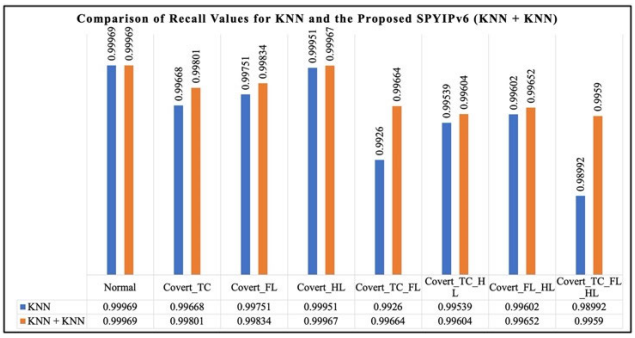



**FIGURE 20.** Recall values-based comparison for KNN and the proposed SPYIPv6 corresponding to each class.

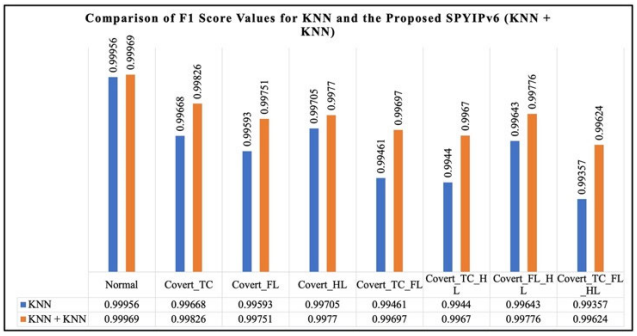



**FIGURE 21.** F1-Score values-based comparison for KNN and the proposed SPYIPv6 corresponding to each class.

packet is normal/covert can only use the predictions made by layer 1, and the applications which need to know the location of hidden data can further use the predictions made by layer 2 to identify the header fields carrying secret data. Thus, the proposed SPYIPv6 provides a complete solution, making it suitable to be used in real-world networks for detecting and further locating covert data present in one or a combination of header fields of an IPv6 packet.

## VI. CONCLUSION

Modern-day attackers mostly try to target younger protocols that are upcoming and whose security aspects are still being researched. With the development of modern threats like stegomalware, there is an urgent need to develop countermeasures to overcome such threats on prominent upcoming protocols like IPv6. For this, a two-layered-based sequential system that detects and identifies the location of hidden data present in one or a combination of the header field(s) of storage-based NCCs over IPv6 protocol has been proposed in this paper.

The dataset needed to train, validate, and test the proposed SPYIPv6 was generated. The normal packets were gathered randomly from a *zCAIDA's Dataset* whereas the covert packets were obtained using Python scripts and the *pcapStego* tool. Rigorous experimentation was done to find efficient classifiers with respect to parameters viz. accuracy, testing time, and training time. Various ML and DL classifiers were evaluated and compared, for selecting appropriate classifiers for the two layers of the proposed SPYIPv6. The proposed system predicted accurate results for detecting any length of covert messages which was one of the limitations of eBPF-based detection techniques discussed in the literature.

The strength of this paper is that this work further added a significant contribution towards detecting as well as identifying the location of covert data in either one or any combination of IPv6 header fields viz. FL, TC, and HL.

Further, the generalization ability of the proposed SPYIPv6 was assessed using *Test dataset* that was generated and used only to evaluate the system. With this test dataset, SPYIPv6 reported 99.85% accuracy in detecting and finding the location of hidden data present in one or a combination of the header field(s) of a covert IPv6 packet. In terms of training time and average testing time per test sample, the proposed SPYIPv6 took approximately 0.027 seconds to train and 0.0222 seconds to provide the location of hidden data. Due to the limitation of hardware, the experiments could not be run on parallel processors like GPU which can further reduce the training and testing times. The same can be taken up as a future work.

On the whole, the proposed SPYIPv6 can be considered an efficient and consolidated solution for detecting as well as locating covert data present in one or a combination of header fields in covert communications developed using the IPv6 protocol.

## REFERENCES

[1] G. Lencse and Y. Kadobayashi, "Comprehensive survey of IPv6 transition technologies: A subjective classification for security analysis," *IEICE Trans. Commun.*, vol. E102.B, no. 10, pp. 2021–2035, 2019.

[2] A. J. Jara, L. Ladid, and A. F. Gomet-Skarmata, "The Internet of Everything through IPv6: An analysis of challenges, solutions and opportunities," *J. Wireless Mobile Netw. Ubiquitous Comput. Dependable Appl.*, vol. 4, no. 3, pp. 97–118, 2013.

[3] K. Fujiwara, M. Shigeno, and U. Sumita, "A new approach for developing segmentation algorithms for strongly imbalanced data," *IEEE Access*, vol. 7, pp. 82970–82977, 2019.

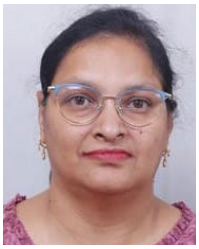
[4] W. Mazurczyk and L. Caviglione, "Information hiding as a challenge for malware detection," *IEEE Secur. Privacy*, vol. 13, no. 2, pp. 89–93, Mar. 2015.

[5] 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.

[6] L. Caviglione, "Trends and challenges in network covert channels countermeasures," *Appl. Sci.*, vol. 11, no. 4, p. 1641, Feb. 2021.
[7] W. Mazurczyk, P. Szary, S. Wendzel, and L. Caviglione, "Towards reversible storage network covert channels," in *Proc. 14th Int. Conf. Availability, Rel. Secur. (ARES)*, New York, NY, USA, Aug. 2019.
[8] X. Zhang, C. Liang, Q. Zhang, Y. Li, J. Zheng, and Y.-A. Tan, "Building covert timing channels by packet rearrangement over mobile networks," *Inf. Sci.*, vols. 445–446, pp. 66–78, Jun. 2018.
[9] Google. (2023). *Google IPv6*. Accessed: May 6, 2023. [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] W. Mazurczyk, K. Powójski, and L. Caviglione, "IPv6 covert channels in the wild," in *Proc. 3rd Central Eur. Cybersecurity Conf.*, Munich, Germany, Nov. 2019, pp. 1–6.
[12] 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. 98–110.
[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.*, New York, NY, USA, Aug. 2021, pp. 1–7.
[15] M. Zuppelli, M. Repetto, A. Schaffhauser, W. Mazurczyk, and L. Caviglione, "Code layering for the detection of network covert channels in agentless systems," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 2282–2294, Sep. 2022.
[16] D. Zhao and K. Wang, "BNS-CNN: A blind network steganalysis model based on convolutional neural network in IPv6 network," in *Proc. Int. Workshop Digit. Watermarking*, 2019, pp. 365–373.
[17] A. Dua, V. Jindal, and P. Bedi, "DICCh-D: Detecting IPv6-based covert channels using DNN," in *Proc. 7th Int. Conf. Inf., Commun. Comput. Technol. (ICICCT)*, Delhi, India, 2022, pp. 42–53.
[18] A. Dua, V. Jindal, and P. Bedi, "Detecting and locating storage-based covert channels in IPv6," *IEEE Access*, vol. 10, pp. 110661–110675, 2022.
[19] S. Deering and R. Hinden. (2017). *Internet Protocol Version 6 (IPv6) Specification*. Accessed: Jan. 16, 2023. [Online]. Available: https://tools.ietf.org/html/rfc8200
[20] S. Zhang, "Cost-sensitive KNN classification," *Neurocomputing*, vol. 391, pp. 234–242, May 2020.
[21] S. Kotsiantis, L. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," in *Emerging Artificial Intelligence applications in Computer Engineering*. Amsterdam, The Netherlands: IOS Press, 2007, pp. 3–24.
[22] S. J. Gustavus, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology*. Boston, MA, USA: Springer, 1984, pp. 51–67.
[23] 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.
[24] T. Zhang, B. Li, Y. Zhu, T. Han, and Q. Wu, "Covert channels in blockchain and blockchain based covert communication: Overview, state-of-the-art, and future directions," *Comput. Commun.*, vol. 205, pp. 136–146, May 2023.
[25] Z. Wang, L. Zhang, R. Guo, G. Wang, J. Qiu, S. Su, Y. Liu, G. Xu, and Z. Tian, "A covert channel over blockchain based on label tree without long waiting times," *Comput. Netw.*, vol. 232, Aug. 2023, Art. no. 109843.
[26] Z. Guo, L. Shi, M. Xu, and H. Yin, "MRCC: A practical covert channel over Monero with provable security," *IEEE Access*, vol. 9, pp. 31816–31825, 2021.
[27] Z. Chen, L. Zhu, P. Jiang, C. Zhang, F. Gao, J. He, D. Xu, and Y. Zhang, "Blockchain meets covert communication: A survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2163–2192, 4th Quart., 2022.
[28] P. Zorawski, L. Caviglione, and W. Mazurczyk, "A long-term perspective of the Internet susceptibility to covert channels," *IEEE Commun. Mag.*, early access, Mar. 7, 2023, doi: 10.1109/MCOM.011.2200744.
[29] 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.
[30] 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.
[31] R. Flowers, "Performance impact of header-based network steganographic countermeasures," *IEEE Access*, vol. 10, pp. 92446–92453, 2022.
[32] X.-G. Zhang, G.-H. Yang, and X.-X. Ren, "Network steganography based security framework for cyber-physical systems," *Inf. Sci.*, vol. 609, pp. 963–983, Sep. 2022.
[33] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, Oct. 1973.
[34] 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.
[35] 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.
[36] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-based survey and categorization of network covert channel techniques," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 1–26, Apr. 2015.
[37] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 44–57, 3rd Quart., 2007.
[38] 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.
[39] P. Bedi and A. Dua, "ARPNetSteg: Network steganography using address resolution protocol," *Int. J. Electron. Telecommun.*, vol. 66, no. 4, pp. 671–677, 2020.
[40] 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.
[41] 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.
[42] 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.
[43] 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.
[44] C. D. Xuan and L. V. Duong, "A new approach for network steganography detection based on deep learning techniques," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 7, pp. 37–42, 2021.
[45] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka, "An improved algorithm for neural network classification of imbalanced training sets," *IEEE Trans. Neural Netw.*, vol. 4, no. 6, pp. 962–969, 1993.
[46] M. Chourib, "Detecting selected network covert channels using machine learning," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2019, pp. 582–588.
[47] D. X. Cho, D. T. H. Thuong, and N. K. Dung, "A method of detecting storage based network steganography using machine learning," *Proc. Comput. Sci.*, vol. 154, pp. 543–548, Jan. 2019.
[48] 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.
[49] (2023). *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: Feb. 11, 2023. [Online]. Available: https://www.caida.org/data/passive/passive_dataset
[50] 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.*, New York, NY, USA, Aug. 2021, pp. 1–8.
[51] (2023). *Scapy*. Accessed: Feb. 14, 2023. [Online]. Available: https://scapy.net
[52] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, "Comprehensive comparative study of multi-label classification methods," *Expert Syst. Appl.*, vol. 203, Oct. 2022, Art. no. 117215.
[53] 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, no. 85, pp. 2825–2830, 2011.

**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, and the Ph.D. degree in computer science from the University of Delhi, in 1999.

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



**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 Ph.D. degree in computer science from the University of Delhi, in 2018.

From July 1999 to July 2001, she was the Manager/Senior Faculty Member with PCTI Ltd. From June 2017 to May 2019, she was the Head of the Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, where she has been a Professor, since November 2021. She is mainly involved 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, vehicular adhoc networks, and cloud security to name a few.

**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 with the University of Delhi.

From June 2009 to July 2010, she was a Software Engineer with Altran [formerly known as Aricent Technologies (Holdings)]. From August 2010 to January 2011, she was an Assistant Professor with Keshav Mahavidhyalaya, University of Delhi. Since February 2011, she has been an Assistant Professor with the Department of Computer Science, Bhaskaracharya College of Applied Sciences, University of Delhi, where she is currently the Head. Her research interests include network steganography and its detection, cybersecurity, information hiding, network covert channels, and their detection.