IoT Botnet Detection with Drift-Aligned Learning and DNS-Based C2 Identification

Jeffrey A. Adjei
Faculty of Computer Science
Dalhousie University
Halifax, Canada
jeffrey.adjei@dal.ca

Nur Zincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, Canada
zincir@cs.dal.ca

Malcom Heywood
Faculty of Computer Science
Dalhousie University
Halifax, Canada
mheywood@cs.dal.ca

Biswajit Nandy
Solana Networks
Ottawa, Canada
bnandy@solananetworks.com

Nabil Seddigh
Solana Networks
Ottawa, Canada
nseddigh@solananetworks.com

***Abstract*—Botnet detection in Internet of Things (IoT) environments remain challenging due to evolving attack behaviors and limited generalizability of traditional detection methods. This paper introduces a lightweight, two-stage detection framework that combines a Random Forest classifier with a Botnet Validation Layer. The proposed system leverages a Drift-Aligned Learning Curve (DALC) to adapt to data drifts by incorporating Domain Name System (DNS) based analysis for botnet validation particularly in Command & Control (C2) activities. We utilize a feature set comprising of 20 flow level and 4 DNS level features. Evaluated on diverse datasets, the proposed system achieves 99.7% F1-Score during testing while demonstrating strong generalization.**

***Index Terms*—IoT, Botnet, Command & Control, AI/ML.**

I. INTRODUCTION

The IoT architecture is divided into four levels: applications, middleware, networks, and perception. Various sensors and data-transmitting devices make up the perception layer. Controlled data transit occurs between the network layer and the application layer [1]. The middleware layer stores and handles the data, while the application layer controls the user interface in the application. DNS servers are instrumental in IoT networks for the naming, identification, and connection to the back-end servers of IoT devices. However, botnets also leverage the DNS service to establish communication with their C2 servers utilized by their botmasters [2] - [5]. C2 is also used by IoT devices to trigger specific actions and receive updates on device status. The DNS functions enable customization and automation of tasks, thus transforming static devices into dynamic and responsive systems [6]. This suggests legitimate and illegitimate IoT communications where legitimate C2 via DNS sends updates and automate tasks [7] and illegitimate C2 involves hostile actors taking control of IoT devices without authorization for destructive actions [8]. Botnets are challenging due to their many variants and their dominance in IoT Distributed Denial of Service (DDoS) attacks [9]. New variants are increasingly intelligent, with some exhibiting behaviors similar to legitimate traffic, while others carry out attacks such as Slowloris. To address this, we propose an adaptive Botnet Validation Layer leveraging DALC and DNS-based C2 validation. This lightweight system uses 20 flow-level features and 4 DNS features, improving upon our previous work [10]. The approach is two-layered: the first layer classifies traffic as Benign, DDoS, or Botnet using flow-based features, while the second layer performs deeper analysis with DNS features to distinguish legitimate from malicious DNS-based C2 activity.

Furthermore, several research efforts [11]– [13] have proposed strategies to enhance generalizability. However, performance degradation remains a challenge as data drifts over time. To further explain the generalizability of the proposed system, we evaluate it across three diverse datasets. We train the system using the CIC IoT Dataset 2023 [14], and test it on the Army Cyber Institute IoT Dataset 2023 (ACI-IoT-2023) [15] and the IoT Network Intrusion Dataset [16]. The results show high performance, generalizability, efficient CPU usage, and rapid deployment across different platforms. In summary, this paper makes the following key contributions:

- A lightweight framework for the detection of Benign traffic, Botnet activity, and DDoS attacks, using flow metadata with Machine Learning (ML).
- A champion feature set that enables effective pattern inference on previously unseen and unlabeled data.
- A DALC methodology that applies a multi-resolution approach based on champion feature set.
- A DNS analysis component to reveal C2 presence within Botnet and Benign traffic.

In the following, motivation is introduced in Section II. The design, and development are presented in Sections III and IV, respectively. Results are discussed in Section V, related works are summarized in Section VI, and conclusions are drawn in Section VII.

978-3-903176-75-1 ©2025 IFIP

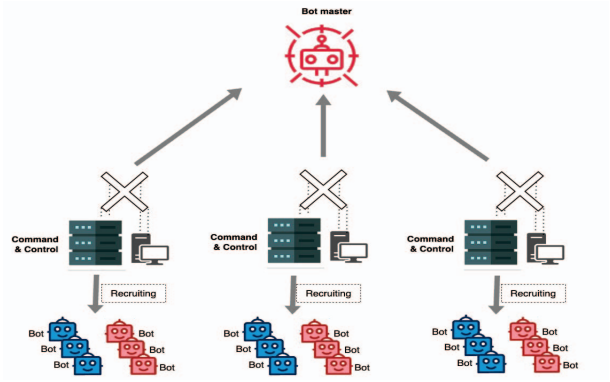

Fig. 1: Command and Control presence in Botnets

II. BACKGROUND AND MOTIVATION

Botnet Detection Systems: Botnets remain one of the most persistent and evolving threats in IoT networks. Botnet Detection Systems (BDS) aim to identify compromised devices participating in coordinated malicious activities. BDS techniques are based on static signatures, and IP blacklists, which offer limited effectiveness against stealthy or rapidly evolving botnets [21], [22]. Recent research has shifted toward behavior-based and ML-driven strategies, which analyze traffic patterns, temporal characteristics to uncover botnet activity. However, challenges persist, including the need for models that generalize well to unseen botnet behaviors and support near real-time detection.

Why Command and Control?: C2 communication forms the backbone of botnet operations, Fig. 1, enabling coordination and data exfiltration. Detecting C2 is vital for early identification of infected hosts, often before overt malicious activity [11]. Yet, C2-like traffic also arises from benign activities such as updates, telemetry, and cloud synchronization. Malicious C2 often exhibit stealthy "low-and-slow" behavior, making them strong compromise indicators and giving BDS a proactive defense edge. Existing systems rely mainly on blacklists and TLS JA3 fingerprints. In contrast, this paper introduces a modular approach using heuristic flow analysis and DNS-based detection to uncover malicious C2 patterns.

III. PROPOSED SYSTEM DESIGN

Our proposed system consists of two stages. In the first stage, a Random Forest classifier identified as the best-performing model in a comparative evaluation of multiple ML models [10] is trained using our champion feature set. This model is designed to classify traffic as Botnet, Benign, or DDoS, and serves as the base model for future analysis and deployment. Further analysis is then made in the next stage to introduce a heuristic approach to identify the presence of malicious C2 behavior within both botnet and benign traffic.

A. Training Workflow

Fig. 2 presents the overview of the two-stage proposed system, consisting of the procedure, in which data are sampled from the training data, namely CIC IoT 2023. This sets the base for the analysis for the initial (re)training procedure.

B. Data and Traffic Flow Sampling

Our Random Forest (RF) Classifier was trained on CIC IoT dataset 2023 [14] which is a benchmark for large-scale attacks in an IoT environment composed of 105 devices. We sampled botnets (greeth, greip, udpplain), benign, and DDoS (Slowloris, Synflood, HTTPflood) traffic to form the training dataset. Our approach implements a time-based sampling strategy to extract a consistent 60-minute window of network traffic flows from each class. The window covers 990.35mins for Benign, 61870.30mins for botnet, and 72303.27mins for DDoS. To add complexity to the dataset, we included the distribution of flows across unique `srcIP:srcPort` combinations for each class. We analyzed the presence of DNS within the flows, where benign traffic had 7,345 DNS queries (133 unique), botnet traffic had 11,550 DNS queries (266 unique), and DDoS traffic had 7,938 DNS queries (223 unique). IP and port-based features were not used in this system to avoid overfitting and also maintain generalizability.

C. Preprocessing, Feature Analysis and ML Optimization

We adopted the preprocessing procedure used in our prior work [10], [19], where we demonstrated robustness in handling data quality issues such as null values and empty columns. To identify optimal features, we began with the 60 features from our previous work [10] as a baseline. To improve computational efficiency and model interpretability, we evaluated feature importance using Mutual Information Gain (MIG) and Mean Decrease Impurity (MDI). Comparing the top-ranked features from both metrics showed strong agreement, particularly among the top 20: 7 features overlapped in the top 10, and all 20 matched in the top 20 (Table I). Based on this alignment, we selected the top 20 features to reduce redundancy without sacrificing predictive performance. Additionally, we identified four proposed DNS features for C2 analysis. Moreover, the RF model was fine-tuned via grid search and cross-validation, yielding 100 estimators, unlimited tree depth, a minimum of 2 samples per split, and 1 sample per leaf. Node feature selection used the square root of total features, with splits based on MDI. Bootstrap sampling enhanced ensemble diversity, balancing complexity, generalization, and computational efficiency.

IV. PROPOSED SYSTEM DEVELOPMENT

The developed Botnet Validation Layer is built using Python. To evaluate our approach, we employ two distinct test datasets. In this section, we provide details on the implementation of the system in a test environment.

A. Validator P1 - Key

This component is crucial for deployment, where labels are unknown. We use our 'key' to infer whether sampled flows are botnet or benign. The validator batches unlabeled test data and extracts the champion feature set plus DNS features using the Tranalyzer (T2) [17] tool. Each feature is then partitioned into

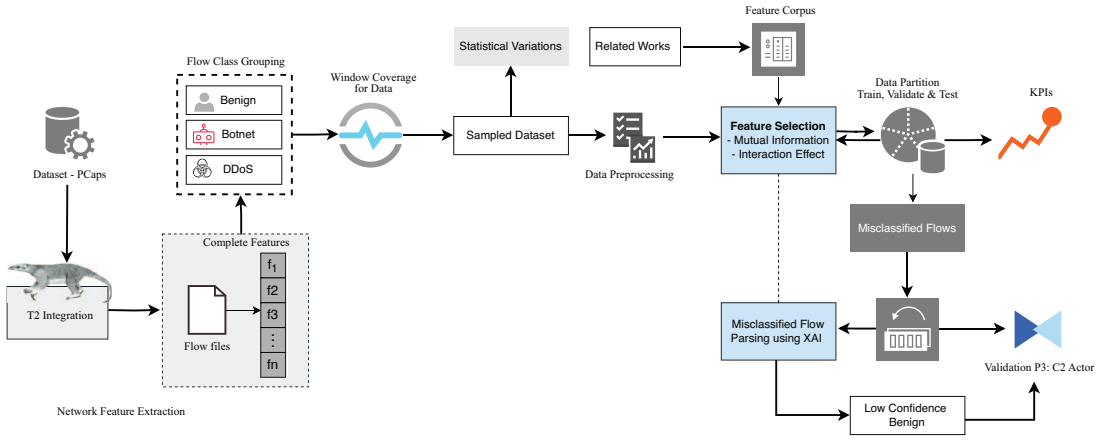

Fig. 2: Overview of the framework for ML Training and Testing

TABLE I: Comparison of Feature Importance: MIG vs. MDI

Rank	MIG	MIG Score	MDI	MDI Score
1	avgL7PktSz	0.6886	dsMinPl	0.1697
2	dsUppQuartilePl	0.6884	minL7PktSz	0.1149
3	dsMeanPl	0.6883	PyldBinRatio	0.0984
4	L2L3L4Pl_Iat	0.6874	L2L3L4Pl_Iat	0.0979
5	dsMedianPl	0.6871	Ps_Iat_Cnt_...	0.0873
6	dsModePl	0.6862	tcpStatesAFlags	0.0483
7	Ps_Iat_Cnt_...	0.6853	avgL7PktSz	0.0417
8	dsLowQuartilePl	0.6851	l7BytesSnt	0.0394
9	dsMinPl	0.6848	dsMeanPl	0.0389
10	minL7PktSz	0.6780	dsModePl	0.0299
11	maxL7PktSz	0.6616	dsMedianPl	0.0294
12	dsMaxPl	0.6613	dsUppQuartilePl	0.0287
13	pktAsm	0.6135	dsLowQuartilePl	0.0251
14	bytAsm	0.6068	pktsRcvd	0.0174
15	PyldChRatio	0.4712	pktAsm	0.0169
16	PyldBinRatio	0.4688	PyldEntropy	0.0148
17	PyldEntropy	0.4526	maxL7PktSz	0.0145
18	l7BytesSnt	0.4492	PyldChRatio	0.0115
19	tcpStatesAFlags	0.4093	dsMaxPl	0.0098
20	pktsRcvd	0.3589	bytAsm	0.0097

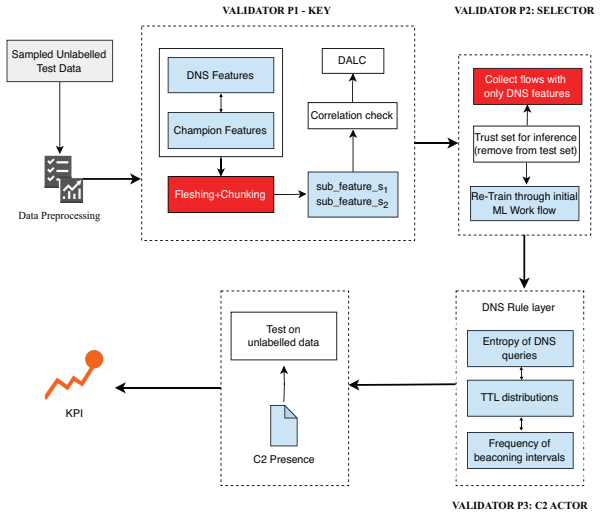

Fig. 3: The Botnet Validation Layer

two chunks based on character count, vectorized correlation, and statistical variance, and then is labeled as features[1...n].

B. Validator P2 - Selector

Once the key identifies the matching flow, it is set as the Trust feature set. Two options are then given: (i) **Full Adaptation** retrains the model with the Trust Set to improve generalization for evolving attacks; (ii) **Fast Inference** uses DNS-extracted features directly, skipping retraining for low-latency detection and reduced compute. The latter is recommended only if the Trust Set exceeds 20% of the data.

C. Validator P3: C2 Actor

This component detects malicious C2 activity using DNS flow features. In IoT environments, overlaps between benign and malicious traffic, especially at scale, make C2 detection challenging. While C2 patterns are clear in controlled setups, large IoT networks with thousands of devices add variability. We use `dnsQname`, `dnsQType`, `dnsATTL`, and `dnsOptStat` with a rule-based heuristic: a flow is flagged only if at least two of these features are non-null. We define a detection score $S(x)$ for a DNS query x, calculated using the following function:

Where:

- $\mathbb{1}[\cdot]$: Indicator function (1 if true, 0 otherwise)
- $H(x)$: Subdomain entropy
- $R_{\text{hex}}(x)$: Hex character ratio
- $TTL(x)$: Time-To-Live of the DNS query
- T_{exp}: Expected TTL for standard DNS types (e.g., A, MX)
- $L(x)$: Subdomain query length

The thresholds shown in the equation in Fig. 4 are dynamically updated during model retraining. A key indication is to ensure that the subdomain (Fig. 9) is not altered between the inference of C2 in Benign and Botnet. This ensures generalizability of the detection rule to evolving data while keeping the rule structure robust.

$$S(x) = 2 \cdot \mathbb{1}[\text{NXDOMAIN}] + 1.5 \cdot \mathbb{1}[H(x) > 3.5] + 1 \cdot \mathbb{1}[R_{\text{hex}}(x) > 0.4] + 1 \cdot \mathbb{1}[TTL(x) \notin T_{\text{exp}} \pm 30\%] + 0.5 \cdot \mathbb{1}[L(x) > 30] \quad (1)$$

Fig. 4: Rule-Based C2 Detection Score Computation

Algorithm 1: Drift-Aligned Learning Curve (DALC)

Input: Flow sequence F, reference patterns P
Output: Assigned label L (if match found)

1. Feature Setup
17 continuous features → `float64`
3 discrete features → `int32`

2. Chunking
Continuous: Sliding window (30% overlap, 5% length), z-score normalize each window
Discrete: Segment on state transitions & thresholds individually for all 3

3. Batching
if $|F| > 10,000$ *divide into batches* **then**
end

4. Pattern Matching
foreach *batch flow* f *in* F **do**
  Compare 1/2 chunks with P
  if *(1 match in 2 chunks)* ***and*** $\geq$*10 feature matches* **then**
    Assign $L \leftarrow$ label from matching pattern
  end
end
Return L

D. T2 Integration

To enable direct execution of the proposed system, we integrated the T2 flow analysis tool. We deployed with TCPdump allowing us to process live packets in real-time, without exiting or interrupting the main system workflow.

V. EVALUATIONS AND RESULTS

We first analyze CIC IoT 2023 data, split into 60% train, 20% validation, and 20% test sets. We then justify using 20 features, highlighting reduced computation time and CPU usage. All experiments were conducted on an 11th Gen Intel Core i7 desktop (3.6,GHz, 16 CPUs, 16,GB RAM).

A. RF Analysis with 60 Features

With 60 features, the RF model was trained in 509.48 seconds, utilizing 835.12 MB of memory. The training process took 128.02 seconds of user CPU time and 7.47 seconds of system CPU time. The resulting RF model had an average tree depth of 38.59, with the deepest tree reaching a depth of 50. On average, each tree consisted of around 9,725 nodes, and the most complex tree had 9,979 nodes.

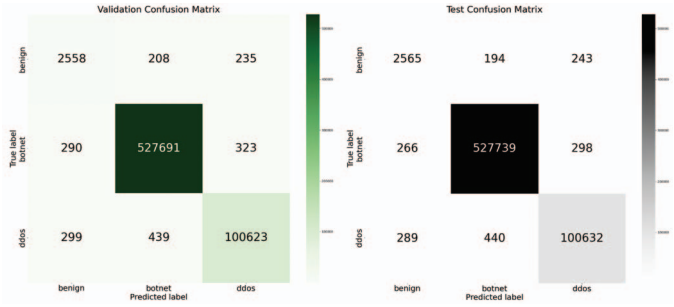

Fig. 5: RF Model Confusion Matrix - 60 Features

TABLE II: RF Model Results using 60 Features

Validation Set Results				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	81%	85%	83%	3,001
Botnet	100%	100%	100%	528,304
DDoS	99%	99%	99%	101,361
Accuracy	-	-	99.72%	632,666
Macro Avg	94%	95%	94%	632,666
Weighted Avg	100%	100%	100%	632,666
Test Set Results				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	82%	85%	84%	3,002
Botnet	100%	100%	100%	528,303
DDoS	99%	99%	99%	101,361
Accuracy	-	-	99.73%	632,666
Macro Avg	94%	95%	94%	632,666
Weighted Avg	100%	100%	100%	632,666

B. RF Analysis with 20 Features

With the 20 selected features, the RF model was trained in 303.63 seconds, and utilized 72.77 MB of memory. It utilized 78.73 seconds of user CPU time and 4.50 seconds of system CPU time. The resulting RF model had an average tree depth of 35.88 with the deepest tree reaching a depth of 45. On average, each tree had 9,310 nodes, with the largest tree having 9,585 nodes.

This shows that the reduced number of features leads to

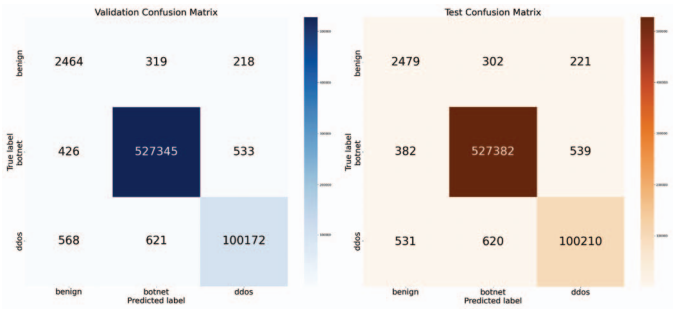

Fig. 6: RF Model Confusion Matrix - 20 Features

TABLE III: RF Model Results using 20 Features

Validation Set Results				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	71%	82%	76%	3,001
Botnet	100%	100%	100%	528,304
DDoS	99%	99%	99%	101,361
Accuracy	-	-	99.58%	632,666
Macro Avg	90%	94%	92%	632,666
Weighted Avg	100%	100%	100%	632,666
Test Set Results				
Class	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	73%	83%	78%	3,002
Botnet	100%	100%	100%	528,303
DDoS	99%	99%	99%	101,361
Accuracy	-	-	99.59%	632,666
Macro Avg	91%	94%	92%	632,666
Weighted Avg	100%	100%	100%	632,666

faster training, lower memory usage, and simpler trees, making the RF model more interpretable. Despite the lower dimensionality, the model maintains a high performance, demonstrating that the 20 features capture the most relevant patterns to distinguish between classes, ultimately making the system more scalable and robust for real-world deployment.

C. Analysis and Interpretability

To improve the interpretability of our model behavior, we employed SHapley Additive Explanations (SHAP) to analyze misclassifications between *Botnet* and *Benign* classes for the C2 analysis. SHAP enabled us to quantify the contribution of features in the prediction and to trace the decision boundaries that led to false positives or false negatives. SHAP values are grounded in cooperative game theory and represent the average marginal contribution of a feature in all possible feature coalitions. For a given model f and input x, the SHAP value ϕ_i for feature i is defined as:

$$\phi_i(f,x) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S) \right] \tag{2}$$

where:

- F is the set of all input features,
- S is a subset of features not containing i,
- $f_S(x_S)$ is the model's output using features in subset S.

By analyzing SHAP values for misclassified instances (Figures 7 and 8), we identified features that consistently drove incorrect predictions. SHAP adds explainability, supporting transparent IoT traffic analysis. Key contributors to botnet–benign misclassification include dsMinPl, dsLowQuartilePl, minL7PktSz, and PyldEntropy. All are tied to payload and packet size variability. Similar patterns emerged when benign was misclassified as botnet, with dsMedianPl and PyldChRatio playing a role. These results underscore the model's sensitivity to subtle flow-level variations.

To ensure that flows labeled as Benign are truly Benign, and that no Botnet traffic is incorrectly included, we define three conditions to filter flows for further analysis. These are:
Condition 1: If a flow is identified as DDoS (true positive), no further action is required.
Condition 2: If a flow is identified as Botnet (true positive), no further action is required.
Condition 3: If a flow is identified as Benign, we require an 80% (or higher) confidence score to affirm it as benign.

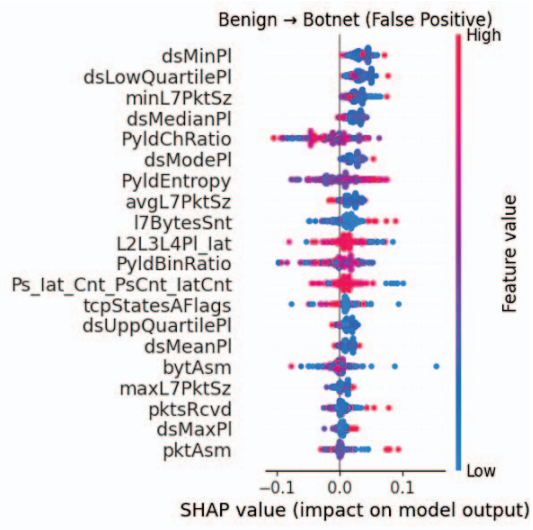

Fig. 7: Features leading to Benign misclassified as Botnet

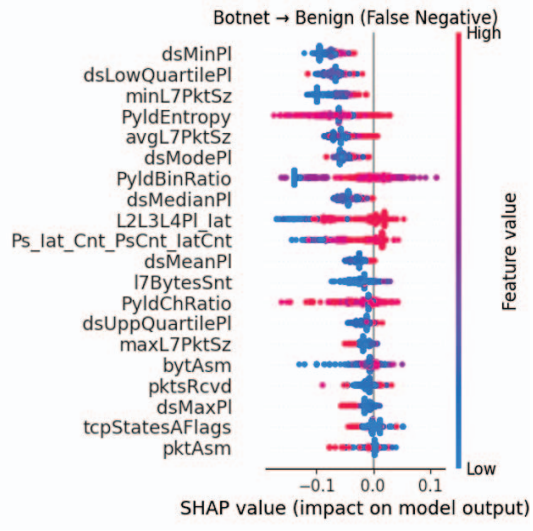

Fig. 8: Features leading to Botnet misclassified as Benign

In the RF model, each decision tree T_j casts a vote for a class label. The confidence score for a flow being classified as *Benign* is calculated as the proportion of trees that vote for the *Benign* label. If this score meets 80% (or higher), we affirm the flow as *Benign*, requiring no further inspection.

However, if a flow is labeled as Benign with less than 80% confidence score, it requires the "C2 Actor" of the Botnet Analysis Layer which is a lightweight DNS-Based Botnet Inference since there is no need for retraining.

TABLE IV: Breakdown of Flows for Validation - CIC IoT

Category	Flow Count
Benign misclassified as Botnet	621
Correctly classified Benign (Confidence $< 80\%$)	2,352
Botnet misclassified as Benign	808
Selected Benign Flows (621 + 2,352)	**2,973**
Selected Botnet Flows	**808**
Total Flows to be Validated	**3,781**

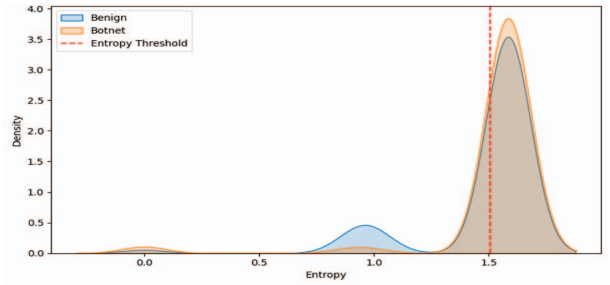

Fig. 9: Sub-domain Entropy

TABLE V: RF Test Results without DALC

Class	Precision	Recall	F1-Score	Support
Benign	9%	97%	16%	4,885
Botnet	57%	0%	1%	47,974
Accuracy	-	-	9.11%	52,859
Macro Avg	33%	49%	8.5%	52,859
Weighted Avg	52%	9%	2%	52,859

C2 Actor Feedback – Training Data (CIC IoT 2023)

- **C2 Detection in Selected Botnet:** 90.22% (729 / 808). Only 79 botnet flows were not indicated to include malicious C2.
- **C2 Detection in Selected Benign:** 8.49% (321 / 3,781) flagged; 3,460 flows correctly retained, recovering low-confidence misclassifications.
- These results show the validator uncovers hidden C2 via DNS and boosts classification confidence.

D. Testing on Different Network Data: Generalizability

To evaluate robustness and generalizability, we merged two external datasets: benign traffic from [15] and botnet traffic from [16], sampling 60-minute windows as in training. Both datasets contain traffic from real IoT devices, processed using the proposed extraction pipeline. These served as test data for our model trained on CIC IoT 2023. Performance was low (Table V), highlighting strong distributional drift and minimal overlap between training and test data. This illustrates a common challenge in cybersecurity: ML models often struggle to generalize across networks, and mitigation strategies frequently degrade or fall short in real-world deployments.

To address this problem, we propose a Botnet Validation Layer for generalizability, i.e. different network data. This analyzes segments to detect patterns and form a "trust set" for retraining, enhancing domain adaptability. Using a 5% similarity threshold between test and training data, it produces

TABLE VI: RF Test Results with DALC

Class	Precision	Recall	F1-Score	Support
Benign	77%	95%	85%	4,633
Botnet	99%	97%	98%	45,584
Accuracy	-	-	97%	50,217
Macro Avg	88%	96%	92%	50,217
Weighted Avg	97%	97%	97%	50,217

the DALC [1], a meta-learning algorithm that adapts to distributional drift without labels.

TABLE VII: Breakdown of Flows for Validation - Test Data

Category	Flow Count
Benign misclassified as Botnet	230
Correctly classified Benign (Confidence $< 80\%$)	309
Botnet misclassified as Benign	1,343
Selected Benign Flows (230 + 309)	**539**
Selected Botnet Flows	**1,343**
Total Flows to be Processed by C2 Actor	**1,882**

C2 Actor Detection Insights - Test Data

- Only **11.24%** of the selected botnet flows exhibited C2 presence, indicating challenges in DNS-based identification within this dataset.
- Among selected benign flows, just **3%** were flagged, confirming high specificity and reinforcing that most benign flows are correctly classified.

VI. RELATED WORK

DNS-Based Botnet Detection: DNS analysis has proven to be a valuable signal in IoT botnet detection. Li et al. [25] proposed an approach leveraging domain-based features such as meaningful Length Ratio, Query Type, and behavioral similarity to detect malicious domains, achieving 95.55% accuracy using classifiers like AdaBoost, Bagging, Naive Bayes, and KNN. Similarly, Alieyan et al. [3] presented 'the Gunner system', a DNS behavior-based approach to improve detection accuracy. However, Holz et al. [26] cautioned against overreliance on low TTL as a signal, noting its prevalence in benign CDN behavior.

IoT DNS Behavior Analysis: Fan et al. [5] analyzed DNS behavior patterns to detect botnets in IoT environments, while Hussain et al. [27] and M.A. et al. [28] explored hybrid and deep learning methods for feature-driven detection. Regis Anne et al. [29] benchmarked traditional ML and deep learning models on the Aposemat IoT-23 dataset, finding GRU models most effective (99.89% accuracy), combining speed and performance. These works demonstrate strong detection capabilities but often overlook the C2 subcomponent in botnet architecture.

Command and Control (C2) Detection in IoT: Research into C2 detection is relatively scarce yet growing. Gopal et al. [30] employed multi-attention recurrent neural networks (MARNNs) to recognize cyclical C2 traffic with 98.5% accuracy. Reynvoet et al. [31] used labeled traffic to differentiate

C2/non-C2 flows, achieving a 0.47% false positive rate. Off-path strategies like C2Miner [32] explored grammar-based clustering of C2 communications in IoT malware binaries, achieving an F1-Score of 86%. Issues of generalizability were not discussed but beyond detection, some studies focus on securing or countering C2 behavior. The Secure Remote Update Protocol (SRUP) [7] uses TLS-encrypted MQTT to secure C2 messages to IoT devices. Yamaguchi [33] and later CCBotnet [34] introduced Botnet Defense Systems (BDS) employing white-hat bots controlled via basic C2 strategies to disrupt malicious networks.

VII. CONCLUSION AND FUTURE WORK

This work introduced a Botnet Validation Layer that uses flow metadata with DNS-based C2 validation. Using only 20 features and the DALC, the framework adapts to data from new and different network while reducing reliance on labels. It achieves 99.7% F1-Score and 100% botnet recall, and in cross-dataset tests, the Validator Layer recovers 5% of "Trust" flows to increase F1-Scores. DNS-based analysis detects 90.2% of malicious C2 communications with low false positives. The lightweight Random Forest model enables edge deployment without sacrificing detection fidelity. While our approach demonstrates strong results, there are notable limitations. The DALC algorithm may struggle with non-IoT traffic, DNS-based C2 detection may miss unseen domains, requiring retraining. Future work will focus on unsupervised adaptation, and deeper study of P2P botnets.

ACKNOWLEDGEMENT

This research is supported by the Mitacs funding program. The research is conducted at Dalhousie NIMS Lab.

REFERENCES

[1] Gupta, B. B., Tewari, A. (2020). Evolution of internet of things (IoT). A beginner's guide to internet of things security, 1-9.

[2] L. Tong, F. Wang and S. Chen, "DGA-based Botnets Detection based on Vertical and Horizontal Analysis of DNS Behavior," 2023 9th Annual International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, China, 2023, pp. 131-136.

[3] K. Alieyan, M. Anbar, A. Almomani, R. Abdullah and M. Alauthman, "Botnets Detecting Attack Based on DNS Features," 2018 International Arab Conference on Information Technology (ACIT), Werdanye, Lebanon, 2018, pp. 1-4.

[4] Agung Udiyono, Charles Lim, and Lukas. 2020. Botnet Detection Using DNS and HTTP Traffic Analysis. In Proceedings of the 2020 International Conference on Engineering and Information Technology for Sustainable Industry (ICONETSI '20). Association for Computing Machinery, New York, NY, USA, Article 28, 1–6.

[5] C. -I. Fan, C. -H. Shie, C. -M. Hsu, T. Ban, T. Morikawa and T. Takahashi, "IoT Botnet Detection Based on the Behaviors of DNS Queries," 2022 IEEE Conference on Dependable and Secure Computing (DSC), Edinburgh, United Kingdom, 2022, pp. 1-7

[6] SpinDance. (2024). Rethinking IoT Command and Control: Beyond AWS Shadows.

[7] A. J. Poulter, S. J. Johnston and S. J. Cox, "pySRUP – Simplifying Secure Communications for Command Control in the Internet of Things," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 273-277.

[8] Othman, T. S., Koy, K. R., Abdullah, S. M. (2023). Intrusion Detection Systems for IoT Attack Detection and Identification Using Intelligent Techniques. Networks, 5, 6.

[9] A. Remillano II and J. Molina, "New Mirai variant expands exploits CVE-2020-1017," Trend Micro Research Blog, Oct. 2020.

[10] J. A. Adjei, N. Zincir-Heywood, B. Nandy, and N. Seddigh, "Can flow metadata-based signatures generalize for identifying attacks on IoT devices?" in Proc. IEEE/IFIP Netw. Operations and Management Symp. (NOMS), 2025, pp. 1–7.

[11] Kahraman Kostas, Rabia Yasa Kostas, Mike Just, and Michael A. Lones, "GMID: Generalizable Models for IoT Device Identification". 2024.

[12] G. -P. Fernando, A. -A. H. Brayan, A. M. Florina, C. -B. Liliana, A. -M. Héctor-Gabriel and T. -S. Reinel, "Enhancing Intrusion Detection in IoT Communications Through ML Model Generalization With a New Dataset (IDSAI)," in IEEE Access, vol. 11, pp. 70542-70559, 2023.

[13] S. Rajendran and Z. Sun, "RF Impairment Model-Based IoT Physical-Layer Identification for Enhanced Domain Generalization," in IEEE Transactions on Information Forensics and Security, vol. 17, pp. 1285-1299, 2022.

[14] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A. A. Ghorbani. "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensor (2023)

[15] N. Bastian, D. Bierbrauer, M. McKenzie, E Nack, December 29, 2023, " ACI IoT Network Traffic Dataset 2023", IEEE Dataport.

[16] Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Ho Park, Huy Kang Kim, "IoT network intrusion dataset", IEEE Dataport, September 27, 2019.

[17] "Tranalyzer: Lightweight flow generator," 2022, https://tranalyzer.com/

[18] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, (2017) DDoS in the IoT: Mirai and other botnets. Computer. 50. 80-84. 10.1109/MC.2017.201.

[19] J. Adjei, N. Z. Heywood, B. Nandy and N. Seddigh, "IoT Device and State Identification based on Usage Patterns", CNSM 2024-2024 IEEE, Prague, , Czech Republic

[20] J. Adjei, N. Z. Heywood, B. Nandy and N. Seddigh, "Identifying IoT Devices: A Machine Learning Analysis Using Traffic Flow Metadata," NOMS 2024-2024 IEEE Network Operations and Management Symposium, Seoul, Korea, Republic of, 2024, pp. 1-7.

[21] M. Feily, A. Shahrestani, and S. Ramadass, "A Survey of Botnet and Botnet Detection," Third International Conference on Emerging Security Information, Systems and Technologies, 2009, pp. 268–273.

[22] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting malware infection through IDS-driven dialog correlation," in *Proc. 16th USENIX Security Symposium*, 2007, pp. 167–182.

[23] S. Yamaguchi, "A Basic Command and Control Strategy in Botnet Defense System," 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2021, pp. 1-5.

[24] D. Panda, N. Padhy and K. Sharma, "Strengthening IoT Resilience: A Study on Backdoor Malware and DNS Spoofing Detection Methods," 2025 International Conference on Emerging Systems and Intelligent Computing (ESIC), Bhubaneswar, India, 2025, pp. 795-800.

[25] W. Li, J. Jin, and J. Lee, "Analysis of botnet domain names for IoT cybersecurity," *IEEE Access*, vol. 7, pp. 94658–94665, 2019.

[26] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proc. 15th Network and Distributed System Security Symposium (NDSS)*, 2008.

[27] F. Hussain *et al.*, "A two-fold machine learning approach to prevent and detect IoT botnet attacks, vol. 9, pp. 163412–163430, 2021.

[28] M. A., S. K. P. J., K. L., K. D., and S. R., "An optimized hybrid deep learning framework for monitoring botnet attacks in IoT networks," in *Int. Conf. IoT Based Control Networks and Intelligent Systems (ICICNIS)*, 2024, pp. 487–492.

[29] R. Anne, S. Sivakumar, and V. S. V. Arvind, "Detection of IoT botnets using machine and deep learning techniques: A case study on Aposemat IoT-23 dataset," *2023*.

[30] P. Gopal, V. Selvamani, and P. Sivakumar, 'A multi-attention recurrent neural network for detecting command and control traffic in IoT botnets, *2021*.

[31] B. Reynvoet, N. Van Den Meerssche, and T. Van Cutsem, "On detecting command and control communication in IoT network traffic," in *Proc. 2022 Int. Conf. Network and Service Management (CNSM)*, 2022.

[32] M. Xu, Y. Wang, Z. Li, and X. Yang, "C2Miner: Discovering live IoT C2 servers from discarded malware binaries," ICC, 2023, pp. 1–6.

[33] Y. Yamaguchi, "Botnet defense system with white-hat worms: An agent-oriented Petri net approach," *2018*.

[34] T. Morikawa and T. Takahashi, "A basic command and control strategy in botnet defense system," *2020*.