An ML-based IDS for DNS cache poisoning detection using convolutional neural networks*

Mariia Stadnyk[1,†], Dmytro Tymoshchuk[1,*,†], Mikolaj Karpinski[1,2,†], Yurii Klots[3,†] and Nataliia Petliak[3,†]

[1] Ternopil Ivan Puluj National Technical University, Ruska str. 56, Ternopil, 46001, Ukraine
[2] University of the National Education Commission, 2 Podchorążych str, Krakow, 30084, Poland
[3] Khmelnytskyi National University, 11, Instytuts'ka str., Khmelnytskyi, 29016, Ukraine

Abstract

DNS cache poisoning remains one of the most dangerous attacks against the Domain Name System because forged DNS responses may comply with protocol specifications and evade traditional signature-based intrusion detection techniques. This study proposes an approach for detecting DNS cache poisoning attacks within an ML-driven intrusion detection system based on convolutional neural networks. Unlike conventional methods relying on manually engineered statistical features, the proposed approach transforms IPv4, UDP, and DNS protocol headers into structured 6×32×8 tensor representations, enabling automatic extraction of informative spatial patterns directly from packet headers. Four architectures were evaluated: a Vanilla CNN, a CNN with Batch Normalization and Dropout, a Mini-ResNet, and a CNN incorporating an Efficient Channel Attention (ECA) mechanism. The models were trained and evaluated using a fuzzing-based DNS cache poisoning dataset containing both legitimate and malicious DNS sessions. The CNN with ECA achieved the best performance on the test dataset, reaching an Accuracy of 96.68%, a Recall of 97.62%, and an MCC of 0.9337, demonstrating a substantial reduction in false negatives, which is particularly important for intrusion detection systems. The proposed approach was integrated into a laboratory IDS prototype deployed in a virtualized KVM/QEMU environment using BIND as the recursive DNS server and an IDS sensor responsible for monitoring DNS traffic and classifying DNS sessions. Evaluation in an independent environment confirmed the model's generalization capability, yielding an Accuracy of 92–94%, a Recall of 94–96%, and an MCC of 0.88–0.90. The obtained results demonstrate that convolutional neural networks enhanced with channel attention mechanisms provide an effective and practical solution for DNS cache poisoning detection in real-world network infrastructures.

Keywords

IDS, DNS cache poisoning, artificial intelligence, machine learning, convolutional neural network, cybersecurity, hypervisor, operating systems

1. Introduction

The Domain Name System (DNS) is a distributed hierarchical system that provides the mapping between domain names and IP addresses, as well as access to various types of resource records (MX, TXT, NS, CNAME, etc.). It represents one of the fundamental components of modern network infrastructure. The correct and secure operation of DNS is critically important for ensuring the uninterrupted functioning of web services, email systems, cloud platforms, and corporate information systems [1]. The architectural characteristics of DNS, along with its initial lack of built-in authentication and cryptographic response protection mechanisms, make it an attractive target for attackers. Compromise of DNS response integrity or the substitution of legitimate responses may result in the redirection of users to malicious resources, enable man-in-the-middle attacks, facilitate traffic interception or modification, and ultimately lead to the compromise of sensitive data.

* IIntelITSIS-2026: 7th International Workshop on Intelligent Information Technologies & Systems of Information Security, July 3, 2026, Khmelnytskyi, Ukraine
* Corresponding author.
† These authors contributed equally.
stadnyk_m@tntu.edu.ua (M. Stadnyk); dmytro.tymoshchuk@gmail.com (D. Tymoshchuk); mikolaj.karpinski@uken.krakow.pl (M. Karpinski); klots@khmnu.edu.ua (Y. Klots); npetlyak@khmnu.edu.ua (N. Petliak)
0000-0002-0287-2254 (M. Stadnyk); 0000-0003-0246-2236 (D. Tymoshchuk); 0000-0002-8846-332X (M. Karpinski); 0000-0002-3914-0989 (Y. Klots); 0000-0001-5971-4428 (N. Petliak)

One of the most dangerous attacks targeting DNS is DNS cache poisoning, which belongs to the class of DNS response spoofing attacks. In such an attack, an attacker sends a forged DNS response to a recursive resolver before the legitimate response from the authoritative server is received. If the attack is successful, the malicious record is stored in the resolver's cache according to the specified TTL value. As a result, users receive incorrect IP addresses for the requested domains and may be transparently redirected to malicious resources [2]. A particular challenge in mitigating this threat lies in the fact that forged DNS responses may fully comply with the formal requirements of the protocol and successfully pass transaction ID and source port verification checks in the absence of cryptographic authentication mechanisms.

Traditional approaches to mitigating DNS cache poisoning attacks rely on the deployment of DNSSEC for cryptographic authentication and integrity verification of DNS responses, randomization of transaction identifiers and source ports, the use of DNS cookies, restrictions on caching parameters, and the application of DNS traffic filtering and monitoring mechanisms [3]. Although these measures significantly reduce the likelihood of successful DNS response spoofing, they exhibit several limitations. In particular, DNSSEC is associated with deployment complexity, cryptographic key management overhead, increased infrastructure load, and potential scalability challenges. Furthermore, most conventional protocol-oriented protection mechanisms are designed to prevent attacks at the protocol level and do not provide behavioral or anomaly-based detection of novel or modified attack variants. Similar limitations apply to traditional intrusion detection systems (IDS) operating on signature-based or statically defined rule-based approaches. Such systems are primarily focused on identifying previously known attack patterns and are less effective against new or evolving DNS response spoofing techniques. The adaptive nature of DNS cache poisoning attacks, the variability of their implementation strategies, and the ability to evade static rules by modifying query parameters or packet structures substantially reduce the effectiveness of signature-based detection mechanisms. Moreover, the rapid growth in DNS traffic volumes in modern networks, driven by the expansion of cloud services, content delivery networks (CDNs), and IoT infrastructures, complicates manual security event analysis and necessitates the adoption of automated, intelligent data processing and analysis methods.

Recent advances in artificial intelligence and machine learning have created new opportunities for the development of effective intrusion detection systems [4–7]. Unlike traditional signature-based approaches, machine learning algorithms operate on feature representations of network traffic, enabling the analysis of large data volumes, the identification of hidden statistical patterns, and the detection of anomalous or atypical behavioral characteristics [8–10]. The application of such methods is particularly promising for DNS attack detection, as malicious responses often formally comply with protocol specifications and may be indistinguishable from legitimate packets in terms of structure, differing only in the content of resource records or in the contextual semantics of their use [11].

In this study, we propose an approach for detecting DNS cache poisoning attacks within network intrusion detection systems using artificial intelligence techniques. The novelty of the proposed approach lies not in the application of convolutional neural networks (CNN) themselves, but in the development of a complete ML-based framework for DNS cache poisoning detection. Specifically, the proposed method introduces a structured session-level tensor representation of DNS traffic constructed from carefully selected IPv4, UDP, and DNS header fields while excluding environment-specific information that could reduce model generalization. Furthermore, the study adapts CNN architectures, including an Efficient Channel Attention (ECA) mechanism, to this representation and validates the proposed solution through implementation in a laboratory intrusion detection system and evaluation in an independent virtualized test environment. Unlike previous approaches that rely on statistical feature engineering or dimensionality reduction techniques such as PCA, the proposed method operates directly on a structured session-level tensor representation of raw protocol headers, enabling automatic feature extraction without manual preprocessing.

2. Background and related work

This section analyzes existing approaches to securing the Domain Name System, with particular emphasis on countering DNS spoofing and cache poisoning attacks. Classical protocol- and network-level protection mechanisms are examined, including the deployment of DNSSEC, randomization of transaction identifiers and source ports, the use of DNS cookies, and other techniques aimed at increasing resilience against response forgery. An overview of contemporary machine learning–based solutions is also provided, focusing on methods for DNS traffic feature analysis, anomaly detection, and malicious activity classification. Special attention is given to the evolution of intrusion detection systems and deep learning–driven IDS architectures that leverage behavioral analysis and deep neural networks to improve detection accuracy and robustness against emerging attack variants.

Classical countermeasures against DNS attacks, particularly DNS cache poisoning, include protocol-level and configuration-based approaches aimed at complicating DNS response forgery and mitigating its potential impact. One of the primary techniques is the randomization of DNS transaction parameters, including the random selection of the 16-bit transaction ID and the UDP source port. This increases the entropy of request parameters and reduces the probability of an attacker successfully guessing the correct response in a blind spoofing scenario [3]. Additional techniques are also employed, such as 0x20-bit encoding [12], which relies on case sensitivity in domain names to introduce an extra source of entropy, and the DNS Cookies mechanism, which implements a limited form of authentication between the resolver and the server, thereby complicating attacks by off-path adversaries. To limit the duration of negative impact in the event of a successful poisoning attempt, control over record Time-To-Live (TTL) values is applied, restricting the period during which cached data remains valid [13]. Furthermore, DNS traffic filtering mechanisms are deployed at the network infrastructure level, including DNS firewalls and Response Policy Zones (RPZ), designed to block suspicious or malicious DNS responses [14].

The most comprehensive protocol-level mechanism for ensuring the authenticity and integrity of DNS data is DNSSEC, which employs cryptographic signatures to validate DNS resource record sets (RRsets) and establish a chain of trust between the root zone, intermediate domains, and authoritative resource records. When correctly deployed, DNSSEC significantly reduces the risk of DNS cache poisoning and response spoofing attacks. Nevertheless, recent studies have identified several practical limitations that hinder its widespread adoption, including the complexity of cryptographic key management, increased computational overhead associated with signature verification, larger DNS message sizes, and the potential amplification of denial-of-service attacks. Furthermore, many real-world deployments still lack complete DNSSEC coverage, and resolver-side validation is not consistently enabled. Consequently, DNSSEC is not universally adopted, particularly in heterogeneous enterprise and hybrid network environments.

The combination of classical countermeasures against DNS attacks enhances the resilience of DNS infrastructures and reduces the probability of successful response spoofing. However, these mechanisms do not guarantee complete protection against all possible attack variants, particularly in cases of misconfiguration or the emergence of new techniques designed to bypass protocol-level safeguards. This underscores the need for complementary threat detection mechanisms alongside preventive protection measures. In this context, recent years have seen growing attention toward machine learning–based approaches that analyze statistical and behavioral features of DNS traffic. Such methods enable the detection of anomalous or malicious activity based on data-driven patterns rather than relying solely on predefined rules.

The authors of [15] propose a hybrid two-stage machine learning approach for detecting DNS-related vulnerabilities, with a particular focus on DNS over HTTPS (DoH), which encrypts DNS queries and complicates their inspection by conventional security mechanisms. At the first stage, network traffic is analyzed using the Random Forest algorithm and classified into DoH and non-DoH traffic. At the second stage, the selected DoH traffic is further processed using AdaBoost Trees to distinguish between legitimate and malicious DoH traffic. To reduce computational complexity,

the authors applied Principal Component Analysis (PCA) for feature reduction, ultimately selecting only six out of thirty-three available features for malicious traffic identification. Additionally, Random Under-Sampling was employed to address class imbalance and reduce the dataset size. The reported experimental results indicate high effectiveness of the proposed approach, achieving classification accuracy of up to 99.4% at the first stage and 100% at the second stage. These findings demonstrate the advantage of the proposed method over existing approaches, particularly in light of the substantial reduction in the number of utilized features and training samples.

The authors of [16] present a critical analysis and refinement of the approach proposed in [17], focusing on improving the detection performance of malicious DoH traffic using the CIRA-CIC_DoHBrw-2020 dataset. In particular, they address excessive noise in the data, introduce feature selection techniques, and enhance the interpretability of dataset characteristics, thereby improving both detection accuracy and efficiency. The proposed modifications include the extraction of informative features using the chi-square (χ^2) test and the Pearson Correlation Coefficient (PCC). Based on the experimental results, the authors recommend the use of the Light Gradient Boosting Machine (LGBM) model for distinguishing between DoH and non-DoH traffic, as well as between legitimate and malicious DoH traffic. LGBM demonstrated higher classification accuracy and shorter training time compared to algorithms such as Random Forest, Decision Tree, and XGBoost.

Using publicly available datasets, the authors of [18] perform the classification of both legitimate and malicious DoH traffic employing several machine learning algorithms, including Random Forest (RF), Gradient Boosting (GB), k-Nearest Neighbors (KNN), Logistic Regression (LR), and Naïve Bayes (NB). The experimental evaluation demonstrates that ensemble methods such as RF and GB achieve the highest classification accuracy and stability. At the same time, KNN and LR also deliver strong performance in detecting malicious DoH traffic, whereas the NB model exhibits comparatively lower effectiveness than the other approaches.

The study in [19] on DNS cache poisoning detection demonstrates that the high-dimensional and multi-packet nature of DNS traffic significantly complicates the application of traditional machine learning methods without prior data preprocessing. To reduce the dimensionality of the input space, PCA was employed to select the most informative features, which on average explained more than 97% of the variance in the original data, thereby enabling efficient training of classical ML models. At the same time, a convolutional neural network (CNN)-based approach achieved superior performance compared to k-Nearest Neighbors, Support Vector Machines, Decision Trees, and Random Forest models in the task of detecting DNS cache poisoning attacks.

A semi-supervised Support Vector Machine approach was proposed in [20] for detecting DoH tunneling. The authors address the challenge of identifying DoH traffic, as HTTPS encryption prevents traditional packet content inspection. The study utilizes statistical features of network flows (34 features) extracted from DoH sessions, followed by the application of a semi-supervised learning algorithm to distinguish between legitimate and malicious tunneled traffic. This approach reduces dependence on fully labeled datasets and enhances the system's ability to detect novel or previously unseen types of DoH tunneling.

The author of [21] focuses on the application of the Fuzzy C-Means (FCM) method, which enables network samples to be assigned to multiple clusters with varying degrees of membership. This property is particularly valuable in scenarios where the boundary between legitimate and malicious traffic is not clearly defined. The study analyzes statistical characteristics of DNS traffic, which are used to construct the feature space for clustering. The proposed approach is designed for anomaly detection without strict reliance on fully labeled data, thereby increasing the flexibility of the detection system in identifying new or variable types of attacks targeting DNS infrastructure.

In [22], the authors propose an ensemble intrusion detection method for Ethernet networks in modern railway transportation systems, combining CNN and recurrent neural networks (RNN). To leverage advantages of both architectures, multiple base CNN and RNN classifiers were constructed and integrated using a dynamic weighted voting strategy. The data were transformed into suitable

formats for each architecture: a “data imaging” technique was applied to generate input representations for CNN models, while temporal sequences were built for RNN-based processing.

Ensemble approaches are also employed in [23], where a deep learning architecture combining convolutional and recurrent neural networks is proposed for network attack detection. The authors focus on automatic extraction of informative features from network traffic without manual feature engineering, enabling effective processing of high-dimensional and temporally dependent data. Specifically, convolutional layers are used to capture local spatial patterns within packets, while recurrent components model sequential dependencies between events in network sessions.

An analysis of recent studies indicates that DNS attack detection encompasses various threat categories, including DNS cache poisoning and tunneling via DNS or DoH. The field has evolved progressively (Table 1) from the application of classical machine learning algorithms to ensemble and hybrid deep learning approaches.

Table 1
Evolution of approaches to DNS attack detection

Article No.	Methods	Dataset	Feature Engineering	Detection target
[15]	RF + AdaBoost	CIRA-CIC_DoHBrw-2020	PCA (6/33 features), Random Under-Sampling	Malicious DoH
[16]	LGBM, RF, DT, XGBoost	CIRA-CIC_DoHBrw-2020	χ^2-test, Pearson Correlation Coefficient (PCC)	Malicious DoH
[18]	RF, GB, KNN, LR, NB	Public DoH datasets	No complex feature engineering	Malicious DoH
[19]	kNN, SVM, DT, RF, CNN	Own testbed	PCA (>97% explained variance)	DNS cache poisoning
[20]	Semi-supervised SVM	Flow-based DoH dataset	34 statistical features	DoH tunneling
[21]	Fuzzy C-Means	DNS traffic dataset	Statistical features	DNS anomalies
[22]	CNN + RNN	ECN testbed	Data imaging + temporal sequences	Network intrusion
[23]	CNN + RNN	IDS dataset	Automatic feature extraction	Network intrusion
This work	CNN+ECA	Own testbed	(6×32×8) constructed from selected IPv4, UDP, and DNS header fields	DNS cache poisoning

Early studies predominantly relied on traditional classifiers such as kNN, SVM, DT, and RF. Over time, the focus shifted toward ensemble models, including RF, AdaBoost, LightGBM, and

XGBoost. More recent research increasingly adopts deep and hybrid architectures, such as CNN, RNN, and combined CNN/RNN models. At the same time, most approaches face the challenge of high-dimensional feature spaces, necessitating the use of feature selection or dimensionality reduction techniques (e.g., PCA, χ^2 test, Pearson correlation coefficient), as well as data balancing methods in the presence of class imbalance. This trend highlights the inherent complexity of DNS traffic analysis and the need for models capable of automatically learning informative data representations. Consequently, the adoption of CNN-oriented architectures within modern ML- and DL-based intrusion detection systems is well justified for robust detection of various types of DNS attacks in real-world network environments.

Unlike the reviewed studies, the proposed approach focuses on a structured session-level tensor representation of DNS traffic, its adaptation to CNN-based architectures with channel attention, and validation within an independent IDS testbed.

3. Data and methods

3.1. Dataset description

In this study, a dataset generated by the authors in [24] using fuzzing techniques was employed, as only a limited number of publicly available datasets specifically address DNS cache poisoning attacks. Fuzzing is a testing method that involves supplying a system with random or intentionally modified input data to identify errors and vulnerabilities. In this case, a black-box approach was adopted, which does not require access to the source code of the target system [25]. Unlike the use of existing fuzzing tools, the authors in [24] developed custom fuzzing scripts. To construct the dataset, a DNS cache poisoning attack scenario was simulated, involving the generation of forged DNS responses and the variation of ten DNS message fields, including resource record parameters such as TTL values. This enabled the creation of a controlled dataset containing both legitimate and malicious transactions for subsequent model training. The data in [24] were obtained through controlled simulation of legitimate and malicious client interactions with a local DNS server. In the malicious scenario, the cache poisoning mechanism was reproduced by injecting forged DNS responses during the DNS resolution process. In contrast, the legitimate scenario consisted of standard DNS queries to external servers. To ensure experimental reproducibility, the DNS server cache was regularly cleared, and the generated network traffic was captured and stored for further processing and session-level labeling.

3.2. Data pre-processing

Within the development of the ML-based intrusion detection system, convolutional neural networks (CNNs) were selected as the core modeling approach, which required an appropriate data representation. To this end, each aggregated DNS traffic session was transformed into a matrix format, where rows correspond to individual network packets within the session and columns represent the byte-level encoding of selected packet fields. Such a representation enables the application of convolutional operations for the automatic extraction of local structural dependencies within the byte-level representation of network headers, facilitating the formation of informative features without explicit manual feature engineering.

Since each attack episode is formed as a set of related DNS transactions comprising a sequence of request and response packets, and the label is assigned at the level of the entire aggregated session, the construction of a single data sample incorporated information from multiple interrelated packets. At the same time, only informative fields from each packet were selected. To prevent the model from learning environment-specific characteristics, all data associated with MAC addresses were removed, and Ethernet link-layer bytes were entirely excluded. The final feature set included only fields from the network, transport, and application layers (IP, UDP, and DNS) that are potentially relevant for detecting DNS cache poisoning attacks. The list of these fields is presented in Table 2.

Table 2
Input features extracted from IPv4, UDP, and DNS headers

Field	Header	Size (bytes)	Explanation
Version	IPv4	0.5 (4 bits)	Indicates the IP protocol version (typically IPv4).
Internet Header Length (IHL)	IPv4	0.5 (4 bits)	Specifies the length of the IPv4 header.
DSCP / ECN	IPv4	1	Quality of Service and congestion notification information.
Total Length	IPv4	2	Total length of the IP packet (header + data).
Identification (IP ID)	IPv4	2	Unique identifier used for packet fragmentation.
Flags	IPv4	3/8 (3 bits)	Controls fragmentation behavior (DF, MF flags).
Fragment Offset	IPv4	13/8 (13 bits)	Indicates the fragment position within the original packet.
Time To Live (TTL)	IPv4	1	Limits packet lifetime in the network.
Protocol	IPv4	1	Identifies the encapsulated upper-layer protocol (e.g., UDP).
Header Checksum	IPv4	2	Error-checking value for the IPv4 header.
Source Port	UDP	2	Source application port number.
Destination Port	UDP	2	Destination application port number.
Length	UDP	2	Length of the UDP header and data.
Checksum	UDP	2	Error-checking value for the UDP segment.
Transaction ID	DNS	2	Identifier used to match DNS queries and responses.
Flags	DNS	2	Indicates query/response type and control information.
Question Count (QDCOUNT)	DNS	2	Number of questions in the DNS message.
Answer Count (ANCOUNT)	DNS	2	Number of answer records.

Field	Header	Size (bytes)	Explanation
Authority Count (NSCOUNT)	DNS	2	Number of authority records.
Additional Count (ARCOUNT)	DNS	2	Number of additional records.
Total extracted header bytes	—	**32 bytes**	Raw header bytes extracted per packet and converted to 32×8 bit-level representation.

Although the Version and IHL fields jointly occupy a single byte in the IPv4 header, and the Flags and Fragment Offset fields share two bytes, they are presented in Table 2 as separate logical protocol subfields for the sake of improved technical clarity.

The formation of a single data sample is carried out through logical aggregation of captured DNS traffic within a single client–local DNS server interaction, where a session is initiated by a client DNS query. Each session consists of a sequence of related packets that together constitute a complete request–response exchange. From each packet, 32 fixed informative bytes are extracted, forming a vector of integer values in the range 0–255. To standardize session length, a sliding window of six packets is applied, resulting in each subsequence being represented as a 6×32 matrix. At the final stage, each byte is converted into its 8-bit representation, producing a tensor of size 6×32×8, which serves as input to the convolutional neural network (Figure 1).

Following this, a dataset balancing stage was performed to ensure an appropriate class distribution. The resulting statistics are presented in Table 3.

One limitation of the proposed approach lies in the use of certain network header fields (notably IP Identification, TTL, Fragment Offset, and UDP checksum), which may reflect characteristics of the operating system, routing algorithms, or the topology of a specific network. This introduces a potential risk that the model may learn dependencies related to infrastructure-specific properties rather than intrinsic features of the attack itself. To ensure the generalizability of the results, the model was therefore evaluated in a separate test environment.

Although an explicit ablation study was beyond the scope of the present work, evaluation in an independent virtualized environment provides indirect evidence that the proposed model does not rely exclusively on infrastructure-specific characteristics. If the classifier had primarily learned environment-dependent properties rather than attack-related patterns, a substantially larger degradation in detection performance would be expected under different deployment conditions. Instead, the model preserved high Recall and MCC values, indicating that the learned representation captures behavioral characteristics of DNS cache poisoning that generalize across environments.

All models were implemented in Python 3.12 using the TensorFlow 2.18 framework. Model training was performed with the Adam optimizer using a learning rate of 0.001 and the Cross-Entropy loss function. The batch size was set to 64, and each model was trained for a maximum of 100 epochs. To reduce the risk of overfitting, an early stopping strategy with a patience of 10 epochs was applied. For reproducibility, a fixed random seed of 42 was used throughout the experiments. The dataset was randomly divided into training and testing subsets using an 80:20 ratio while preserving the original class distribution. For binary classification, a decision threshold of 0.5 was employed to assign samples to the Attack and Normal classes. Model training was performed on a workstation equipped with an NVIDIA GeForce RTX 3060 GPU, an Intel Core i7 processor, and 16 GB of RAM, running Microsoft Windows 11.

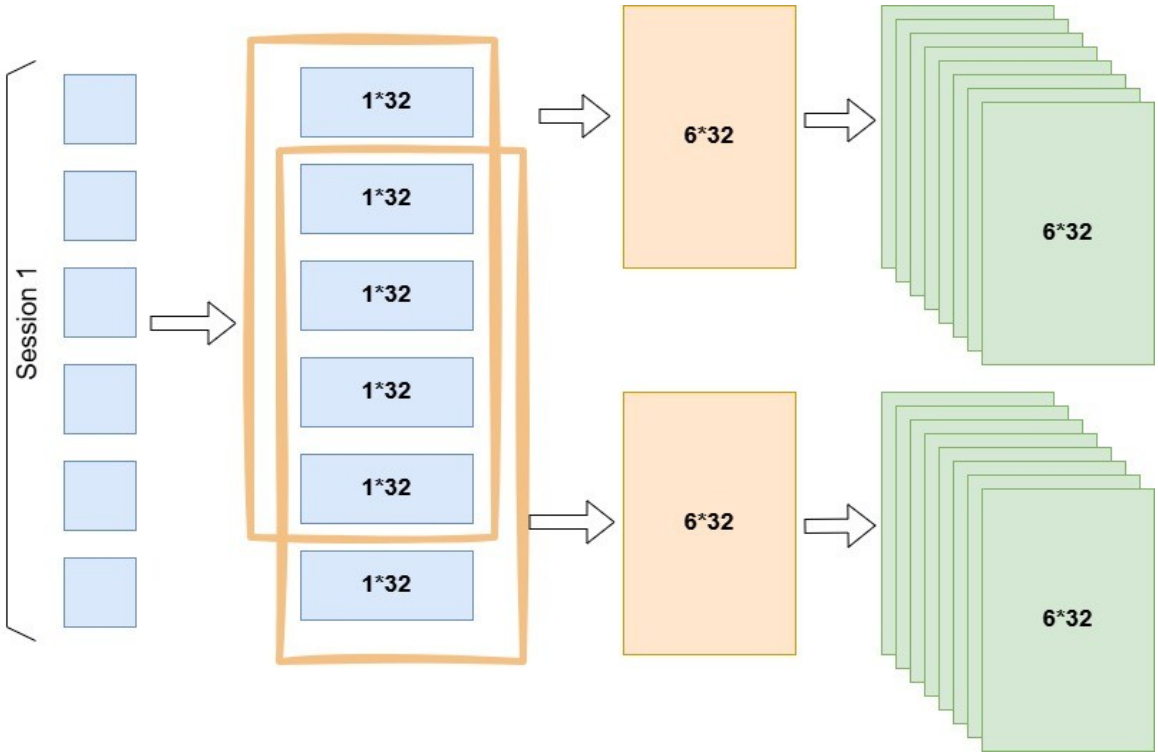


Figure 1: Schematic representation of transforming data into an image-like format

Table 3
Dataset statistics

Data set	Volume, records	Ratio of malicious to benign
Train	30925	1,004:1
Test	7732	0,9800:1

3.3. CNN models

Convolutional neural networks have demonstrated high effectiveness in tasks involving automatic extraction of local features due to the use of convolutional filters, weight sharing mechanisms, and hierarchical feature representation learning. Although CNNs were originally developed for computer vision applications, their ability to model spatial dependencies makes these architectures well suited for analyzing structured tensor representations of network traffic.

In the present study, the input data are represented as a compact three-dimensional tensor of size 6×32×8, capturing the temporal sequence of packets, the byte-level structure of headers, and their corresponding bit planes. This representation naturally aligns with the local receptive field mechanism of CNNs, enabling efficient detection of characteristic patterns within protocol headers.

To systematically analyze the impact of architectural enhancements, four models were selected, reflecting contemporary directions in the development of convolutional networks for traffic analysis tasks:

- Vanilla CNN, serving as a baseline convolutional architecture to establish a reference performance level.

- CNN with Batch Normalization and Dropout, enabling the assessment of the impact of training stabilization and regularization on the generalization capability of models in IDS tasks.
- Mini-ResNet, implementing the residual learning principle to improve optimization of deeper models and enhance detection accuracy.
- CNN with an Efficient Channel Attention (ECA) mechanism, which provides adaptive channel-wise feature weighting and increases model selectivity without significantly increasing computational complexity.

The selected set of models allows for a systematic investigation of the influence of key architectural components under identical training conditions and a unified input data format. This approach ensures a consistent and fair comparison of the contribution of each architectural enhancement to improved detection performance.

3.3.1. Vanilla CNN

A typical convolutional network consists of a sequence of convolutional layers (Conv layers) that perform filtering operations on the input tensor, subsampling layers for reducing the spatial dimensionality of the representation, and one or more fully connected layers for final classification. Convolutional layers are responsible for the automatic extraction of local patterns, whereas pooling layers reduce dimensionality and increase robustness to minor spatial variations.

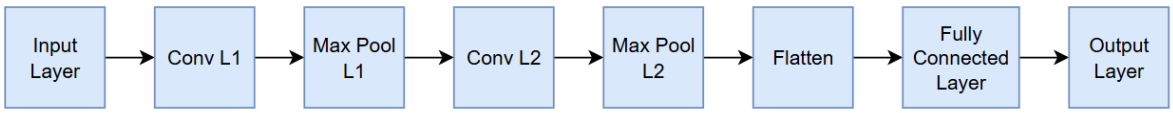


Figure 2: Vanilla CNN for DNS cache poisoning detection

The Vanilla CNN (Figure 2) represents a simplified convolutional architecture consisting of convolution, pooling, and fully connected layers. It does not incorporate complex architectural mechanisms and consists of two or more convolutional layers with a nonlinear ReLU activation function, followed by subsampling operations (most commonly MaxPooling). The extracted feature maps are then flattened into a vector and passed to a fully connected layer and the final output classifier. The ReLU activation function is employed due to its computational simplicity and its ability to mitigate the vanishing gradient problem compared to sigmoid-based activation functions.

The advantages of the Vanilla CNN include its implementation simplicity, a relatively small number of parameters, and stable training behavior at moderate depth. However, it also exhibits certain limitations, including susceptibility to overfitting in the absence of additional regularization, the lack of adaptive feature weighting mechanisms, and limited capacity to model global dependencies.

In this study, the Vanilla CNN serves as the baseline model, providing a reference performance level against which the effectiveness of the proposed architectural enhancements is evaluated.

3.3.2. CNN with batch normalization and dropout

To improve training stability and reduce the risk of overfitting, Batch Normalization and Dropout mechanisms were integrated into the convolutional architecture, thereby extending the previous baseline model.

Batch Normalization is a technique for normalizing internal representations by standardizing the activations of each layer during training. The normalization of intermediate feature maps reduces internal covariate shift, ensures more stable gradient propagation, enables the use of higher learning rates, and accelerates model convergence [26]. Dropout is a regularization

technique that randomly deactivates a subset of neurons during training. This approach reduces overfitting by preventing neuron co-adaptation and improving the model's generalization capability.

In this architecture, Batch Normalization layers are placed after convolutional operations to stabilize activation distributions, while Dropout is applied both within convolutional blocks and in the fully connected layer. This modification enhances the model's generalization capability without significantly increasing architectural complexity.

3.3.3. Mini-ResNet

As the number of layers increases, deep neural networks may become more difficult to optimize, leading to degradation in training performance despite their greater representational capacity. To overcome this limitation, residual learning was introduced through the ResNet architecture [27].

The core idea of residual learning is the use of skip connections that add the input of a residual block directly to its output. Rather than learning the underlying mapping $H(x)$, the network learns the residual function $F(x)=H(x)-x$, thereby facilitating gradient propagation during backpropagation and enabling the training of deeper and more stable models. In this study, the proposed Mini-ResNet represents a compact adaptation of this architecture for analyzing structured tensors of size $6\times32\times8$. The model consists of an initial convolutional layer followed by two residual blocks, each comprising two convolutional layers with nonlinear activation and normalization together with identity or projection-based skip connections.

The Mini-ResNet architecture is employed to investigate whether residual learning improves the ability to capture complex dependencies within packet header structures.

3.3.4. CNN with the ECA mechanism

The CNN model incorporating the ECA mechanism is employed to test the hypothesis that adaptive channel-wise feature weighting enhances the identification of informative patterns within packet header structures represented as $6\times32\times8$ tensors. Comparing this model with the baseline architectures enables an assessment of the impact of channel attention mechanisms on attack detection performance and determines the feasibility of their application in network traffic analysis tasks.

ECA is an enhanced channel attention mechanism designed to reduce computational complexity and the number of additional parameters [28]. Unlike classical attention mechanisms that rely on fully connected layers for dimensionality reduction and restoration, ECA employs a local 1D convolution to model inter-channel dependencies without performing dimensionality reduction. This enables efficient capture of cross-channel relationships while maintaining architectural simplicity.

In the proposed model, the ECA mechanism is integrated into the convolutional blocks following the convolution operations. First, global average pooling is applied to obtain a compact channel-wise descriptor. Subsequently, local inter-channel interaction is modeled using a 1D convolution. The resulting coefficients are then used to rescale the corresponding feature channels.

3.3.5. Evaluation metrics

The evaluation of models intended for deployment in intrusion detection systems (IDS) is conducted in accordance with the methodology proposed in [29, 30 ,31], where the effectiveness of intrusion detection techniques is assessed using metrics derived from the confusion matrix, including its row-normalized variant (normalized by true classes) and related classification performance indicators.

In the confusion matrix:

- True Positive (TP) – an attack correctly classified as an attack.
- False Negative (FN) – a missed attack, which may lead to compromise of the network infrastructure.
- False Positive (FP) – a false alarm, generating unnecessary load on the system or security operators.
- True Negative (TN) – normal traffic correctly classified as normal.

Accuracy (A) is one of the most commonly used metrics for evaluating classification performance. It represents the ratio of correctly predicted instances to the total number of analyzed samples. Precision (P) reflects the proportion of correctly identified attacks among all instances classified by the model as attacks. Recall (R) characterizes the model's ability to detect malicious activity and is defined as the proportion of correctly identified attacks among all actual attack instances in the dataset (Table 4).

Table 4
Evaluation Metric Calculation Formulas

Formula	
$A = \frac{TP + TN}{TP + TN + FP + FN}$	(1)
$P = \frac{TP}{TP + FP}$	(2)
$R = \frac{TP}{TP + FN}$	(3)
$F_\beta = (1 + \beta^2) \times \frac{P \times R}{\beta^2 * P + R}$	(4)
$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$	(5)

Since, in the context of DNS cache poisoning detection, missing an attack (False Negative) is more critical than generating a false alarm (False Positive), the evaluation primarily focuses on the "Attack" class. For an integral assessment, the F_β-score is employed, assigning greater weight to Recall while preserving the contribution of Precision. The value of $\beta = 1.2$ was selected to ensure increased model sensitivity to attacks, as failure to detect malicious activity may result in compromise of the network infrastructure. The MCC can be interpreted as a correlation coefficient between the true class labels and the model's predictions. This metric measures the degree of agreement between actual and predicted values while accounting for all four components of the confusion matrix (TP, TN, FP, FN). MCC integrates the effects of both types of errors into a single numerical measure and prevents the model from achieving a high score based solely on one strong metric. Therefore, the use of MCC provides a more objective and comprehensive evaluation of classification performance in the IDS context.

4. Results and prototype implementation

4.1. Evaluation results on CNN models

To determine the optimal configuration of the baseline Vanilla CNN model, an experimental hyperparameter tuning process was conducted by evaluating different combinations of convolutional filter counts, kernel sizes, and Dropout regularization rates. Within the study, the following convolutional layer configurations were explored in terms of filter counts: {64, 64, 16}, {64, 32, 16}, {32, 32, 16}, {32, 32, 8}, {32, 16, 8}, and {16, 16, 8}. For each configuration, convolutional kernel sizes of (2×2), (3×3), and (4×4) were tested, along with Dropout rates ranging from 0.05 to 0.25. Based on the experimental analysis, the best performance metrics were achieved by the Vanilla CNN architecture featuring a gradual reduction in the number of filters (Conv L1 = 64, Conv L2 = 32, Flatten = 16), convolutional kernels of size (3×3), and a Dropout rate of 0.2. This configuration provided an optimal balance between the model's representational capacity and its generalization ability.

For the second model, in which Batch Normalization layers are placed after the convolutional layers and Dropout is applied both within convolutional blocks and in the fully connected layer, additional hyperparameters were determined empirically through a systematic exploration of possible configurations. The tuning process involved experimentally analyzing the impact of different parameter values on model performance metrics. As a result, the optimal value of the Batch Normalization momentum parameter was found to be 0.85, providing sufficient adaptability of layer statistics while maintaining training stability. For the Dropout mechanism, a differentiated strategy was adopted: a rate of 0.05 was applied within convolutional blocks to minimize the loss of spatial information, whereas a stronger regularization rate of 0.15 was used in the fully connected layer to reduce the risk of overfitting during the final classification stage.

For the Mini-ResNet model, hyperparameter optimization was also performed to determine the optimal architectural configuration for the input tensor of size 6×32×8. Parameter selection was conducted empirically through systematic exploration of combinations involving the number of filters in the residual blocks, convolutional kernel sizes, normalization parameters, and learning rates. Based on the experimental analysis, the best performance was achieved with a configuration comprising an initial convolutional layer with 64 filters, followed by two residual blocks with 64 and 32 channels, respectively, using 3×3 convolutional kernels and a learning rate of 10^{-3}. This configuration provided an optimal balance between model depth and training stability.

For the CNN with ECA architecture, a baseline two-block CNN structure is employed, augmented with an ECA module inserted after each convolutional block. Each convolutional block consists of a 3×3 convolution operation, followed by Batch Normalization and ReLU activation. The ECA module is then applied to perform adaptive channel-wise feature weighting prior to the MaxPooling 2×2 operation. After two such blocks, Global Average Pooling is performed, followed by a fully connected layer and the final output classifier. The optimal configuration includes the following parameters: 64 filters in the first convolutional layer and 32 filters in the second layer, with 3×3 kernels used in both layers. Batch Normalization employs a momentum coefficient of 0.85. For the ECA module, the 1D convolution kernel size along the channel dimension is set to $k = 3$, representing a compromise between expressive capacity and parameter efficiency. Dropout regularization is set to 0.05 within the convolutional blocks and 0.15 in the fully connected layer. The fully connected layer consists of 128 neurons with ReLU activation.

In this study, the "Attack" class is defined as the positive class, while "Normal" (legitimate network traffic) is treated as the negative class. Based on this designation, the quantitative values of binary classification performance metrics were calculated for each of the evaluated models in accordance with the previously defined measures (Table 5). This setup ensures a correct interpretation of the results from the perspective of attack detection effectiveness.

Table 5
Performance indicators

Model	TP	FP	FN	TN	A	P	R	F_β	MCC
Vanilla CNN	3320	486	506	3420	0.8717	0.8723	0.8677	0.8696	0.7434
CNN+ BN-Dropout	3600	326	226	3580	0.9286	0.9170	0.9409	0.9310	0.8575
Mini ResNet	3680	216	146	3690	0.9532	0.9446	0.9618	0.9547	0.9065
CNN +ECA	3735	166	91	3740	0.9668	0.9574	0.9762	0.9684	0.9337

The Accuracy and Precision metrics exhibit a positive trend when transitioning from the Vanilla CNN to Mini-ResNet and CNN+ECA architectures. However, Recall and the F_β-score are of primary importance in the IDS context. Figure 3 presents the confusion matrices illustrating the classification results obtained using the four evaluated models.

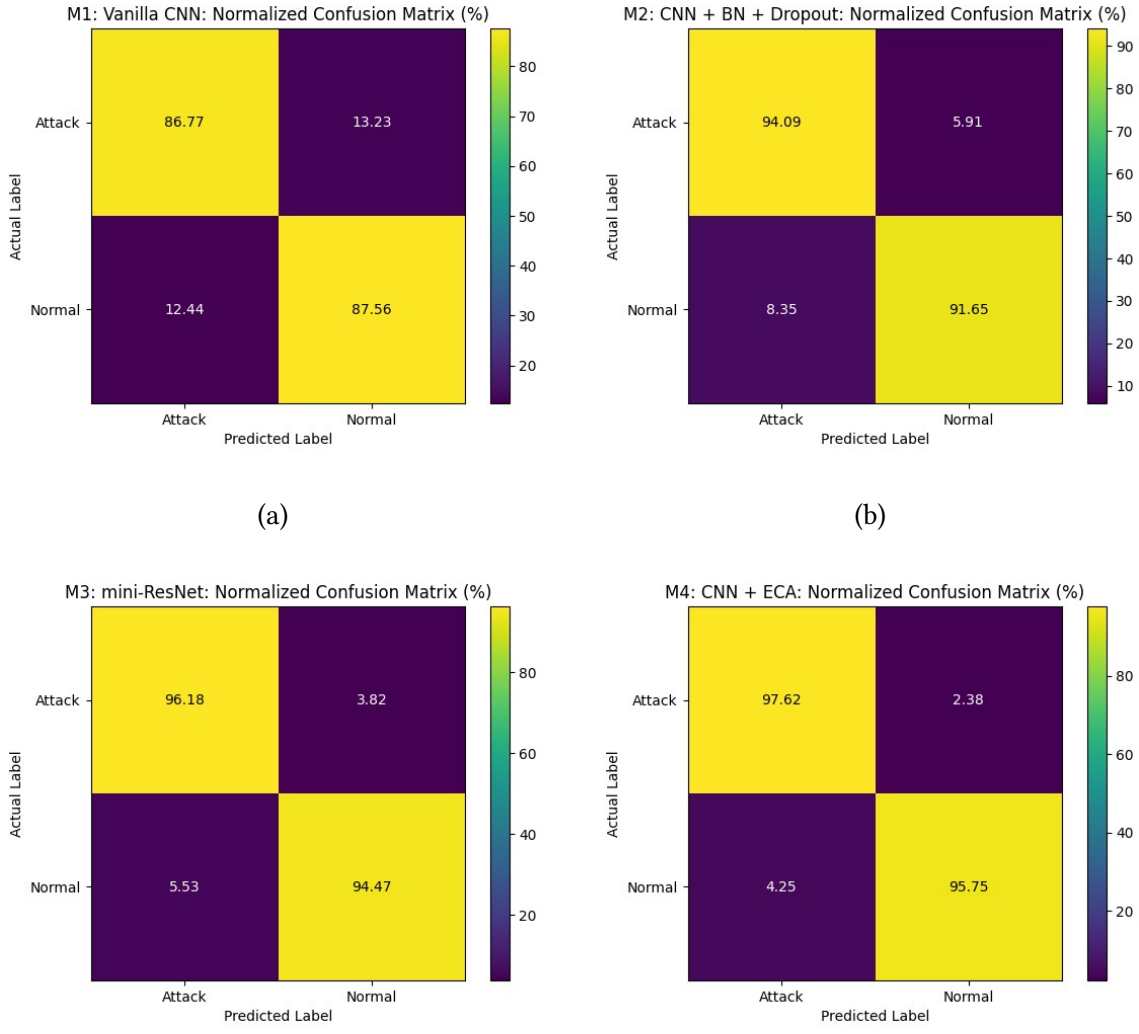
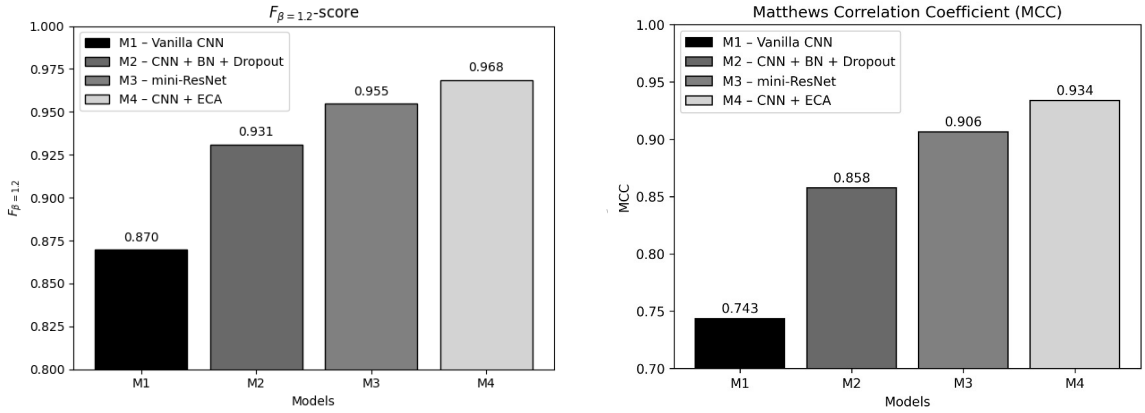


Figure 3: Confusion matrices for the test dataset obtained using Vanilla CNN (a), CNN+ BN-Dropout (b), Mini ResNet (c), CNN +ECA (d).

The confusion matrices reveal a gradual reduction in both false negatives (FN) and false positives (FP) when transitioning from the baseline model M1 (Vanilla CNN) to more advanced architectures. Model M4 (CNN + ECA) achieved the lowest number of missed attacks, which directly translated into the highest Recall value. Given that minimizing false negatives is critically important in IDS applications, this result indicates the superior practical suitability of the model incorporating the channel attention mechanism.

Figure 4(a) presents a comparative analysis of the evaluated models based on the F_β-score with β = 1.2. The choice of β = 1.2 reflects the need to assign greater weight to Recall in the IDS context, where missed attacks (FN) are more critical than false alarms. As shown in the diagram, the highest F_β-score is achieved by model M4 (CNN + ECA), indicating its superior ability to detect attacks while maintaining high precision. Model M3 (Mini-ResNet) demonstrates closely comparable performance, whereas M2 (CNN + Batch Normalization + Dropout) provides a noticeable improvement over the baseline M1 (Vanilla CNN).

Figure 4(b) illustrates the comparison of models in terms of the Matthews Correlation Coefficient (MCC). The obtained values confirm the previous findings: CNN + ECA (M4) exhibits the highest agreement between predicted and true class labels, with Mini-ResNet (M3) ranking second. The baseline Vanilla CNN (M1) yields the lowest MCC values, further substantiating the effectiveness of normalization, regularization, and attention mechanisms in enhancing attack detection performance.



(a) (b)

Figure 4: F_β-score comparison (a), MCC comparison (b).

Thus, the results presented in Figure 4 consistently demonstrate the superiority of architectures incorporating residual connections and channel attention mechanisms over the baseline convolutional model.

4.2. ML-based IDS prototype and experimental testbed

The experimental environment for evaluating the IDS with an ML-based DNS cache poisoning detection module was deployed on the KVM/QEMU hypervisor. All components operated as separate virtual machines running Ubuntu Linux 22.04 LTS, ensuring network stack stability and enabling automated experiment execution using shell and Python scripts. Virtual machine lifecycle management was performed via libvirt, while network communication was implemented using Linux Bridge. This configuration established an isolated segment for DNS traffic exchange and enabled traffic mirroring to the monitoring node (Figure 5).

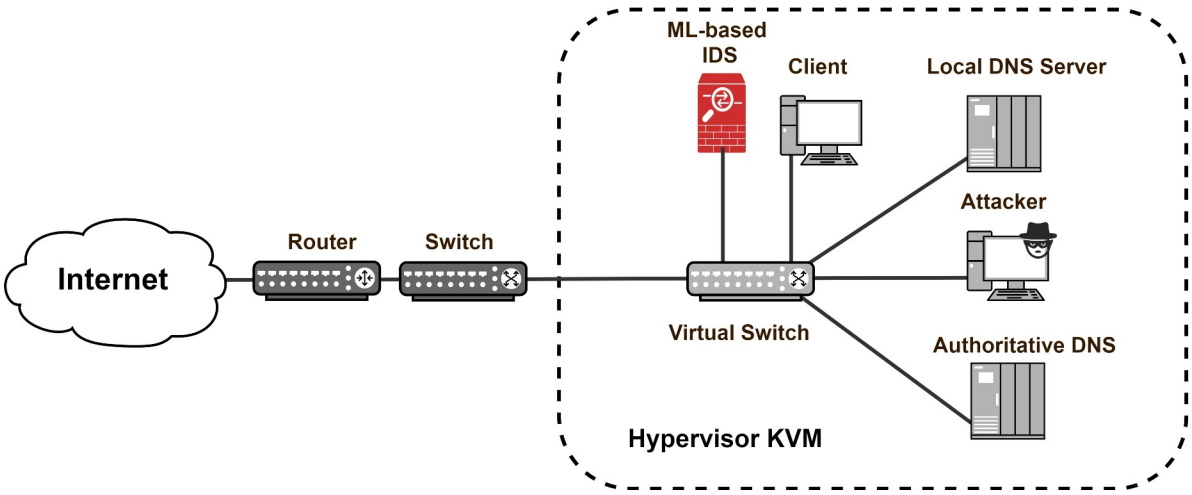


Figure 5: Laboratory testing environment of the ML-based IDS.

The test environment included a local recursive DNS server with caching enabled, a client machine, a node for generating anomalous traffic, and a dedicated IDS machine equipped with a machine learning module. The local DNS server was implemented using BIND configured as a recursive caching resolver with DNS query and response logging enabled. The configuration provided detailed logging to capture relevant events. To ensure experimental reproducibility, the DNS server cache was cleared before each test series, restoring the server to a controlled initial state. When necessary, a separate authoritative test zone server, also based on BIND, was deployed to serve specially prepared domains with controlled resource records.

The client machine periodically generated DNS queries to the local DNS server using the standard dig utility. The query frequency was programmatically controlled, enabling the generation of both single and periodic requests to selected domains. The query results were logged together with timestamps, allowing correlation of responses with events recorded on the server and within the IDS. In parallel, network traffic was captured in PCAP format using tcpdump, providing the possibility for subsequent offline analysis in Wireshark.

In the DNS cache poisoning scenario, the client machine issued a DNS query for a domain whose record was not present in the cache of the local DNS server, thereby initiating the recursive resolution process. The anomalous traffic generation node, connected to the same virtual network segment, operated in monitoring mode to observe DNS exchanges and, during designated test series, generated modified DNS responses using a fuzzing-based approach. Response generation was implemented in Python using the scapy and dnspython libraries, enabling programmatic construction of DNS messages with controlled variation of specific header and resource record fields, including TTL values and control flags. Within the laboratory setup, these forged responses were injected into the network segment as alternatives to the legitimate response. If a simulated response was accepted by the local DNS server, the corresponding record was cached for the duration specified by the TTL, after which the client machine received an incorrect IP address in response to subsequent queries. In the absence of an attack scenario, the traffic generation node remained passive, allowing the collection of a reference dataset of legitimate sessions.

The IDS was implemented as a passive network sensor receiving a mirrored copy of traffic via the Linux Bridge mirroring mechanism. On the IDS node, packet capture was performed, followed by a preprocessing module that extracted 32 bytes from the IPv4, UDP, and DNS headers of each packet. Packets were aggregated using a sliding window of six consecutive packets corresponding to a single DNS session. Each byte was then converted into its 8-bit representation, resulting in a three-dimensional tensor of size $6 \times 32 \times 8$, which was provided as input to the convolutional neural networks. The model returned the session classification (Normal or Attack) along with a confidence probability. In the case of an Attack classification, an IDS log entry and corresponding

alert were generated. Centralized log collection was implemented using rsyslog, enabling event correlation and statistical analysis of the results.

The independent evaluation dataset comprised 1,500 DNS sessions, of which 682 corresponded to DNS cache poisoning attacks and 818 represented legitimate DNS traffic. To evaluate the stability of the proposed IDS under deployment conditions, the trained CNN+ECA model was evaluated four times in an independent virtualized test environment using newly collected DNS traffic captured under the same experimental scenario. The reported results are presented as Mean ± SD together with the corresponding 95% confidence intervals calculated using the Student's t-distribution ($n = 4$).

The testing results indicate that the CNN + ECA model maintains a high level of classification performance. In the independent environment, the proposed model achieved an Accuracy of 93.33 ± 0.78% (95% CI: 92.09–94.56%), which is slightly lower than the performance obtained on the original test dataset but remains sufficiently high for practical IDS applications. Of particular importance is the preservation of a high Recall of 94.94 ± 0.87% (95% CI: 93.55–96.33%), demonstrating the model's ability to effectively detect DNS cache poisoning attacks even under variations in environmental conditions. Since missing an attack is more critical than generating a false alarm in intrusion detection tasks, maintaining high sensitivity confirms the practical applicability of the proposed approach. The MCC also remained stable at 0.8865 ± 0.0057 (95% CI: 0.8774–0.8955), confirming the balanced classification performance of the proposed approach. The slight decrease in performance compared to the initial testing phase is a typical manifestation of data distribution shift between the training and the new environment. Nevertheless, the obtained results demonstrate sufficient robustness of the CNN + ECA architecture to variations in network characteristics.

Thus, the conducted evaluation confirms that the proposed model is not tightly bound to a specific training environment configuration and demonstrates an adequate generalization capability. This property is critically important for the practical deployment of ML-driven IDS solutions in real-world network infrastructures.

5. Conclusion

The proposed approach contributes a novel session-level representation of DNS traffic together with its adaptation to CNN-based architectures and validation in an independent IDS prototype, demonstrating its suitability for practical DNS cache poisoning detection. Unlike traditional signature-based and protocol-oriented mechanisms, the proposed model analyzes a structured tensor representation of DNS traffic, constructed by transforming IPv4, UDP, and DNS headers into a 6× 32×8 format. This approach enables automatic extraction of informative local patterns without manual feature engineering and reduces dependence on static detection rules.

A comparative evaluation of four architectures was conducted: Vanilla CNN, CNN with Batch Normalization and Dropout, Mini-ResNet, and CNN with an Efficient Channel Attention mechanism. Experimental results demonstrate a consistent improvement in performance when transitioning from the baseline architecture to models incorporating normalization, residual connections, and channel attention. The CNN + ECA model achieved the best results, reaching an Accuracy of 96.68%, a Recall of 97.62%, and an MCC of 0.9337 on the test dataset. Particularly noteworthy is the reduction in false negatives, which is critically important for IDS applications.

Unlike previous studies that report only offline experimental results, the proposed approach was implemented as a functioning ML-based IDS prototype operating in a virtualized KVM/QEMU environment (using BIND as the recursive DNS server) and evaluated on an independent testbed. Evaluation in an independent environment confirmed the preservation of high sensitivity (Recall 93.55–96.33%) and stable MCC values (0.8774–0.8955), indicating sufficient generalization capability and robustness of the architecture to variations in network infrastructure parameters.

The obtained results substantiate the feasibility of employing CNN-based architectures with attention mechanisms for DNS traffic analysis and DNS cache poisoning detection.

Future work will focus on several directions. First, an ablation study will be conducted to quantify the contribution of individual IPv4, UDP, and DNS header fields, particularly those that may exhibit environment-specific characteristics, such as IP Identification, TTL, Fragment Offset, and UDP checksum. Furthermore, the proposed approach will be evaluated on more diverse DNS datasets, including encrypted DNS traffic (DoH and DoT), compared with non-convolutional machine learning models, and assessed with respect to online inference latency and deployment in real-world network environments.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to grammar and spell check, and improve the text readability. After using the tool, the authors reviewed and edited the content as needed to take full responsibility for the publication's content.

References

[1] T. H. Kim, D. Reeves, A survey of domain name system vulnerabilities and attacks, J. Surveill., Secur. Saf. (2020). doi:10.20517/jsss.2020.14.

[2] H. Gattu, J. Karimireddy, Kanishka G, DNS Under Siege: Ethical DNS Spoofing and Countermeasures, Int. Res. J. Innov. Eng. Technol. 09.Special Issue (2025) 250–254. doi:10.47001/irjiet/2025.inspire40.

[3] O. van der Toorn, M. Müller, S. Dickinson, C. Hesselman, A. Sperotto, R. van Rijswijk-Deij, Addressing the challenges of modern DNS a comprehensive tutorial, Comput. Sci. Rev. 45 (2022) 100469. doi:10.1016/j.cosrev.2022.100469.

[4] Mahendra S Dalvi, Machine learning based intrusion detection system, J. Inf. Syst. Eng. Manag. 10.36s (2025) 550–555. doi:10.52783/jisem.v10i36s.6528.

[5] Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V. Detection and classification of DDoS flooding attacks by machine learning method. CEUR Workshop Proceedings, (2024), 3842, pp. 184 – 195.

[6] O. Atamaniuk, V. Dudnyk, N. Lysenko, Method for computer network traffic analysis based on entropy characteristics and multivariate mathematical statistics, Comput. Syst. Inf. Technol. (2) (2026) 8–16. doi:10.31891/csit-2026-2-1.

[7] G. Harman, E. Cengiz, Deep Learning Based Network Intrusion Detection, Muhendislik Bilim. Ve Tasar. Derg. 12.3 (2024) 517–530. doi:10.21923/jesd.1417622.

[8] Tymoshchuk, D., Sverstiuk, A., Klots, Y., Petliak, N., Titova, V. An explainable artificial intelligence approach for detecting network attacks. CEUR Workshop Proceedings, 2025, 4141, pp. 38-51.

[9] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, J. Inf. Secur. Appl. 50 (2020) 102419. doi:10.1016/j.jisa.2019.102419.

[10] Tymoshchuk, D., Zagorodna, N, Klots, Y., Yatskiv, V., Petliak, N. AutoML and explainable AI-based approach to enhance the efficiency and interpretability of IDS. CEUR Workshop Proceedings, 2025, 4163, pp. 231-246.

[11] A. Alshammari, A. Aldribi, Apply machine learning techniques to detect malicious network traffic in cloud computing, J. Big Data 8.1 (2021). doi:10.1186/s40537-021-00475-1.

[12] DNS and The Bit 0x20 – Hypothetical Me. URL: https://hypothetical.me/short/dns-0x20/.

[13] Yogesh Kumar Bhardwaj, DNS security: The overlooked vulnerability in modern infrastructure, World J. Adv. Eng. Technol. Sci. 15.2 (2025) 1783–1790. doi:10.30574/wjaets.2025.15.2.0707.

[14] A. Maesya, V. I. Sugara, A. H. Saputra, Application of RPZ-Based DNS Filtering in Educational Network Security, Int. J. Saf. Secur. Eng. 15.7 (2025). doi:10.18280/ijsse.150716.

[15] Q. Abu Al-Haija, M. Alohaly, A. Odeh, A lightweight double-stage scheme to identify malicious DNS over HTTPS traffic using a hybrid learning approach, Sensors 23.7 (2023) 3489. doi:10.3390/s23073489.
[16] M. Behnke, N. Briner, D. Cullen, K. Schwerdtfeger, J. Warren, R. Basnet, T. Doleck, Feature Engineering and Machine Learning Model Comparison for Malicious Activity Detection in the DNS-Over-HTTPS Protocol, IEEE Access (2021) 1. doi:10.1109/access.2021.3113294.
[17] Y. M. Banadaki, Detecting malicious DNS over HTTPS traffic in domain name system using machine learning classifiers, J. Comput. Sci. Appl. 8.2 (2020) 46–55. doi:10.12691/jcsa-8-2-2.
[18] Shashank Biradar, Shramik S Shetty, Pradeep Nayak, Prajakta Shetty, Shwetha R Sharma, A Detection Method Against DNS Cache Poisoning Attacks using Machine Learning Techniques, Int. J. Adv. Res. Sci., Commun. Technol. (2022) 671–675. doi:10.48175/ijarsct-7033.
[19] N. Zagorodna, M. Stadnyk, B. Lypa, M. Gavrylov, R. Kozak, Network Attack Detection Using Machine Learning Methods, Challenges to National Defence in Contemporary Geopolitical Situation 2022 (1) (2022) 55–61. doi:10.47459/cndcgs.2022.7.
[20] A. T. Nguyen, M. Park, Detection of DoH Tunneling using Semi-supervised Learning Method, Proceedings of the 2022 International Conference on Information Networking (ICOIN) (2022) 450–453. doi: 10.1109/ICOIN53446.2022.9687157.
[21] Q.-V. Dang, Studying the Fuzzy clustering algorithm for intrusion detection on the attacks to the Domain Name System, y: 2021 fifth world conference on smart trends in systems security and sustainability (worlds4), IEEE, 2021. doi:10.1109/worlds451998.2021.9514038.
[22] C. Yue, L. Wang, D. Wang, R. Duo, X. Nie, An ensemble intrusion detection method for train ethernet consist network based on CNN and RNN, IEEE Access 9 (2021) 59527–59539. doi:10.1109/access.2021.3073413.
[23] L. A. Chijioke Ahakonye, C. Ifeanyi Nwakanma, S. O. Ajakwe, J. Min Lee, D.-S. Kim, Countering DNS vulnerability to attacks using ensemble learning, in: 2022 international conference on artificial intelligence in information and communication (ICAIIC), IEEE, 2022. doi:10.1109/icaiic54071.2022.9722641.
[24] AIforCybersecurity/Chapter5-DL-for-Detecting-DNS-Cache-Poisoning/data at main · PSUCyberSecurityLab/AIforCybersecurity. URL: https://github.com/PSUCyberSecurityLab/AIforCybersecurity/tree/main/Chapter5-DL-for-Detecting-DNS-Cache-Poisoning/data.
[25] V. J. M. Manes, H. Han, C. Han, S. K. Cha, M. Egele, E. J. Schwartz, M. Woo, The art, science, and engineering of fuzzing: A survey, IEEE Trans. Softw. Eng. (2019) 1. doi:10.1109/tse.2019.2946563.
[26] Batch normalization, in: Graduate Studies in Mathematics, American Mathematical Society, Providence, Rhode Island, 2025, pp. 151–157. doi:10.1090/gsm/252/10.
[27] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.
[28] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, Q. Hu, ECA-Net: efficient channel attention for deep convolutional neural networks, in: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, 2020. doi:10.1109/cvpr42600.2020.01155.
[29] H. Ji, Y. Wang, H. Qin, Y. Wang, H. Li, Comparative Performance Evaluation of Intrusion Detection Methods for In-Vehicle Networks, IEEE Access 6 (2018) 37523–37532. doi:10.1109/access.2018.2848106.
[30] I. Ramskyi, A. Drozd, O. Lyhun, & O. Ponochovna, (2025) System for cybersecurity evaluation of corporate networks. Computer Systems and Information Technologies, (2) 123–131. https://doi.org/10.31891/csit-2025-2-14.
[31] O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots, Botnet detection technique for corporate area network. In: Proceedings of the 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2013), Berlin, Germany, September 12–14, 2013, pp. 363–368.