DNS-Based Data Exfiltration Detection in Cloud Environments: A Stacked Ensemble Approach with Entropy-Aware Feature Engineering and Adversarial Robustness Evaluation

Aliza Kashif
Department of Information and Communication Engineering
The Islamia University of Bahawalpur, Pakistan
aliza3kashif@gmail.com

Rahmeen Tahir
Department of Information and Communication Engineering
The Islamia University of Bahawalpur, Pakistan
rahmeenmalick@gmail.com

Aoun Muhammad
Faculty of Engineering and Technology
The Islamia University of Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

Sana Tariq
Department of Information and Communication Engineering
The Islamia University of Bahawalpur, Pakistan
sana.tariq@iub.edu.pk

***Abstract*—Covert Data Exfiltration (CDE) is always a concern in modern networks as adversaries routinely embed malicious payload within seemingly benign traffic [1], [2]. This is even more complex in the cloud-based infrastructures, where the constant large volume of data that flows in and out provides a lot of protection for the hidden activity [3], [4]. In this study, supervised machine learning techniques are used to detect DNS-based exfiltration methods based on behavioural and structural characteristics of network traffic [5], [6]. The detection problem is outlined as a two-class classification problem and the experimental pipeline includes data cleaning, feature transformation, z-score normalisation, stratified partition creation, mitigating class imbalance, and comparing the performance of multiple models [7]. Classifiers under consideration include Logistic Regression, Support Vector Machine (SVM), Random Forest, XGBoost and a new stacked ensemble [8], [9]. The classifiers are evaluated on six criteria: accuracy, precision, recall, F1-score, AUC and false alarm rate [10]. According to the experimental results, all the tree-based models achieve close-to-same accuracy, and the stacked ensemble is not able to surpass a single Random Forest or XGBoost model [11]. Two new features are added, label-entropy variance and consonant-cluster density, and an ablation demonstrates that they contribute little after the query length is added [5], [12]. It is found that the length of queries has the greatest influence on detection on this benchmark, and that accuracy decreases in cross-dataset scenarios as well as when a realistic entropy-suppression attack is applied, indicating that the high in-distribution numbers do not provide accurate estimates of the reliability of the detection [13], [14].**

***Index Terms*—Data exfiltration, DNS tunneling, machine learning, Logistic Regression, Support Vector Machine, Random Forest, XGBoost, stacked ensemble, cloud security, network anomaly detection, adversarial robustness**

I. Introduction

Cloud connected infrastructure has become the standard for enterprise computing with the ability to scale data storage, scale distributed communication, remote working and delivery of services on-demand [1]. This technological change has at the same time increased the organization's attack surface, finding new methods to covertly acquire sensitive information in controlled areas [2], [15]. Unauthorized data exfiltration—shifting confidential information stealthily outside of the organization's boundaries is among the most serious risks [13], [14]. Sophisticated adversaries do not exfiltrate using obvious methods, but rather through exfiltration payloads designed to fit in amongst the common forms of communication traffic, making traditional monitoring methods relatively ineffective most of the time [1]–[3].

Among the different protocols available, DNS is particularly noteworthy because it is typically an exfiltration vector: it is generally open by default on all networks, it is a very popular protocol for many legitimate queries, and it can be misused to include data in query names, subdomain strings or record-type fields [16], [17]. This results in a hard time to notice malicious DNS-based exfiltration, and malicious actions are frequently not noticed for a long time [6], [12]. However, the rule-based and signature-based detection systems can fail to detect new patterns and/or when an attacker changes the patterns of communication or intentionally mimics them (e.g., reduces the rate of communication or payload length, or changes how the payload is encoded) [13], [18], [19].

Such limitations have inspired a lot of research on the machine learning approach for network threat detection [14], [20], [21]. At the flow level, suitable feature representations can be designed to enable classifiers to be trained in a supervised fashion to recognize differences in traffic flows which are statistical- and/or behavioral-based and not present in fixed signature libraries [5], [22].

This study was stimulated by three gaps in the literature. Firstly, most papers focus on features of the network flow,

DOI: https://doi.org/10.67535/tsp.000003.005
Published under the ICACNC 2026 Proceedings

including bytes per second or packet inter-arrival time, but they do not reference structural features of the flow, such as DNS specific; fingerprint of DNS tunnelling [5], [12]. Secondly, most studies on DNS are based on a single classifier, while stacked ensemble learning methods are scarce [23], [24] that can benefit from the combination of the strengths of a large number of classifiers. Thirdly, few papers have tested the robustness of models against an adversary that can actively change the structure of the queries to which the models are supposed to respond [20], [21].

The main contributions of this paper are four fold:

1) It adds two new DNS-structural features (label-entropy variance and consonant-cluster density) and assesses them using an ablation, demonstrating that these features are mostly redundant to the current set of features found in this dataset.
2) It trains a stacked ensemble of Random Forest and XGBoost as base learners with Logistic Regression as a meta-classifier on out-of-fold predictions and demonstrates that on this data it is not any better than a single tree model.
3) It enables a fair comparison of five classifiers, which were trained and evaluated following the same preprocessing/evaluation protocol across two public DNS exfiltration datasets.
4) It pushes an adversarial evaluation of entropy suppression, query-rate throttling and combination attacks, and identifies which features are discriminative under each attack.

II. Literature Review

Studies on data exfiltration cover multiple intersecting areas, such as insider threat analysis [6], network anomaly detection, and covert channel investigation at the protocol level [13], [14]. In the past, surveys have shown that it is necessary to model behaviour to detect low volume, intermittent, protocol-aware attack patterns, and not only a blacklist-based approach [14], [15], in order to detect effective exfiltrations. Some of the common observations are that classifiers generated using old or limited benchmark databases tend to be poor classifiers when applied to the newer and more diverse network traffic [13], [15].

Experiments conducted in literature have been based on existing intrusion detection benchmarks like KDD99, NSL-KDD, UNSW-NB15, CICIDS2017, CSE-CICIDS2018, and TON_IoT—to assess the performance of classifiers in the various attacks [17], [19], [25], [26]. While these collections are valuable for general purpose machine learning analysis, and have been modified and adapted to many applications, they are not directly applicable to the DNS exfiltration scenario, where the semantics of traffic and the nature of attacks differs greatly from the typical distinctions among intrusion categories [5], [22], [27].

To detect DNS tunnelling in real-time, Abualghanam et al. suggested a feature selection based machine learning approach showing that packet-length statistics, combined with optimum selected traffic attributes, significantly boosted the classification accuracy [5]. Ozery et al. proposed an information-theoretic heavy hitter approach which has been proven to benefit in terms of low latency and the interpretation of the alerts [28]. In a public cloud environment, Salat et al. showed that covert data leakage by DNS tunnelling can be achieved and that traditional firewalls are not sufficient to block it [6].

CNN-LSTM hybrid anomaly detection, autoencoder pipeline, gradient boosting frameworks and ensemble classifiers have been shown to be competitive on tasks related to DNS and tunnelling [29]. Notwithstanding all of these advancements, lightweight decision-tree ensembles are still appealing to use in practice because of their model interpretability, low inference overhead and ability to work well across diverse numerical feature spaces [24], [30]. For certain types of scenarios, such as streaming applications, or when resources are constrained, such as an edge deployment, the false alarm rate and the cost of inference per sample may be more relevant to the practical impact than the peak accuracy metrics [11], [31].

Ziza et al. studied the robustness of classifiers in a deliberately adverse setting, and found that a significant decrease in detection performance is possible, even with the use of parameter variation or deliberate entropy reduction by the adversary [20]. In order to detect low-rate, slow-paced exfiltration campaigns with high detection rates, Mahdavifar et al. suggested a lightweight hybrid classifier that can run in two stages [21]. Ahmed et al. proposed a DNS monitoring architecture applicable at the enterprise scale, which allows to identify the exfiltration from an internal network endpoint in real time [32]. Almuhanna and Dardouri came up with a multi-architecture anomaly detection system comprising of XGBoost, Random Forest, graph neural networks, LSTM, and autoencoders, which enabled the complete classification of traffic in fifteen classes [23].

There is no literature available that provides a reproducible example of such a combined novel DNS specific feature with a stacked ensemble architecture with systematic adversarial evaluation, however [14], [20], [21], [23]. This work is filling in that gap.

III. Methodology

The five steps in the experimental pipeline are: dataset selection, data preprocessing, feature engineering, model training and quantitative model performance evaluation [5], [7], [13].

A. Dataset Selection

Two sets of labels are provided to help reduce arriving at the conclusion that the only way to draw conclusions is by using one benchmark.

Dataset A – Netrack [22]: A set of DNS query packets generated from a normal network environment and labelled malicious packets created using tools such as `iodine`, `dns2tcp`, `dnscapy` and `tuns`, and placed in a dataset that is open and accessible. Multi-class (one benign class and four tool classes) raw labels are mapped to a binary label; the four tool classes

are considered malicious and the normal traffic is considered benign. Following pre-processing, there are 40,000 flows used, of which 15,291 are benign and 24,709 malicious, around 62% of which are malicious. This high malicious share is not necessarily due to the algorithm performance, but rather to the construction of Netrack (a fixed 4:1 malicious-to-benign ratio in the training set, with a near-balanced test set) rather than a real-world prevalence, and the balanced (inverse-frequency) class weights used in training compensate for that.

Dataset B – Daumel [27]: An alternative collection, taking place under different conditions and using a different tool set. Post-deduplication: 31,840 flows (21,230 benign, 10,610 malicious), class ratio approximately 67:33.

The observations are labeled with a binary label: $y = 0$ represents benign traffic while $y = 1$ represents exfiltration traffic [5], [22].

B. Feature Engineering

We only include attributes of the query-string of a DNS query—we do not include generic network-flow statistics—for two reasons: (1) they are available earlier in the communication timeline and (2) they capture protocol-specific structure that flow-level aggregates miss. Eight features are obtained for each instance of DNS transaction, six of these have already been published in the literature and two are original contributions of this work [5], [12].

1) **Query entropy** (H_q): Shannon entropy of the distribution of query string characters (full query string).
2) **Query length** (L_q): Number of characters in a query string.
3) **Subdomain depth** (D_s): Number of labels between the registered domain and the subdomain.
4) **Longest label length** (L_{max}): Length (in characters) of the longest label.
5) **Digit ratio** (R_d): Ratio of the number of characters that are numeric digits.
6) **Uppercase ratio** (R_u): Percentage of the characters in the string that are uppercase alphabetic characters.
7) **Label entropy variance** (σ_H^2) **[Novel]:** Variance of Shannon entropy over the subdomain labels. The concept here is to capture the inter-label variation that is not captured by the mean-entropy features when a tunnelling tool pads queries with legitimate looking labels to reduce the mean entropy.
8) **Consonant-cluster density** (C_{cc}) **[Novel]:** Maximal consonant-run length / Query length. Unnatural consonant clusters not found in natural language domain names are possible with binary-encoded payloads.

Label entropy variance is mathematically defined as:

$$\sigma_H^2 = \frac{1}{n}\sum_{i=1}^{n}\left(H(\ell_i) - \bar{H}\right)^2 \tag{1}$$

where ℓ_i denotes the i-th subdomain label, $H(\ell_i)$ the Shannon entropy of the subdomain label ℓ_i, $\bar{H}$ the average entropy over the labels, and n the number of labels.

C. Data Preprocessing

Pre-processing is a series of three steps: encoding of categorical variables by integer mapping, missing value resolution by median imputation and normalisation of continuous variables by z-score standardisation [7], [13]:

$$z = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}} \tag{2}$$

where μ_{train} and σ_{train} are calculated over the training folds only, excluding information leakage into the held-out fold. Stratified 5-fold cross-validation is used to evaluate the data and the result is averaged over the five folds as reported in Table I, the adversarial tests are performed on a single held-out fold, while the confusion matrix is performed on a randomly selected fold. The class imbalance is addressed by (inverse-frequency) class weights during training, and stratified sampling maintains the original class distribution across each fold [7].

D. Model Set

There are five classifiers to be compared:

1) Logistic Regression (LR)
2) Support Vector Machine (SVM)
3) Random Forest (RF)
4) Extreme Gradient Boosting (XGBoost)
5) Stacked Ensemble (RF + XGBoost → LR meta-learner) **[Novel]**

The stacked ensemble is based on a two level generalisation approach [33]. The data of the train set is divided into five folds. In each fold, RF and XGBoost are trained on the other four folds, and make predictions for the out-of-fold data for the held out fold. These estimates are used to create a meta-feature matrix $\mathbf{Z} \in \mathbb{R}^{n_{\text{train}} \times 2}$ given as input to the Logistic Regression meta-learner. This procedure is called out-of-fold as it makes sure that the meta-learner never sees an instance used to train the base learners. Fig. 1 shows the architecture.

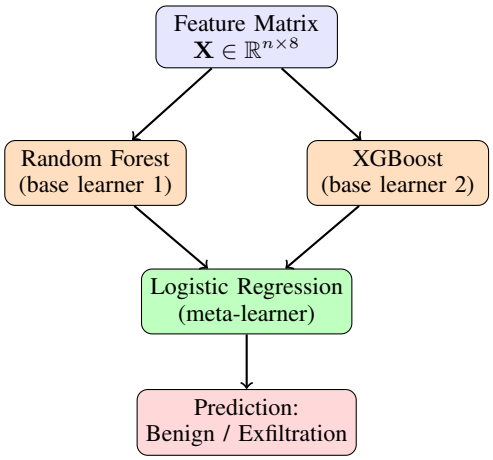

Fig. 1. Proposed stacked ensemble architecture.

No hyperparameter search was performed. All settings were kept identical for all models, and equal across all experiments:

RF – 300 trees; max depth 12; min samples per leaf 4; balanced class weights; XGBoost – 400 estimators; learning rate 0.05; max depth 8; subsample 0.8; per-fold positive-class weight $n_{\text{neg}}/n_{\text{pos}}$; LR (baseline and meta-learner) – `lbfgs` solver; balanced class weights; default $C = 1.0$; SVM – RBF kernel; balanced class weights. The default parameters are used in the library, and the 5-fold cross-validation is only used for performance estimation, not to select the models.

E. Evaluation Metrics

The following standard metrics are used to quantify the performance [10]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

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

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

$$\text{FAR} = \frac{FP}{FP + TN} \tag{7}$$

All results are presented as the mean of the 5-fold stratified cross-validation. The ROC curve is used to calculate AUC. From the operational standpoint, FAR has an important role, as a very noisy classifier, even if it has a good nominal accuracy, will burden the analyst workflows [10], [11], [31].

F. Adversarial Evaluation

Three test conditions are constructed that are adversarial towards malicious samples only, following the work of Ziza et al. [20].

- **Entropy suppression:** The encoding of the payload is preserved and the entropy is decreased by approximately 30% by introducing common English trigrams to subdomain labels.
- **Rate throttling:** The interval between queries in the exfiltrated traffic was increased to be as close as possible to the 90th percentile interval between queries of benign traffic from the same source IP.
- **Combined attack:** Both modifications are used together, which makes the opponent stronger.

IV. Experimental Results and Discussion

A. Overall Model Comparison

All metrics for the models are reported in Table I for Dataset A [5], [8], [9] for the direct comparison. High accuracy is achieved for all classifiers, even Logistic Regression, which is above 0.98. The three tree-based models (RF, XGBoost and the stacked ensemble) are very close, with an accuracy close to 0.9995, Thus, neither an XGBoost nor a Random Forest model is defeated by the stacked ensemble.

All the tree models are close to 1.0 for AUC and hence do not separate them [8], [9]. The stacked ensemble also requires a huge amount of training time ($\approx$113 s versus $\approx$7 s for XGBoost) with no improvement in accuracy [11]. The benefits of the different models are only evident in cross-dataset tests and the adversarial attack (discussed below).

B. Cross-Dataset Generalisation

Table II displays the results of training on Dataset A and testing on Dataset B, but the performance is not just a little worse, it is significantly worse: F1 drops to between 0.57 and 0.66 and the false alarm rate increases considerably. XGBoost generalises best and the stacked ensemble is not the most robust. This indicates that the models primarily capture patterns associated with the specific datasets and tools, but not the general concept of exfiltration.

TABLE II
Cross-dataset generalisation: trained on Dataset A, tested on Dataset B

Model	Acc.	F1	Recall	FAR
Random Forest	0.700	0.569	0.456	0.113
XGBoost	0.721	0.662	0.631	0.210
Stacked Ensemble	0.703	0.628	0.577	0.199

C. Confusion Matrix

For Dataset A ($n = 8{,}000$: benign $= 3{,}058$; exfiltration $= 4{,}942$), the cross-validated value of $TN = 3{,}057$, $FP = 1$, $FN = 2$, $TP = 4{,}940$ is obtained from the stacked ensemble for a single held-out fold:

	Pred. Benign	Pred. Exfil.
Actual Benign	TN = 99.97%	FP = 0.03%
Actual Exfil.	FN = 0.04%	TP = 99.96%

D. Feature Importance

The feature importance (mean decrease in impurity) of the Random Forest model is shown in Fig. 2. The highest importances are digit ratio (R_d), subdomain depth (D_s) and query length (L_q), while the two novel features rank low, and all importances are stable across folds (SD below 0.004). The features are highly correlated (long tunnel queries are also high on digit content and are deep in structure), so a high score here may not necessarily indicate that a feature is needed. This is directly verified by the ablation in Section IV-F.

E. Adversarial Robustness

Table III shows the results of F1 and malicious recall with a DNS-realistic entropy-suppression attack [20]. The attack is maintained to be plausible: each label is limited to 63 characters (per RFC) and the total length of a query is limited to the 95th percentile of benign query length (34 characters), and entropy is reduced by appending fragments of natural-language labels without altering the payload. All three models drop together, by about 17–18 points of F1 and around 30 points of recall, with Random Forest only moderately better.

TABLE I
COMPARATIVE PERFORMANCE OF EVALUATED MODELS ON DATASET A
(MEAN ± STD OVER 5-FOLD STRATIFIED CROSS-VALIDATION)

Model	Acc.	Prec.	Rec.	F1	FAR	AUC
LR	$0.9891 \pm .0010$	$0.9859 \pm .0017$	$0.9966 \pm .0005$	$0.9912 \pm .0008$	$0.0230 \pm .0029$	0.9991
SVM	$0.9977 \pm .0003$	$0.9994 \pm .0004$	$0.9968 \pm .0005$	$0.9981 \pm .0002$	$0.0010 \pm .0006$	0.9998
RF	$0.9995 \pm .0002$	$1.0000 \pm .0001$	$0.9992 \pm .0003$	$0.9996 \pm .0001$	$0.0001 \pm .0001$	1.0000
XGB	$0.9995 \pm .0001$	$0.9997 \pm .0003$	$0.9995 \pm .0002$	$0.9996 \pm .0001$	$0.0005 \pm .0004$	1.0000
Stack	$0.9995 \pm .0001$	$0.9998 \pm .0002$	$0.9994 \pm .0003$	$0.9996 \pm .0001$	$0.0003 \pm .0004$	1.0000

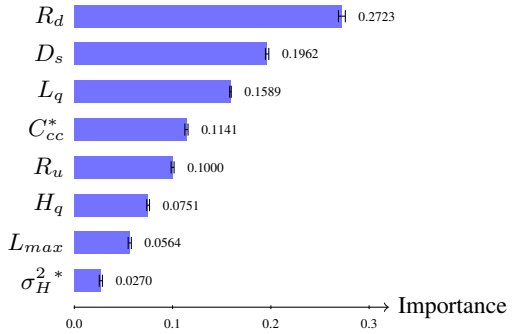

Fig. 2. Random Forest feature importances (mean decrease in impurity, mean ± 1 SD across the five cross-validation folds). Superscript * marks novel features.

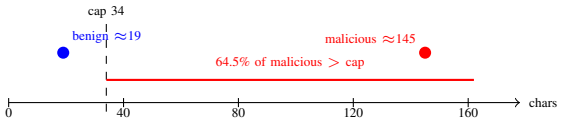

Fig. 3. Query-length separation on Dataset A. Benign queries average 19 characters and malicious 145; 64.5% of malicious queries exceed the 34-character attack cap and stay detectable by length alone.

There is no effect through rate throttling and the combined attack is equivalent to entropy suppression. These numbers are from one held-out fold, but the drop is due to the attack and not to random variation across models (as the three models almost match in their drops, about a point or less).

The only malicious queries that can be altered are the 35.5% that are short enough to fall below the cap; if a query is long, it is still detected as malicious (Fig. 3). The length alone is close to a sufficient discriminant on this dataset.

TABLE III
ADVERSARIAL ROBUSTNESS UNDER THE DNS-REALISTIC ENTROPY-SUPPRESSION ATTACK (CLEAN F1 ≈ 1.000; RATE THROTTLING STAYS AT BASELINE; COMBINED = ENTROPY SUPPRESSION)

Model	Clean F1	Attack F1	Attack Rec.	ΔF1
RF	0.9998	0.828	0.706	−0.172
XGB	0.9997	0.820	0.696	−0.179
Stack	0.9997	0.823	0.699	−0.177

F. Feature Ablation

Random Forest is retrained with and without the novel features, and with only the length features (Table IV), to determine whether the novel features help. The two new features come at a very low cost, with 0.27 of an F1 point sacrificed under attack and 0.02 on clean data (almost nothing). A length-only model costs 4.4 points under attack, so the residual signal beyond length comes from the other character features (mainly digit ratio and subdomain depth), not from the novel characters. In this dataset the two proposed features are thus redundant and are not reported as detection gains but as a negative result.

TABLE IV
FEATURE-SUFFICIENCY ABLATION (RANDOM FOREST, F1)

Configuration	Clean F1	Attack F1
Full (8 features)	0.9998	0.8275
w/o novel (σ^2_H, C_{cc})	0.9996	0.8248
Length only (L_q, L_{max})	0.9986	0.7832

G. Discussion

There are three highlights. First, with this set of queries, length is the sole distinguishing feature of malicious queries from benign ones (on average, a malicious query is 145 characters long, whereas a benign query is 19 characters long), making a good accuracy on this set by itself not a good indication of a good detector. Second, the two novel features have no measurable effect once query length is incorporated, and the ablation suggests that they are redundant. Thirdly, the stacked ensemble does not bring any improvement over a single tree model and is more expensive to train. Across datasets and under a realistic attack, accuracy also drops significantly, so the in-distribution numbers overstate how robust the detection really is [5], [20], [33].

A drawback is that both datasets were collected during a test phase with specific tunnelling applications; other adversaries with unique and non-standard implementations of tunnelling may result in a different pattern of traffic. A further constraint is that all of the features are computed from the cleartext query string, so when the query is encrypted (using encrypted DNS, DoH or DNS-over-TLS), the query name is not exposed and these features cannot be computed from passive network traffic

without logging by the resolver or endpoint. Further research activities will be to add the evaluation of DoH (DNS-over-HTTPS) traffic and to test the stacked ensemble in the live SIEM integration pipelines [4], [6], [21], [32].

V. Conclusion

This study is performed on two datasets, including cross-dataset and adversarial tests [6], [13], [14]. All classifiers achieved high in-distribution accuracy, but the accuracy of the stacked classifier did not surpass that of a single Random Forest or XGBoost, and decreased drastically when tested with cross-dataset test data and under an attack with realistic entropy suppression [2], [8], [10]. The primary reason for the high scores is that malicious queries tend to be much longer than benign ones, and length alone is almost sufficient to discriminate between the classes on this benchmark.

Consequently, the two proposed features, consonant-cluster density and label-entropy variance, were not found to be useful when the query length is also available, and we report this as a negative result [5], [12], [28]. Further testing on additional tunnelling tools and encoders, encrypted DNS (DoH) traffic, and robust training to resist these attacks should be conducted in future research [4], [21], [23], [32].

References

[1] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 20, pp. 1–22, 2019.

[3] M. Sammour, M. F. I. Othman, A. Hassan, O. Bhais, and M. S. Talib, "Advanced DNS tunneling detection: A hybrid reinforcement learning and metaheuristic approach," *Frontiers in Computer Science*, vol. 7, p. 1728980, 2026.

[4] H. S. Talabani, Z. K. Abdul, and H. M. M. Saleh, "DNS over HTTPS tunneling detection system based on selected features via ant colony optimization," *Future Internet*, vol. 17, no. 5, p. 211, 2025.

[5] O. Abualghanam, M. Alazzam, A. Sharieh, and M. Alshinwan, "Real-time detection system for data exfiltration over DNS tunneling using machine learning," *Electronics*, vol. 12, no. 6, p. 1467, 2023.

[6] L. Salat, J. M. Such, and G. Stringhini, "DNS tunnelling, exfiltration and detection over cloud environments," *Sensors*, vol. 23, no. 6, p. 3195, 2023.

[7] G. Lemaître, F. Pedregosa, G. Varoquaux, and A. Gramfort, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[11] A. Samy, H. Yu, and H. Zhang, "Fog-based attack detection framework for internet of things using deep learning," *IEEE Access*, vol. 8, pp. 74 571–74 585, 2020.

[12] I. Homem, P. Papapetrou, and S. Ridenour, "Detecting DNS tunnels using character frequency analysis," in *Proceedings of the IEEE Conference on Communications and Network Security Workshops*, 2016, pp. 1–6.

[13] I. Ullah and Q. H. Mahmoud, "Machine learning for detecting data exfiltration: A review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–36, 2021.

[14] B. Sabir, F. Ullah, M. A. Babar, and R. Gaire, "Machine learning for detecting data exfiltration: A review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–47, 2021.

[15] R. Bejtlich, *The Practice of Network Security Monitoring*. San Francisco, CA, USA: No Starch Press, 2013.

[16] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.

[17] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proceedings of the Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[18] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[20] K. Žiža, P. Tadic, and P. Vuletić, "DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour," *International Journal of Information Security*, vol. 22, no. 6, pp. 1865–1880, 2023.

[21] S. Mahdavifar, A. H. Salem, P. Victor, A. H. Razavi, M. Garzon, N. Hellberg, and A. H. Lashkari, "Lightweight hybrid detection of data exfiltration using DNS based on machine learning," in *Proceedings of the 11th International Conference on Communication and Network Security (ICCNS)*, 2021, pp. 80–86.

[22] Netrack, "Labeled DNS exfiltration datasets and algorithms of DNS tunneling detection," GitHub repository, 2022, [Online]. Available: https://github.com/netrack/learn.

[23] R. Almuhanna and S. Dardouri, "A deep learning/machine learning approach for anomaly based network intrusion detection," *Frontiers in Artificial Intelligence*, vol. 8, p. 1625891, 2025.

[24] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for IoT networks," *Future Generation Computer Systems*, vol. 127, pp. 276–285, 2022.

[25] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009, pp. 1–6.

[26] N. Moustafa, G. Creech, and J. Slay, "TON_IoT datasets: A new generation of benchmark datasets for IoT and IIoT intrusion detection systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.

[27] Daumel, "DNS exfiltration dataset," GitHub repository, 2022, [Online]. Available: https://github.com/Daumel/dns-exfiltration-dataset.

[28] Y. Ozery, A. Nadler, and A. Shabtai, "Information-based heavy hitters for real-time DNS data exfiltration detection," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2024.

[29] H. Alzahrani, T. Sheltami, A. Barnawi, M. Imam, and A. Yaser, "A lightweight intrusion detection system using convolutional neural network and long short-term memory in fog computing," *Computers, Materials & Continua*, vol. 80, no. 3, pp. 4703–4728, 2024.

[30] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," *IEEE Access*, vol. 8, pp. 167 059–167 068, 2020.

[31] A. A. Wardana, G. Kołaczek, and P. Sukarno, "Lightweight, trust-managing, and privacy-preserving collaborative intrusion detection for internet of things," *Applied Sciences*, vol. 14, no. 10, p. 4109, 2024.

[32] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, and V. Sivaraman, "Monitoring enterprise DNS queries for detecting data exfiltration from internal hosts," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 265–279, 2020.

[33] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.