Journal of Innovations in Engineering Education
JIEE 2026, Vol. 9, Issue 1.

https://doi.org/10.3126/jiee.v9i1.90356

THAPATHALI CAMPUS
Institute of Engineering
Tribhuvan University

Detecting DNS tunneling-based data exfiltration using machine learning

Saroj Gautam[a], Binod Sapkota[a,*], Babu R. Dawadi[b] and Utkarsha Shukla[a]

[a]*Department of Electronics and Computer Engineering, Thapathali Campus, Tribhuvan University, Kathmandu, Nepal*
[b]*Department of Electronics and Computer Engineering, Pulchowk Campus, Tribhuvan University, Kathmandu, Nepal*

ARTICLE INFO

Article history:

Received 31 January 2026
Revised in 16 February 2026
Accepted 1 April 2026

Keywords:

Convolutional Neural Network
Data exfiltration
DNS tunneling
Support Vector Machine

Abstract

The misuse of DNS tunneling for covert data exfiltration is a significant threat to network security because DNS traffic is typically trusted and allowed through firewalls without deep inspection. Attackers exploit this protocol by embedding encoded payloads within DNS queries and responses, enabling them to bypass traditional security mechanisms that struggle to identify such stealthy, low-and-slow exfiltration patterns. This paper presents a hybrid detection framework that combines a one-dimensional Convolutional Neural Network (1D-CNN) for automated feature extraction with a Support Vector Machine (SVM) classifier for robust decision-making. The model is trained and evaluated on the BCCC-CIC-Bell-DNS-EXF dataset, which contains diverse benign, light, and heavy exfiltration samples generated using real tunneling tools such as Iodine. Experimental results demonstrate high effectiveness, with the CNN+SVM approach achieving up to 99.91% accuracy, minimal false positives, and consistent performance under 5-fold cross-validation. Furthermore, an NFQUEUE-based real-time prototype demonstrates live DNS packet interception and online feature computation, and enables low-latency (millisecond-level) decision-making using a baseline SVM classifier trained on entropy feature(s) in our test environment (approximately 1.4–2.4 ms per query). Overall, this paper provides an effective approach for DNS tunneling detection and establishes a foundation for future work, including integration of the proposed model into real-time operation and extensions to encrypted DNS protocols and explainable detection.

1. Introduction

The Domain Name System (DNS) is a core Internet protocol that translates domain names into IP addresses. DNS traffic is often poorly protected, allowing attackers to exploit it via DNS tunneling, where malicious data is encoded within DNS queries and responses. Traditional firewalls and intrusion detection systems (IDS) struggle to detect such attacks, as the traffic appears legitimate and uses standard DNS ports (UDP/TCP 53). Consequently, machine learning approaches are being explored to analyze DNS traffic patterns and detect tunneling effectively.

Existing DNS tunneling detectors largely follow two directions: (i) manual feature–based machine learning (e.g., entropy/length/statistics), which can miss subtle character-level patterns in low-and-slow or obfuscated tunneling, and (ii) standalone deep-learning classifiers, which may require careful calibration and can be less stable when training data are limited or imbalanced. As a result, there is a gap for an approach that can automatically learn discriminative representations from raw DNS query strings while maintaining a robust and well-calibrated decision boundary to reduce false positives. To address this gap, this paper proposes a hybrid 1D-CNN + SVM framework, where the CNN performs automated feature learning and the SVM provides robust margin-based classification.

Despite advances in network security, DNS tunneling remains a low-visibility, high-risk technique for data exfiltration. A key reason is that if current systems fails to detect these threats or produce high false positives is the lack of contextual understanding of DNS communication.

The problem being solved by the proposed work is listed

*Corresponding author:
replybinod@gmail.com (B. Sapkota)

below:

1. To design and train a hybrid 1D-CNN and SVM–based model for detecting DNS tunneling–based data exfiltration.
2. To implement a real-time monitoring prototype that captures live DNS traffic and supports low-latency baseline detection.

The proposed approach leverages a 1D-CNN to automatically extract discriminative features from DNS traffic and uses an SVM for classification. It is evaluated on the BCCC-CICBell-DNS-EXF benchmark dataset [1], which contains benign and tunneling traffic generated using tools such as Iodine. The method is intended for enterprise DNS monitoring, and a real-time NFQUEUE prototype is developed to demonstrate inline traffic interception and low-latency processing, providing a foundation for future real-time deployment of the proposed model.

The proposed work does not cover real-time deployment related features such as integration with SIEM systems, nor does it address evasion methods like encrypted DNS over HTTPS (DoH) in this phase. Additionally, the approach may incur higher computational overhead compared to traditional machine learning-only models.

The novelty of this work lies in combining deep learning–based feature learning with an SVM classifier for robust DNS tunneling detection. The work focuses on a newly available dataset on DNS exfiltration attacks (CIC-Bell-DNS EXF, 2024), which offers a large number of attack scenarios. Research work on the "perfection of "low-and-slow" attacks, remains a problem area for many machine learning models. Moreover, this work aims to create lean methods of detection that can be used in resource-constrained settings.

This paper is organized as follows: Section 1 introduces the research background, motivation, problem statement, objectives, scope, applications, and originality. Section 2 reviews related work on DNS tunneling detection. Section 3 describes the proposed methodology, including the hybrid 1DCNN–SVM model, entropy-based features, system architecture, and dataset. Section 4 presents the experimental results, discusses the findings, and compares them with existing methods. Section 5 presents the conclusion.

2. Literature review

Early DNS tunneling detection research focused on enterprise and mobile networks, where attackers evade mechanisms like the Policy and Charging Enforcement Function (PCEF) by embedding covert data in DNS queries. Recent work has focused on lightweight and deployable detection frameworks.

Mahdavifar et al. [1] proposed a two-layer hybrid model using stateless features for rapid screening and stateful features for deeper analysis which was evaluated with a Random Forest classifier on the CIC-Bell-DNSEXF-2021 dataset. The model achieved 99.97% accuracy and demonstrated suitability for edge deployment. However, the work depends on comprehensive DNS logs, which may be impractical in high-speed or privacy-sensitive environments.

Zhang et al. [2] integrated a CNN-based DNS tunneling detector into a broader intrusion detection framework.

Van Thuan Do et al. [3] noted that traditional defenses like firewalls, intrusion detection systems, passive DNS replication, and traffic analysis are mostly signature-based and ineffective against adaptive tunneling. They applied anomaly detection using One-Class SVM and K-Means clustering, showing that OC-SVM with an RBF kernel achieved an F1 score of about 96%, outperforming K-Means.

Subsequent studies increasingly adopted machine learning to enhance DNS tunneling detection. Das et al. [4] applied K-means clustering to DNS TXT records but reported limited performance due to a narrow feature space.

Later work demonstrated that supervised learning generally outperforms unsupervised methods for anomalous DNS detection.

Ahmed et al. [5] proposed an Isolation Forest–based approach using stateless features such as entropy and character distribution, achieving up to 98% accuracy; however, it assumed unencrypted DNS traffic and relied on manual feature engineering, limiting adaptability.

To overcome these limitations, deep learning methods were introduced.

Lai et al. [6] used a feedforward neural network trained on raw packet bytes, achieving 99.96% accuracy, while Liu et al. [7] proposed a byte-level CNN with 99.98% accuracy.

Hybrid architectures have been proposed to combine automatic feature learning with robust decision boundaries.

Lal et al. [8] introduced DNS-Tunnel, a CNN–SVM framework that integrates CNN-based representation

learning with SVM classification, achieving a state-of-the-art F1 score of 99.98% and outperforming ensemble methods such as Random Forests and Voting Classifiers without manual feature extraction.

Efforts to move beyond binary classification include Bubnov [9], who formulated DNS tunneling detection as a multi-label classification problem using a feedforward neural network with handcrafted features such as entropy, query length, and resource record type, achieving 83% accuracy.

Similarly, Sammour et al. [10] compared classical classifiers including SVM, Naive Bayes, and J48 Decision Trees using traffic and payload-based features, with SVM attaining the highest F-measure of 83%. Nonetheless, these methods rely on static statistical features and thresholds, making them susceptible to evasion by adaptive adversaries.

Abualghanam et al. [11] note that DNS tunneling enables covert data exfiltration by abusing the permissive nature of DNS traffic, making signature-based detection ineffective. Although ML and DL methods report high accuracy (94–99.9%), which relies on large feature sets, labeled data, and high computational cost, limiting real-time use. Recent hybrid approaches mitigate this by combining lightweight statistical features with optimized feature selection while preserving strong performance.

Liang et al. [12] stated that DNS tunneling exploits the permissive nature of DNS traffic to enable covert data exfiltration and command-and-control communication, rendering early feature-engineering–based detection methods fragile and easily evaded. Convolutional neural network (CNN)-based approaches improve robustness by automatically learning discriminative features from raw DNS traffic, even under encryption and encoding.

Zhan et al. [13] proposed DNS tunneling, which exploits the permissive nature of DNS and, with DNS over HTTPS (DoH), allows attackers to hide exfiltration traffic within encrypted channels. Recent flow- and metadata-based approaches using TLS fingerprints and traffic features show effective detection of DoH tunneling. However, these methods face robustness issues under diverse network conditions, evolving DoH implementations, and adversarial evasion such as fingerprint spoofing, highlighting a gap in developing scalable, privacy preserving, and adversary-resilient DoH detection frameworks that generalize beyond controlled settings.

Pitch et al. [14] proposed a low-latency, passive DoH tunneling detection method based on features extracted from a single encrypted packet, achieving up to 99.93% accuracy using a Random Forest classifier. While the results highlight the effectiveness of packet-level analysis for encrypted DNS traffic, the evaluation is limited to controlled datasets.

Ding et al. [15] proposed a semi-supervised VAE with bidirectional GRU and attention to detect encrypted DNS tunnels over DoH using flow-level features, achieving 99% accuracy without manual feature engineering.

Isik et al. [16] introduced DNS Sentinel, a hybrid ML framework using supervised and unsupervised models to detect DNS tunneling via lexical features, achieving high recall on real and synthetic datasets.

Salat et al. [17] highlighted the growing misuse of DNS tunneling in cloud environments and the limitations of signature-based and traditional ML methods in detecting obfuscated or zeroday attacks. They proposed a cloud-focused framework combining entropy analysis, behavioural traffic features, and ML classifiers, improving accuracy and reducing false positives.

Sui et al. [18] analyzes network tunnel detection across multiple protocols, comparing traditional and ML-based methods, and highlight challenges such as encryption, protocol encapsulation, and fine-grained identification. While quantitative evaluations and AHP ranking are provided Adiwal et al. [19] presented DNS Intrusion Detection (DID), an extension of the SNORT IDS, designed to detect DNS-based attacks such as amplification, tunneling, and DoS. Novel IDS signatures were developed and integrated into SNORT, enabling effective detection of real-world DNS attacks.

Mitsuhashi et al. [20] addressed the challenge of detecting malicious DNS tunnel tools over encrypted DNS traffic (DoH), which limits traditional network security monitoring. They propose a hierarchical machine learning-based system that analyzes persistent DoH traffic and continuously updates its knowledge to recognize both well-known tools (dns2tcp, dnscat2, iodine) and emerging tools (dnstt, tcp-over-dns, tuns) with 98.02% classification accuracy.

Nguyen et al. [21] highlighted DNS tunneling, increasingly exploited by attackers such as Advanced Persistent Threats to bypass security systems and exfiltrate data. While unsupervised methods like DBSCAN can detect simple malicious DNS tunnels, they require manual hyperparameter tuning, limiting adaptability. This article proposes AutoRoC-DBSCAN, which automatically selects optimal hyperparameters to detect malicious DNS tunnels across new datasets. The study is limited by small datasets, restricted tunneling diversity, a lack of

real-world mobile traffic, and the absence of scalability evaluation, which raises concerns about generalization to unseen attacks and real-world environments. The binary classification setting further limits multiclass detection. Existing DNS tunneling detection methods largely depend on supervised learning, handcrafted features, static thresholds, and fixed-length representations, making them less robust to obfuscated, evolving, or encrypted attacks such as DoH. Many studies rely on imbalanced datasets that lack cross-dataset validation and report high offline accuracy in controlled settings without addressing scalability, real-time inference, or live deployment. These gaps highlight the need for scalable, low-overhead, and encryption-aware detection frameworks suitable for practical use.

3. Methodology

3.1. Theoretical formulations

3.1.1. Convolutional Neural Network (1D CNN)

A 1D CNN is well-suited for sequential data such as network traffic time-series or character sequences in DNS queries. The core operation is the 1D convolution. If the input sequence is denoted as x, which is described in Equation 1:

$$x = \{x[1], x[2], \ldots, x[n]\} \tag{1}$$

(e.g., a numerical representation of a domain name or a sequence of DNS query features over time), and a CNN filter (kernel) of width k with weights in Equation 2:

$$w = \{w_1, \ldots, w_k\} \tag{2}$$

then the convolution output at position j (pre-activation) is given in Equation (3):

$$y[j] = \sum_{i=1}^{k} (w[i] \cdot x[j+i-1] + b) \tag{3}$$

where b is a bias term. In our theoretical formulation, the 1D CNN acts as an automatic feature extractor $f_\theta(\cdot)$ parameterized by weights θ. It transforms the raw input x (DNS query string or features) into a higher-level representation $h = f_\theta(x)$; for instance, the activation of the last convolutional layer or a subsequent fully-connected layer can serve as a feature vector.

The CNN training involves minimizing a loss (e.g., cross-entropy for classification) over the labeled training data, using stochastic gradient descent or adaptive optimizers, to adjust θ such that meaningful features for distinguishing tunnel vs. normal are learned.

3.1.2. Support Vector Machine (SVM)

The SVM is a supervised learning algorithm that finds the optimal hyperplane to separate data points of different classes with maximum margin. In a binary classification setting (malicious vs. benign), and given the feature vector h from the CNN as input, the SVM's decision function can be written as given in Equation 4:

$$f(w, b; h) = w^T h + b \tag{4}$$

where w is the normal vector to the hyperplane and b is the bias. The SVM aims to find w and b that satisfy, for all training examples i in Equation 5:

$$y_i(w^T h_i + b) \geq 1, \quad \forall i, \quad \text{where } y_i \in \{+1, -1\}$$

$$\text{margin} = \frac{2}{\|w\|} \tag{5}$$

In practice, since perfect separation is rare, a soft-margin SVM is used, introducing slack variables and a penalty parameter C. The optimization problem can be formulated in Equation 6 :

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} \max\left(0, 1 - y_i(w^T h_i + b)\right) \tag{6}$$

where $y_i \in \{+1, -1\}$ are class labels (e.g., +1 for tunneling detected, −1 for benign). The term inside the sum is the hinge loss, which penalizes misclassifications or points inside the margin. Solving this convex optimization yields the optimal w^*, b^* that define the decision boundary.

Often, a kernel function $K(h_i, h_j)$ is used to implicitly map features to a higher-dimensional space when data is not linearly separable. In this work, the CNN-learned features are expected to be sufficiently linearly separable; if not, an RBF (Gaussian) or polynomial kernel is used in the SVM to capture non-linear relationships.

3.1.3. Entropy as a feature for DNS traffic analysis

Entropy, in the context of information theory, is a quantitative measure of the uncertainty or randomness present in a sequence of symbols. Introduced by Shannon (1948), it evaluates how unpredictable the occurrence of a symbol is within a dataset. In DNS tunneling detection, the entropy of domain names is a useful metric because malicious domains tend to exhibit higher randomness than legitimate, human-readable domains. Let X be

a discrete random variable representing the characters of a domain name, and let $A = \{x_1, x_2, \ldots, x_n\}$ be the set of unique characters in that domain. The Shannon entropy $H(X)$ is defined in Equation 7:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{7}$$

where $p(x_i)$ denotes the empirical probability of the character x_i, and n represents the total number of unique characters in the domain. The use of the base-2 logarithm ensures that the entropy is measured in bits. The entropy value $H(X)$ satisfies:

- $H(X) = 0$ when the domain contains only a single repeated character (completely predictable),
- $H(X)$ is maximized when all characters occur with equal probability (completely random).

In this work, entropy is used as a handcrafted baseline feature and is also computed in the NFQUEUE-based real-time prototype for low-latency detection. The proposed 1DCNN+SVM model primarily relies on automatically learned representations from the domain string.

3.1.4. Principal Component Analysis (PCA)

In this paper, Principal Component Analysis (PCA) was used to assess the discriminative capability of features learned by the proposed CNN–SVM hybrid model. The projection of high-dimensional features onto the top two principal components shows clear separation between benign DNS queries and tunneling malware. The PCA results indicate that the learned embeddings effectively capture key statistical and lexical characteristics of DNS payloads, such as entropy, character distribution irregularities, and encoded subdomain structures. The compact and well-separated feature representations confirm effective feature extraction, strong class discriminability, and the generalization capability of the model across different DNS tunneling activities.

3.2. System Block Diagram

The proposed system is designed in modular blocks, as illustrated in Figure 1.

The major components and their interconnections are:

Dataset: This module uses the BCCC-CIC-Bell-DNS-EXE dataset, a customized DNS traffic dataset for tunneling and exfiltration studies. It extends earlier CIC-Bell datasets (e.g., CIC-Bell-DNS-2021 and CIC-Bell-DNS-EXF-2021) and contains labeled benign and malicious DNS queries with associated query metadata used as input to the detection system.

Data Preprocessing: The raw DNS queries are processed through a structured preprocessing pipeline before model training. This stage extracts relevant query features, converts them into a machine-readable representation, and ensures consistent input dimensions for the deep learning model. The preprocessing workflow is shown in Figure 2.

Embedding Layer: The embedding layer maps each character's integer index to a dense, learnable vector representation. Instead of sparse one-hot encodings, characters are embedded into a d-dimensional continuous space (e.g., $d = 16$), where characters with similar contextual usage may have similar representations. These embeddings are initialized randomly and optimized during training to capture character-level patterns that help distinguish benign from malicious domain names. The output is a sequence of dense vectors, which serves as the input to the subsequent convolutional feature extractor.

1D Convolution layers and ReLU activation: One-dimensional convolutional layers extract features by sliding learnable filters over embedded domain name sequences, functioning as character-level n-gram detectors. These filters produce strong activations when encountering patterns associated with DNS tunneling, such as high-entropy substrings or unusual character combinations typical of encoded payloads. Following each convolution, the Rectified Linear Unit (ReLU) activation function introduces nonlinearity by suppressing negative responses and emphasizing informative features. Max pooling is then applied to reduce the sequence length while retaining the most salient activations, improving computational efficiency and robustness. By stacking multiple convolution, ReLU, and pooling layers, the network learns hierarchical representations, where shallow layers capture low-level character patterns and deeper layers model higher-level abstractions such as malicious domain prefixes. Overall, the one-dimensional CNN effectively distills domain name information into discriminative features that differentiate benign and suspicious DNS queries.

Fully connected layer and output feature vector: After the convolutional layers extract local and hierarchical features, a fully connected layer integrates them into a global representation of each domain. The resulting feature maps are flattened or globally pooled into a single feature vector, which is then passed through one or more dense layers for feature combination and dimensionality reduction. Since each neuron is connected to all activations from the previous layer, the network can learn global patterns and feature co-occurrences. By reducing the dimensionality of the feature space, the network produces a compact domain-level embedding that preserves

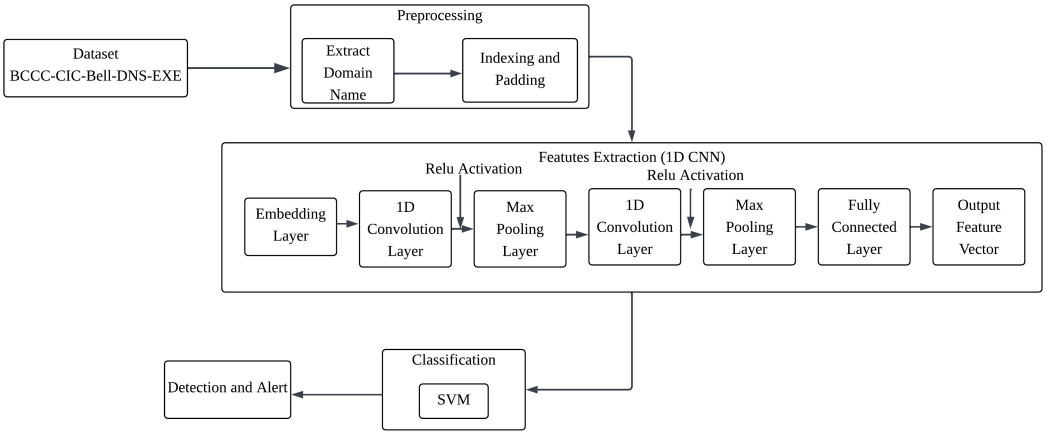

Figure 1: System block diagram

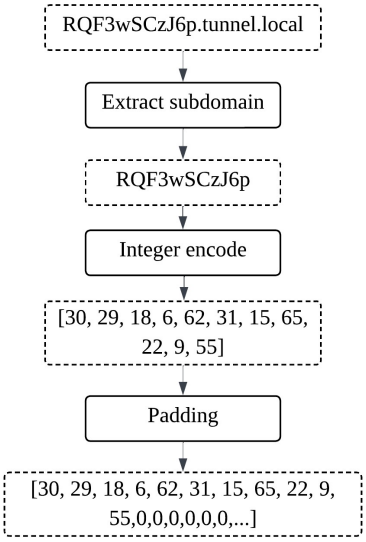

Figure 2: Preprocessing workflow

informative characteristics while discarding redundancy. This embedding acts as a learned fingerprint of the domain and, in the proposed hybrid pipeline, is provided as the input feature vector to the SVM classifier.

SVM classifier: The SVM classifier takes the feature vector produced by the neural network and performs the final binary classification between malicious DNS tunnels and benign DNS traffic. Operating in a high-dimensional feature space, the SVM learns an optimal separating hyperplane that maximizes the margin between the two classes. By focusing on support vectors, it achieves strong generalization and can model complex decision boundaries, making it well suited for binary classification tasks with limited training data.

Detection and alert: In the deployment setting, the outputs of the SVM classifier must be translated into actionable security events. The Detection and Alert block is responsible for interpreting the SVM's predictions and raising alerts when a DNS tunneling threat is identified. Each DNS query (or domain) processed through the model is assigned a prediction if the prediction is malicious, the system flags that query as an exfiltration attempt.

3.3. Network architecture of proposed system design

The architecture depicted in Figure 3 captures a typical DNS-based tunneling attack and its monitoring flow through a defensive sensor. Each component and arrow represents a distinct role in the exfiltration chain:

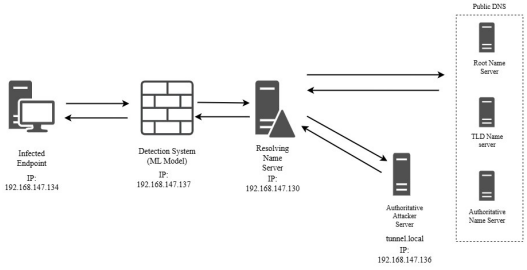

Figure 3: Proposed System Design Scenario for DNS Tunneling Detection

3.3.1. Infected endpoint

This is the compromised client machine whose DNS resolution requests carry encoded payloads. Malware on the host intercepts sensitive data like credentials,

files, keystrokes, and encodes it into the subdomain portion of each DNS query (for example, by Base64). In a benign environment, this host would make only occasional, short queries (e.g. www.example.com), but once infected, it generates high-entropy subdomain strings (e.g. adfj39dkfj3jf9j30.tunnel.local) at regular intervals to tunnel data out.

3.3.2. Detection system

This system intercepts outbound DNS queries and performs inline analysis. In the current implementation, the real-time prototype computes lightweight handcrafted features (e.g., Shannon entropy of the queried domain/subdomain) and applies a baseline decision rule/model for low-latency detection. The proposed 1D-CNN+SVM model is evaluated offline in this work; integrating it into the real-time pipeline is left for future work.

3.3.3. Resolving name solver

This is the legitimate DNS resolver that actually resolves benign queries. When the proxy forwards an approved query, the resolver performs the standard iterative lookup (contacting root, TLD, authoritative servers as needed) and returns the real IP address. The proxy then passes that answer back to the requesting endpoint. Because the resolver has no visibility into the classification logic, it operates exactly as before; only the proxy's forwarding logic changes the traffic path.

3.3.4. Authoritative attacker server

This machine hosts the attacker's authoritative DNS server for the tunneling domain (e.g., *.tunnel.local). It extracts exfiltrated data from subdomain labels in incoming queries and may embed command-and-control instructions in DNS responses, such as TXT records or crafted A/AAAA records. Blocking or spoofing NXDOMAIN responses at the proxy disrupts this covert channel, whereas allowing the queries enables bidirectional communication with the attacker's server.

3.3.5. Public DNS

The public DNS hierarchy (root name server, TLD servers and other public authoritative servers) is the global infrastructure the resolving server contacts when it must perform recursion for unknown domains; normal benign lookups causes the resolver to traverse this chain, whereas private or specially configured zones (like tunnel.local) may not touch public TLDs and instead go directly to a configured authoritative host. DNS traffic was captured using Wireshark by filtering port 53 traffic and storing the packets as PCAP files for offline analysis. Scapy was then used to parse these captures and extract relevant DNS fields, such as query names and types, during feature extraction. To generate realistic malicious traffic, DNS tunneling tools such as Iodine were deployed in a controlled setting to simulate data exfiltration through encoded DNS queries. The experiments were conducted in an isolated VMware-based virtualized network consisting of multiple virtual machines: a benign client generating normal DNS traffic, an infected client performing DNS tunneling, and a DNS server acting as the attacker endpoint. The machine learning components were implemented in Python 3 using deep learning frameworks such as TensorFlow (Keras) or PyTorch for the 1D CNN, and scikit-learn for training and evaluating the SVM classifier. Additional utilities, including shell and PowerShell scripts, were used to automate data collection.

3.4. Description of algorithms

The solution involves two main algorithms working in tandem: a One-Dimensional Convolutional Neural Network (1D CNN) and a Support Vector Machine (SVM). The following components are described below, along with how they integrate into the overall process shown in Figure 4:

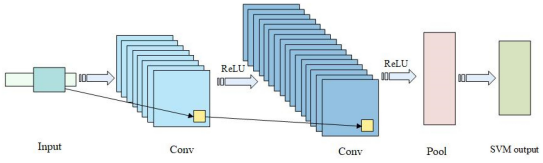

Figure 4: The improved 1D-CNN-SVM structure

3.4.1. 1D CNN for feature extraction

The 1D CNN algorithm was to be designed to process sequential data derived from DNS queries. Depending on input choice, the sequence could be:

- The character sequence of the domain name in a DNS query.
- A time-series of feature values (though typically CNN on time-series might assume some local temporal relationships).

In our approach, using the domain name string as input to the CNN is appealing, treating it similarly to how NLP models treat text. Each character (a–z, 0–9, hyphen, etc.) is mapped to an integer index (embedding index). The CNN architecture might look like:

- Embedding layer: e.g., map each character to a 50-dimensional trainable vector.
- Convolutional layer 1: 128 filters of size 3 (to capture tri-gram patterns in the domain), stride 1. Apply ReLU activation.

- Convolutional layer 2: 64 filters of size 3, stride 1. ReLU activation.
- Global Max Pooling layer: to reduce each filter's output to a single value by taking the max over the sequence (this helps capture the most significant pattern per filter across the domain string).
- Fully Connected layer: takes the pooled outputs (one per filter) and further reduce dimensionality to 32 features. This layer's output h is the feature vector passed to SVM.

Interpretation of CNN-learned discriminative features: In this character-level setting, each convolution filter acts as a learned detector for local character motifs (n-gram– like patterns) within the domain string. The global maxpooling layer makes the representation position-invariant by retaining the strongest activation of each motif anywhere in the sequence. For DNS tunneling, these learned motifs typically correspond to structural and lexical cues such as unusually long subdomains, dense alphanumeric/Base32-like runs, dot-separated repeated chunks consistent with payload fragmentation, and reduced word-like structure compared to benign domains. Therefore, the extracted feature vector captures the presence and strength of these tunneling-related patterns and is used by the SVM for final classification.

The CNN algorithm involves forward propagation through these layers to compute outputs, and a training procedure using backpropagation to adjust weights. The CNN is trained with a temporary softmax output layer during training to predict benign vs. malicious labels, using cross-entropy loss, as a proxy for feature learning. The reason is that integrating SVM in training is more complex. Once the CNN is trained (and validated to achieve adequate feature separation), the softmax layer is removed and the second-to-last layer output is used as the feature vector for the SVM.

3.4.2. SVM for classification

After CNN training, the SVM is trained using the feature vectors hi output by the CNN for each training sample. The SVM training algorithm (for a linear or kernel SVM) is typically done via quadratic programming or a stochastic gradient on the hinge loss. Scikitlearn implementation is used which automatically handles this via an efficient SMO or similar solver.

3.4.3. Hyperparameter tuning

The SVM uses an RBF kernel and tunes the regularization parameter C and the kernel-width parameter γ, since they control the bias–variance tradeoff and the decision-boundary complexity. Grid search is performed over $C \in \{0.1, 1, 10\}$ and $\gamma \in \{\text{scale}, 0.01, 0.1, 1\}$ using stratified cross-validation on the training folds, and selects the best-performing configuration within this search space.

Feature scaling is applied using StandardScaler fitted on the training fold only.

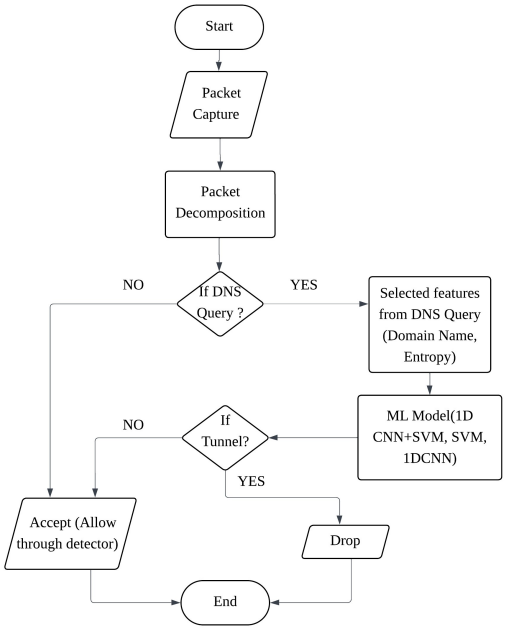

Figure 5: Flowchart of real time DNS tunneling blocking system (entropy-based baseline)

Figure 5 illustrates the working process of the real-time DNS tunneling monitoring and blocking prototype implemented using NFQUEUE. The process starts with the interception of network packets, where each incoming packet is parsed for analysis. The system first checks whether the packet contains a DNS query. Non-DNS packets are immediately accepted without further inspection to minimize overhead and preserve normal network performance. When a packet is identified as a DNS query, the system enters the online feature computation phase, where lightweight handcrafted attributes are computed from the query name, such as Shannon entropy of the domain/subdomain string. These features are then used by a baseline SVM classifier trained on entropy feature(s) to determine whether the DNS query exhibits tunneling behavior. If classified as suspicious, indicating a potential DNS tunneling attempt, the packet is dropped to prevent data exfiltration; otherwise, benign queries are accepted and forwarded normally. This decision-making process enables millisecond-level inline detection in our test environment, which is approximately 1.4-2.4 ms per query.

Integration of the proposed hybrid 1D-CNN+SVM model into the real-time pipeline is left for future work.

3.5. Verification and validation procedures

Accuracy and detection metrics: The trained model was evaluated on a hold-out test set from the BCCC-CIC-BellDNS-EXF dataset. Key metrics are as follows:

Accuracy: Overall fraction of correct classifications. Given the dataset size, a high accuracy (target >95%) is expected. However, if the dataset is unbalanced, accuracy may be deceptive (as in this case, with more malicious than benign samples). Therefore, the following metrics are emphasized from Equation 8 to 11 :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Precision (Positive Predictive Value): Out of all queries the model flags as malicious, how many were truly malicious. A high precision means few false alarms, which is important for practical use.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

Recall (True Positive Rate or Detection Rate): Out of all actual malicious tunnel queries, this metric indicates how many were correctly identified by the model. A high recall value reflects the model's ability to detect the majority of attacks. The target is in the high 90% range; however, even a recall of 95% combined with a false positive rate of 1% would be considered excellent.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

F1-Score: The harmonic mean of precision and recall, to balance the two.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

False Positive Rate: Specifically, the percentage of benign queries misclassified as malicious.

This rate should be kept very low (preferably <1%). Given DNS traffic volume in real networks, even 1% false positives could be noisy, but our design of the hybrid model is to maximize precision.

Cross-Validation and Robustness: Cross-validation (e.g., 5-fold cross-validation on training data) was used to ensure the model's performance is consistent and not the result of a single train–test split.

The real-time prototype was qualitatively validated using live-simulated DNS traffic generated in a VMware environment running iodine. DNS queries captured from a custom iodine alongside DNS queries captured from a custom iodine tunneling session were analyzed alongside benign DNS traffic generated through normal web browsing. The baseline SVM classifier trained on entropy features successfully flagged iodine-related tunneling queries while producing few false alarms on legitimate background traffic in our test environment. This provides qualitative validation of the real-time monitoring prototype in a realistic scenario.

4. Results and analysis

4.1. Experimental setup

In the experimental setup, a dataset of DNS queries was assembled from the BCCC-CIC-Bell-DNS-EXF dataset. Benign samples were labeled as 0, and tunneling samples as 1. The records were then concatenated and shuffled, yielding approximately 370,189 total samples (117,982 benign and 252,207 malicious). Each sample consists of the queried domain name, which is treated as a character sequence. The specific features of the dataset and model parameters are displayed in Table 1, 2, and 3 below.

Table 1: Dataset composition and train/test/validation split

Dataset	Number of domain
BCCC-CIC-Bell-DNS-EXF	370189
Training Dataset (70%)	266536
Test Dataset (20%)	74037
Validation Dataset (10%)	29615

Table 2: Class distribution in dataset

Class	Samples
Benign	117982
Attack	252207

4.2. Training results of CNN and SVM models using domain name feature

4.2.1. Model accuracy

The accuracy curves show that after a very large jump between the first two epochs, both the training and validation curves quickly converge above 99.8% and then

Table 3: Model parameter

Parameter	Value
Optimizer	Adam
Learning rate	1×10^{-3}
Batch size	128
Maximum epochs	10
Early stopping	Monitor validation loss, patience = 2, restore best weights
Validation split	10% of training set
SVM kernel	RBF
SVM hyperparameter tuning	Grid search with stratified cross-validation
C (search space)	$\{0.1, 1, 10\}$
γ (search space)	$\{\text{scale}, 0.01, 0.1, 1\}$
Feature scaling	StandardScaler (fit on training fold only)

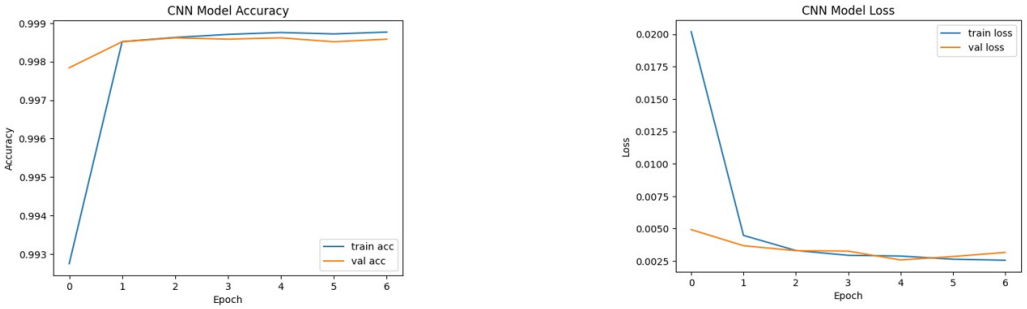

Figure 6: CNN model accuracy and loss curve

slowly inch upward toward 99.9% over the remaining epochs. The tiny gap between the blue (train) and orange (validation) lines indicates the model is fitting the data extremely well with minimal overfitting. Overall, this plot demonstrates that the CNN is highly accurate and stable, but also that additional regularization or early stopping might be used to prevent redundant epochs once the curves level off in Figure 6.

4.2.2. Model loss

The loss curve shows a very sharp drop in training loss from about 0.0175 in epoch 0 down to roughly 0.004 by epoch 1, indicating that the network quickly learns a good initial representation of the data. After that first major jump, the training loss continues to decrease more gradually, leveling off around 0.0025 by epoch 7. Overall, this curve demonstrates stable convergence: the network quickly captures the bulk of the signal in the first two epochs, then fine-tunes more slowly thereafter, with both training and validation losses settling to low, closely matched values in Figure 6.

4.2.3. ROC-AUC curve

The ROC curve illustrates the strong performance of the model in detecting DNS tunneling activity. The curve rises sharply toward the upper-left corner, indicating a high true positive rate with minimal false positives. An AUC value of 1.00 confirms perfect separation between benign and tunneled DNS queries based on the extracted features. This demonstrates the effectiveness of DNS traffic behavioral characteristics and highlights the robustness of the proposed machine learning approach for detecting DNS tunneling–based data exfiltration, as shown in Figure 7.

4.3. PCA projection of CNN-extracted features

The 2D PCA plot in Figure 8 shows the CNN feature representations projected onto the first two principal components (PC1 and PC2). Each point represents a sample, colored by its binary class label (0–1). The well-separated, compact clusters indicate that the CNN learned highly discriminative features with low intra-class variability and a large inter-class margin, demonstrating strong class separability on the benchmark dataset.

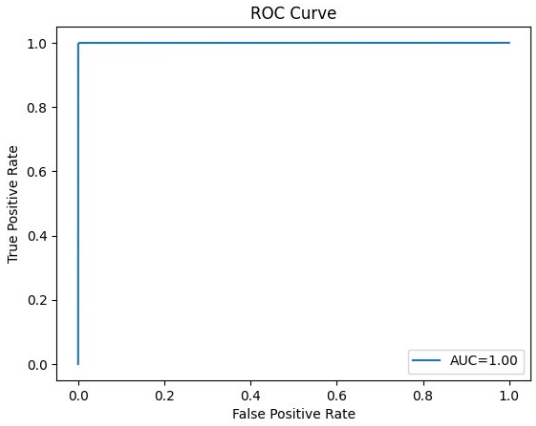

Figure 7: ROC-AUC curve

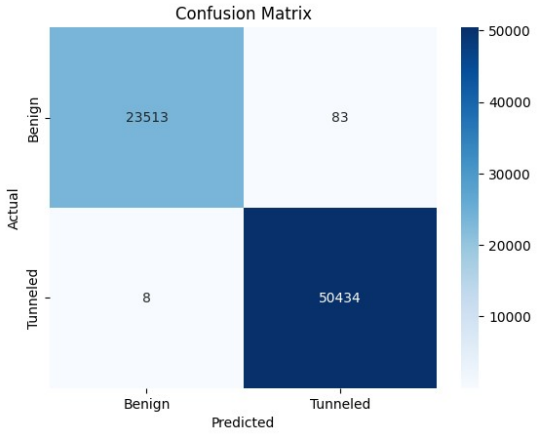

Figure 9: Confusion matrix of SVM classifier

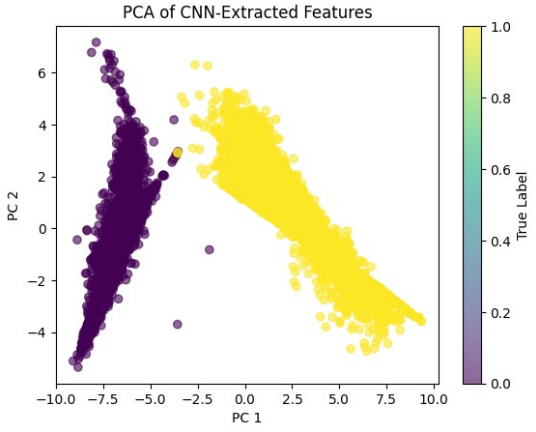

Figure 8: PCA of CNN-extracted features

The confusion matrix in Figure 9 shows strong detection performance, correctly classifying 23,513 benign DNS samples and 50,434 tunneled queries. Only 83 benign queries were misclassified (0.38% false positives), and just 8 tunneling attempts were missed (approximately 0.01% false negatives), indicating high accuracy and effective DNS tunneling detection.

4.4. Evaluation of DNS traffic using 5-fold cross-validation

The confusion matrix in Figure 10 shows a representative fold (fold 5) from the 5-fold stratified cross-validation (with shuffling) using the Support Vector Machine (SVM) classifier. In this fold, the model correctly identified 23,525 benign samples and 50,411 tunneled samples. Misclassifications were minimal, with 71 benign instances incorrectly classified as tunneled (false positives) and 30 tunneled instances predicted as benign (false negatives). These results indicate strong classification performance within this fold, characterized by high precision and recall and low false-positive/false-negative rates. We note that stratified random cross-validation may still allow overlap of similar domain patterns across folds; therefore, results are interpreted as performance on the benchmark dataset, and group-aware validation is left for future work.

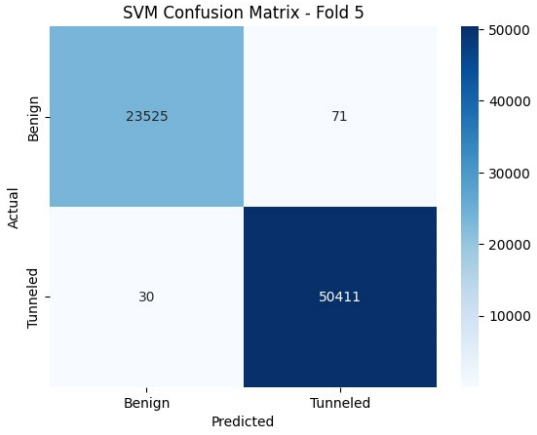

Figure 10: Confusion matrix-fold 5

4.5. Performance comparison of different models using domain name and entropy features for DNS tunneling detection

The experimental results in Table 4 show that all tested models are capable of achieving near-perfect detection performance on the DNS tunneling classification task. However, the best performance was observed when combining a 1D-CNN with an SVM classifier, indicating the benefit of using deep learning for feature extraction followed by a strong classifier. Moreover, even simpler models using entropy as a single feature provide high performance, which makes them suitable for real-time

or resource-constrained environments.

4.6. Lab implementation of detecting DNS tunneling

Figure 11 represents the implementation of detecting DNS tunneling.

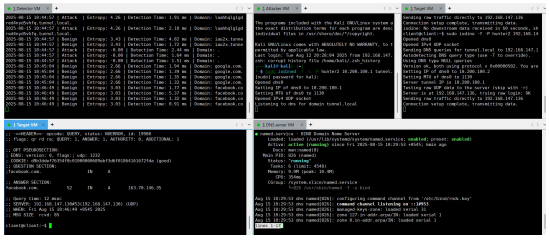

Figure 11: DNS tunneling attack and detection setup showing Attacker VM, Target VM, DNS Server VM, and Detector VM

4.6.1. Detector VM

The Detector VM is responsible for continuously monitoring DNS traffic and computing key features such as entropy and domain name patterns. The terminal output shows real-time detection results where each DNS query is classified as either Attack or Benign. For every captured packet, the system calculates entropy, measures the detection time in milliseconds, and extracts the queried domain name. High-entropy domains and unusual query patterns are flagged as suspicious, demonstrating the effectiveness of the real-time baseline SVM (entropy feature) deployed in this VM.

4.6.2. Attacker VM

The Attacker VM simulates an adversary attempting to exfiltrate data using DNS tunneling techniques. The screenshot shows the use of the iodine tool to create a DNS tunnel between the attacker and the target network. Commands like sudo iodined -f -c -P hunter2 10.200.100.1 tunnel.local indicate the setup of a covert communication channel using specially crafted DNS packets. This VM generates malicious DNS traffic that is later detected by the Detector VM, enabling the evaluation of the detection model's accuracy.

4.6.3. Target VM

The Target VM serves as the endpoint receiving DNS queries from the Attacker VM and processing them within the tunneling workflow. It displays raw DNS queries and responses, including headers, questions, and answers, helping validate DNS tunneling propagation and demonstrating how legitimate DNS traffic can carry covert data.

4.6.4. DNS Server VM

The DNS Server VM runs a configured BIND DNS server that resolves incoming DNS queries. The terminal output indicates that the named. The service is active and running, providing authoritative and recursive DNS resolution. It logs incoming DNS requests from both benign clients and the attacker's DNS tunnel. By monitoring the DNS server's behavior, the experiment verifies how DNS tunneling modifies standard DNS traffic patterns and how these deviations are captured by the Detector VM.

Sample Detection Log Output: Table 5 presents sample DNS query records from the experiment, including both benign and tunneling-based malicious traffic. Each record contains the timestamp, traffic class, entropy value, detection time, and queried domain name. Malicious DNS tunneling queries exhibit high entropy values (approximately 4.8–5.0) due to randomized subdomains used to conceal encoded data (e.g., vfs9k2lf9sfd.tunnel.local), whereas benign queries show lower entropy values (approximately 2.1–2.4) and correspond to legitimate domains such as mail.google.com and news.bbc.co.uk. Detection times remain consistently low (1.4–2.4 ms), demonstrating that the baseline SVM classifier trained on entropy feature(s) can efficiently classify DNS traffic in real time and distinguish normal behavior from DNS tunneling attacks in our test environment.

4.6.5. Log analysis of DNS tunnel traffic

The log output shown in Figure 12 represents the real-time detection of DNS tunneling–based activity generated by the NFQUEUE-based monitoring prototype using a baseline classifier trained on handcrafted statistical features (e.g., entropy). Each log entry corresponds to an observed DNS query classified as an attack, indicating suspicious tunneling behavior. This output provides qualitative validation that the deployed baseline model can flag anomalous DNS queries in live traffic, supporting the feasibility of inline detection in the test environment.

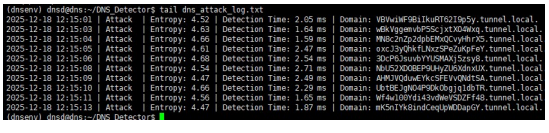

Figure 12: Log of DNS tunnel traffic

4.7. Comparison of theoretical and simulated outputs

The hybrid 1D-CNN + SVM model, combining CNN feature extraction with SVM classification, was expected to exceed 95% accuracy with low false posi-

Table 4: Experimental results on different models

Model	Features	Acc.	Prec.	Rec.	F1
1D-CNN+SVM	Domain Name	99.88	99.84	99.98	99.91
1D-CNN+SVM	Domain Name, Entropy	99.91	99.87	99.99	99.93
1D-CNN+SVM	Entropy	99.76	99.99	99.67	99.83
1D-CNN	Domain Name	99.87	99.82	99.99	99.90
SVM	Domain Name	99.88	99.88	99.94	99.91
SVM	Entropy	99.76	99.99	99.67	99.83
1D-CNN	Entropy	99.76	99.99	99.67	99.83

Table 5: Sample output of real-time DNS tunneling detection (baseline SVM using Entropy Feature)

Timestamp	Class	Entropy	DT (ms)	Queried Domain
2025-08-11 19:34:45	Benign	2.35	1.73	www.example.org
2025-08-11 19:34:48	Benign	2.42	1.67	docs.microsoft.com
2025-08-11 19:34:43	Attack	4.91	1.90	vfs9k2lf9sfd.tunnel.local
2025-08-11 19:34:50	Attack	5.02	1.900	kd9fjsk2d9f.tunnel.local

tives. Simulations surpassed expectations - by achieving 99.91% accuracy, 99.87% precision, and 99.99% recall using domain name and entropy features, with a very low false positive rate (0.38%). The slight gain over the domain name, i.e., only (99.91% vs. 99.88%) confirms that entropy adds complementary discriminative power, especially for low-and-slow tunneling queries.

4.8. Error analysis and sources of discrepancy

Despite near-perfect performance, minor misclassifications occurred. False positives arose from high-entropy benign domains such as dynamic CDN subdomains or tracking URLs, while false negatives occurred in light attacks producing short, low-entropy domains resembling legitimate queries. Detection latency was generally under 3 ms per query, with occasional spikes during bursts due to NFQUEUE buffering and Python's GIL. These errors mainly stem from dataset limitations, feature overlap, and slight training biases.

4.9. Quantitative comparison with state-of-the-art

Table 6 presents a comparative analysis of existing DNS tunneling detection methods using standard performance metrics. The approaches proposed by Lal et al., employing a 1D CNN combined with SVM, and Mahdavifar et al., using a Random Forest classifier, both evaluated on the CIC-Bell-DNSEXF-2021 dataset, demonstrate exceptionally high accuracy, precision, recall, and F1-scores, indicating strong and reliable detection capabilities.

Among these, the Random Forest-based method achieves the best overall performance. In contrast, Bubnov's feedforward neural network, evaluated on a synthetic dataset, exhibits relatively lower performance, emphasizing the influence of dataset realism on detection effectiveness.

5. Conclusion

This paper proposes a hybrid DNS tunneling detection framework combining a one-dimensional Convolutional Neural Network (1D-CNN) for automated feature extraction with a Support Vector Machine (SVM) for classification. By learning character-level patterns from DNS subdomains, the proposed model detects both light and heavy data exfiltration scenarios. Evaluation on the BCCC-CIC-Bell-DNS-EXF dataset shows high effectiveness, achieving up to 99.91% accuracy with high precision and recall and minimal false positives under the adopted evaluation protocol.

In addition, an NFQUEUE-based real-time monitoring prototype demonstrates live DNS packet interception and online feature computation, enabling low-latency (millisecond-level) decision-making in our test environment using a baseline SVM classifier trained on entropy features. This prototype validates the feasibility of inline DNS monitoring and provides a foundation for future integration of the proposed 1D-CNN–SVM model into real-time deployment.

Although the hybrid 1D-CNN–SVM model showed strong offline performance, several directions remain for future research. Extensions could target encrypted DNS

Table 6: Performance of existing DNS tunneling detection methods

Dataset	Acc.	Prec.	Rec.	F1
CIC-Bell-DNS-EXF-2021	99.80	99.72	99.94	99.83
CIC-Bell-DNS-EXF-2021	99.97	99.96	99.98	99.97
Synthetic	97.50	97.10	97.80	97.45

traffic (DoH/DoT) using traffic fingerprinting, TLS analysis, and temporal modeling. Advanced architectures such as Transformers, Bi-LSTMs, or GRUs could better capture sequential dependencies, while incorporating additional contextual features (e.g., query frequency, burstiness, response anomalies, RTT, and session-level DNS patterns) may improve robustness. Finally, large-scale real-world evaluation on high-volume traffic is needed to assess scalability, end-to-end latency, and practical deployment.

To further enhance DNS tunneling detection, future models should move beyond binary classification to identify specific tunneling tools and exfiltration intensity to support more precise incident response. Training on diverse datasets, including obfuscated and emerging variants, can improve robustness against unseen attacks. Detection methods should also address encrypted DNS (DoH/DoT) using flow analysis and behavioral profiling without payload inspection. Incorporating temporal and flow-based features (e.g., query rate, inter-arrival times, and session duration) via sequence models can help capture subtle tunneling behaviors, while adaptive features such as packet-size variability and response anomalies can better distinguish stealthy tunnels from legitimate traffic.

Acknowledgement

The authors declare that they have not received any financial support for this work.

References

[1] Mahdavifar S, Hanafy Salem A, Victor P, et al. Lightweight hybrid detection of data exfiltration using DNS based on machine learning[C/OL]// Proceedings of the 2021 11th International Conference on Communication and Network Security. 2021: 80-86. DOI: 10.1109/ICCR67387.2025.11292120.

[2] Zhang J, Yang L, Yu S, et al. A DNS tunneling detection method based on deep learning models to prevent data exfiltration[C/OL]// Network and System Security: 13th International Conference, NSS 2019. Sapporo, Japan: Springer, 2019: 520-535. DOI: 10.1007/978-3-030-36938-5_32.

[3] Do V T, Engelstad P, Feng B, et al. Detection of DNS tunneling in mobile networks using machine learning[C/OL]// Information Science and Applications 2017: ICISA 2017. Springer, 2017: 221-230. DOI: 10.1007/978-981-10-4154-9_26.

[4] Das A, Shen M Y, Shashanka M, et al. Detection of exfiltration and tunneling over DNS[C/OL]// 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2017: 737-742. DOI: 10.1109/ICMLA.2017.00-71.

[5] Ahmed J, Gharakheili H H, Raza Q, et al. Real-time detection of DNS exfiltration and tunneling from enterprise networks[C/OL]// 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2019: 649-653. DOI: 10.3390/electronics12061467.

[6] Lai C M, Huang B C, Huang S Y, et al. Detection of DNS tunneling by feature-free mechanism[C]// 2018 IEEE Conference on Dependable and Secure Computing (DSC). IEEE, 2018: 1-2.

[7] Liu C, Dai L, Cui W, et al. A byte-level CNN method to detect DNS tunnels[C]// 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). IEEE, 2019: 1-8.

[8] Lal A, Prasad A, Kumar A, et al. DNS-TunNet: A hybrid approach for DNS tunneling detection[C]// 2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC). IEEE, 2022: 1-6.

[9] Bubnov Y. DNS tunneling detection using feedforward neural network[J]. European Journal of Engineering and Technology Research, 2018, 3(11): 16-19.

[10] Sammour M, Hussin B, Othman M F I. Comparative analysis for detecting DNS tunneling using machine learning techniques[J]. International Journal of Applied Engineering Research, 2017, 12(22): 12762-12766.

[11] Abualghanam O, Alazzam H, Elshqeirat B, et al. Real-time detection system for data exfiltration over DNS tunneling using machine learning[J/OL]. Electronics, 2023, 12(6): 1467. DOI: 10.3390/electronics12061467.

[12] Liang J, Wang S, Zhao S, et al. FECC: DNS tunnel detection model based on CNN and clustering[J/OL]. Computers & Security, 2023, 128: 103132. DOI: 10.1016/j.cose.2023.103132.

[13] Zhan M, Li Y, Yu G, et al. Detecting DNS over HTTPS based data exfiltration[J/OL]. Computer Networks, 2022, 209: 108919. DOI: 10.1016/j.comnet.2022.108919.

[14] Pich R, Sreng S, Colin J N. DoH tunneling traffic detection based on single packet features analysis[C/OL]// Proceedings of the 12th International Conference on Networks, Communication and Computing (ICNCC 2023). Osaka, Japan: Association for Computing Machinery (ACM), 2023: 1-7. DOI: 10.1145/3638837.3638861.

[15] Ding S, Zhang D, Ge J, et al. Encrypt DNS traffic: Automated feature learning method for detecting DNS tunnels[C]// Proceedings of the 2021 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Social Computing & Networking, and Sustainable Computing & Communications (ISPA/BDCloud/SocialCom/SustainCom). IEEE, 2021: 352-359.

[16] Işık B, Tuncer B İ, Amirov N, et al. DNS tunneling: Threat landscape and improved detection solutions[Z]. 2025.

[17] Salat L, Davis M, Khan N. DNS tunnelling, exfiltration and detection over cloud environments[J/OL]. Sensors, 2023, 23 (5): 2760. DOI: 10.3390/s23052760.

[18] Sui Z, Shu H, Kang F, et al. A comprehensive review of tunnel detection on multilayer protocols: From traditional to machine

learning approaches[J/OL]. Applied Sciences, 2023, 13(3): 1974. DOI: 10.3390/app13031974.

[19] Adiwal S, Rajendran B, Shetty P D, et al. DNS intrusion detection (DID)—a Snort-based solution to detect DNS amplification and DNS tunneling attacks[J/OL]. Franklin Open, 2023, 2(2): 100010. DOI: 10.1016/j.fraope.2023.100010.

[20] Mitsuhashi R, Jin Y, Iida K, et al. Malicious DNS tunnel tool recognition using persistent DoH traffic analysis[J/OL]. IEEE Transactions on Network and Service Management, 2022, 20 (2): 2086-2095. DOI: 10.1109/TNSM.2022.3215681.

[21] Nguyen T Q, Laborde R, Benzekri A, et al. AutoROC-DBSCAN: Automatic tuning of DBSCAN to detect malicious DNS tunnels[C/OL]// Communications in Computer and Information Science: volume 1641 Emerging Information Security and Applications. Cham: Springer, 2023: 126-144. DOI: 10.1007/978-3-031-23098-1_8.