

RESEARCH ARTICLE



Leveraging AI and Network Forensics for Smart Botnet Detection

Shivanna Doddegowda^{1,2} and Pooja Mysore Raghupathi^{1,*}

¹Department of Computer Science and Engineering, Vidyavardhaka College of Engineering, India

²Department of Computer Science and Engineering, Visvesvaraya Technological University, India

Abstract: Nowadays, ensuring security for computers, networking devices and handheld devices has become a major concern, as numerous security threats are evolving rapidly and creating serious security issues. Botnet belongs to one such threat that is creating security concerns. Botnet is a collection of hijacked devices managed by a botmaster that are used to launch massive cyber threats. As cybercriminals aim to target users' computer networks, they have grown stealthier and more adaptive, often slipping past conventional security systems undetected. Devices that get connected to a botnet consist of systems, handheld devices like mobiles, and Internet of Things devices. The proposed approach introduces a hybrid behavioral–temporal framework for detecting bots that are compromised through live traffic monitoring. The traditional approaches are more dependent on publicly available datasets, whereas the proposed framework, instead of relying on such methods, utilizes live or real-time traffic, which enables to find the bots in real operational conditions. This framework utilizes multidimensional behavioral analysis to find the various activities associated with the botnet. By integrating real-time anomaly detection, it helps in finding both rapid and stealthy low-rate infectious activities of the botnet. Nevertheless, the framework eliminates the need for deep packet inspection, which is more suitable for scalable and even encrypted traffic networks. When this framework is compared to conventional methods, it achieves high precision, recall, and F1-score values while maintaining low false-positive rates, making it a highly effective tool for real-time threat detection.

Keywords: botmaster, DDoS, spam, jNetPcap, fuzzy

1. Introduction

“Botnet” combines the terms “robot” and “network.” Once a cybercriminal gains control of a collection of compromised devices, they can remotely command each device to perform coordinated actions, including malicious and illegal ones like distributed denial-of-service (DDoS) attacks. The security vulnerabilities of the infected system are taken advantage of, which leads to a breach of security and a shift in control of the devices to third parties. With the increasing digitization of daily activities, such as banking, shopping, education, and investments, and their shift to the online space, has led to a rise in malicious cyber activities, ranging from financial theft to identity fraud [1]. These use the size of the botnet to send a flood of requests or payloads to a target server, overloading it and preventing access for legitimate users [2].

Botnets pose numerous critical threats to the digital landscape and even in the Internet of Things (IoT) space, most notably distributed DDoS, spamming, information theft, and many more [3]. DDoS attacks are attacks that target networks and services by using several compromised computers, or “bots” [4].

The communication channel known as Command and Control (C2) is typically set up for malicious activity using remote shells or backdoors [5].

DDoS attacks are intended to prevent legitimate users from using a service for a certain amount of time [6]. Botnets are groups of compromised computers that carry out DDoS attacks [7]. Hackers exploit botnet attacks to take advantage of newly discovered vulnerabilities, using them to carry out intensive scraping, distributed denial-of-service (DDoS) attacks, and other large-scale cybercrimes [8]. Spamming is another behavior of botnets, where infected computers are used for sending spam emails. Botnets are especially well-suited for spam attacks, since infected devices carry residential IP addresses and their numbers keep growing as new devices get infected [9]. Like receiving unsolicited emails in the mail, spam is used to promote a service or product. Detecting botnets—and particularly telling apart normal traffic, non-spam botnets, and spam botnets—remains a critical challenge in cybersecurity [10]. With spam making up around 60% of all emails, it is one of the biggest risks to the Internet. The primary objective of spam botmasters is to obfuscate their identities to evade tracking and attribution.

It is very difficult to identify whether the system is affected by the bot or not. At present, there are many AI and machine learning (ML) algorithms that detect the botnet based on only the behavior they perform during the attacks [11].

*Corresponding author: Pooja Mysore Raghupathi, Department of Computer Science and Engineering, Vidyavardhaka College of Engineering, India. Email: pooja.mr@vvce.ac.in

Decentralized peer-to-peer (P2P) botnets pose a significant challenge to cybersecurity, largely because of their ability to adapt and evade standard mitigation techniques. Because of their far-reaching effects, it's essential to develop approaches that reduce botnets' impact and strengthen security measures against them [12, 13].

The primary intent of anomaly-based botnet identification is to look for abnormal patterns in the host and also in the network behavior, such as increased traffic volume, elevated latency, strange system behavior, etc. [14]. To establish reliable, accurate botnet detection models, anomaly detection has increasingly been paired with ML methods like Random Forest (RF) and support vector machine (SVM) in recent years [6, 15].

To detect bots and intrusion detection systems (IDSs), graph-based representations of network traffic logs and graph-based analytics were used [16, 17].

Every botnet detection technique [18, 19] has advantages and disadvantages, and frequently, a combination of these techniques is applied to increase botnet detection's precision and efficacy.

Network forensics and the log examination of the hosts are important sources for identifying botnet activities; this makes network forensics valuable for detecting malicious communication patterns [20]. Ali et al. [20] demonstrated ML techniques that can effectively learn complex botnet attack patterns from network traffic and help in detecting automated bots in IoT environments.

The methodology used here to detect botnet infections is based on real-time behavior, specifically analyzing traffic patterns at both the transport and application layers. At the network layer, only for well-known protocols will the data be captured like Simple Mail Transfer Protocol (SMTP), POP, Domain Name System (DNS), and so on by looking at the particular protocol's port. As understanding of network security and the importance of privacy protection grows, the use of encrypted DNS technologies (such as DNS over HTTPS, DNS over Transport Layer Security (TLS), and DNS over Quick UDP Internet Connections (QUIC)) has become widespread. By locking down DNS requests, these tools stop outsiders from snooping on or messing with web traffic, giving everyday users their privacy back. Nevertheless, encrypted DNS makes network traffic analysis more challenging, especially in recognizing harmful traffic such as botnets [21]. At the transport layer, flooding will be checked for both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols. Even some protocols like Internet Control Message Protocol (ICMP) at the network layer will be checked if the system is involved in any kind of attacks sponsored by the Botmaster.

This study examines the network traffic in real time and also closely watches the traffic volume. If any gradual or abrupt spike shows up in incoming or outgoing traffic, the fuzzy concept is used to forecast system behavior and flag the presence of a botnet [22]. Fuzzy logic is a generalized, many-valued logical framework where the truth value of a proposition is treated as a continuous variable within the closed interval $[0,1]$, instead of being decided as strictly binary states. Fuzzy logic was formally introduced to model vagueness and partial truth. It allows uncertain, imprecise, or linguistically expressed knowledge to be represented through membership functions and fuzzy sets. Given a universe of discourse X , a fuzzy set $A \subseteq X$ is characterized by a membership function $\mu_A(x): X \rightarrow [0,1]$, which quantifies the degree to which an element belongs to the set. This mathematical approach gives us a solid baseline for making decisions that are unpredictable. In practice, intelligent decision systems make use of fuzzy inference to bring out key features from raw datasets, allowing them to map out the underlying dependencies between variables

[23]. Let $R \subseteq X \times Y$ denote a fuzzy relation between feature space X and decision space Y , represented by a membership function $\mu_R(x,y)$.

The massive increase of IoT devices in heavy industries has significantly increased the attack surface for enterprise networks. Industrial Internet of Things (IIoT) environments integrate heterogeneous devices with constrained computational resources, dynamic topology, and distributed communication protocols, thereby increasing susceptibility to large-scale botnet infiltration [24].

1.1. Contributions of the proposed work

The main contributions of this work are summarized as follows:

- 1) Temporal Deviation Analysis:
This temporal mechanism relies on autocorrelation indicators to capture behavioral changes or deviations in network activity and helps to catch bots in their initial stages.
- 2) Fuzzy-Based Multi-Indicator Fusion:
A fuzzy inference framework combines multiple anomaly indicators into a unified anomaly score. This allows the system to make robust decisions under uncertainty and also minimizes false positives.
- 3) Lightweight Flow-Level Detection Model:
A scalable behavioral detection method was developed, built on aggregated network-forensic traffic features. It also eliminates the need for deep packet inspection (DPI) while still keeping detection efficiency high.

This work focuses on detecting botnet-related anomalies such as scanning activity, flooding attacks, and abnormal communication patterns at the network flow level within conventional enterprise infrastructures. This framework is also applicable to IoT and IIoT environments.

This paper provides contributions in two core paradigms, in order to distinguish between the theoretical design and its physical application. The conceptual framework establishment is presented in Sections 3.1–3.7, and the lag-1 autocorrelation and multi-feature fuzzy behavioral fusion are stated. On the contrary, Sections 3.8 and 4 show the experimental validity of this framework and even demonstrate the strength and operational scalability in real-time operational environments. The rest of this paper is organized as follows. Section 2 looks into the existing literature on botnet detection and anomaly-based network monitoring. Section 3 lays out the proposed framework in detail, which covers traffic aggregation, feature extraction, temporal deviation modeling, and fuzzy inference. Section 4 shows the findings of the framework against baseline techniques. The last section ends with the concluding remarks and also highlights the potential directions for future research.

2. Related Works

Conventional methods like DPI approaches are becoming obsolete, and they are being replaced by intelligent ML techniques.

A common approach to find the activities of the botnet is through linking alerts across separate IDSs. Administrators can stay ahead of active botnet campaigns by integrating this data to provide cross-network awareness [25]. This will find botnet attacks through a series of events; otherwise, it will not work for every single process, and also the installation of sensors at each network

would be a little difficult. The Mirai botnet frequently targets resource-constrained IoT devices. While conventional ML techniques are applied to detect these IoT botnets, their performance and classification accuracy degrade significantly when scaling to large datasets [26, 27]. IDS was used to prevent botnet attacks using ML methods. The IDS's ensemble voting classifier combines several ML techniques to increase accuracy and adaptability as its fundamental component. This technique provided the tool that is required to identify emerging threats in addition to increasing its capacity to recognize well-known Botnet patterns [28]. Overall, the accuracy has been improved by 3%, but the method required to collect a huge amount of data and then feed it into the algorithms is an expensive and tedious task.

S. Haq and Y. Singh have proposed a model that utilizes ML, clustering, and classification approaches. This approach shows the performance and outcomes of both clustering and classification on one side and the hybrid approach on the other. The key concepts focused on in this work are botnet's human engagement and unexpected or unknown traffic [29].

A neuro-fuzzy model was presented by Pradeepthi and Kannan [30] in order to increase accuracy and reduce false positives. The idea is put into practice using a large number of virtual computers on the Eucalyptus cloud. However, no testing or experimentation was done on the technology in any other actual cloud environments. Using botnet simulation tools, the functioning model has only been evaluated and analyzed in a simulation environment.

Javier, Victor, Eduardo, and Enrique used ML on high-bandwidth networks to identify the botnet. TCP- and UDP-based communications between computers with bandwidths up to 10 Gbps are used for network traffic analysis [12]. But this only works in an environment where the bandwidth is huge like more than 10 Gbps and does not work on networks where bandwidth is usually around 1 Gbps.

The Botnet Attack Detection and Defense (BADD) framework was presented by Prasanth [31]. It uses high-dimensional network properties to detect IoT botnets using a supervised Convolutional Neural Network (CNN) trained on the Bot-IoT dataset. High classification performance is shown by the framework in meticulously controlled experimental conditions. But this model heavily depends on spatial feature extraction and has mostly been tested on static benchmark datasets; it may not be so good for dynamic network environments. It also depends on packet-level features, which limits how well it works on encrypted traffic where payload information simply isn't available.

Oskera [5] framed a frequency-domain approach based on the Lomb-Scargle periodogram to detect periodic C2 communications. Experimental evaluation on the CESNET-CC25 dataset showed it could identify recurring beaconing behavior without needing host-level information. Since the approach focuses on periodic communication patterns, its performance may be reduced for randomized communication intervals produced by botnets, particularly when timing jitter is introduced or when short-lived C2 sessions are used.

Hostiadi and Ahmad [32] used sliding-window optimization to look at the temporal traffic patterns using the CTU-13 dataset. The result found that adaptive windowing improves temporal correlation and helps in the identification of different stages of botnet activity. However, the approach relies more on preprocessed traffic and predefined window sizes, which adds preprocessing overhead and makes it less suited to real-time environments.

The proposed approach uses a different approach that integrates behavioral features with temporal analysis to support continuous botnet detection in dynamic and real-time network

environments. The framework puts less effort to preprocess, and at the same time, it pulls together behavioral and temporal information. The model is better equipped to adapt to the continuously changing botnet behavioral patterns. Traditional deep learning methods tend to depend heavily on predefined, well-known datasets and complex payload-specific features. But this framework takes a lighter path, working with nothing more than minimal flow-level metadata. Also eliminates the need to look into the DPI approach. That makes the design scalable and reduces computational overhead, and it will also be effective in such environments where packet payloads cannot be accessed due to encryption. Frequency-domain methods mostly emphasize strict communication periodicity, whereas our system moves further and looks at bidirectional traffic symmetry, communication continuity, unusual port usage, traffic burst patterns, latency variations, and temporal coherence all at once. That's what keeps it effective even when adversaries deliberately shift their transmission intervals to dodge conventional timing-based detection.

The earlier studies were used to assess the performance with the use of offline benchmark datasets. This framework works for live network traffic captures in realistic operational settings. This makes it more relevant for practical deployment and strengthens its ability to generalize. The framework reinforces its usability for real-world intelligent cyber defense in corporate network settings.

3. Methodology

3.1. Problem formulation and research hypothesis

This work models botnet detection as an online temporal persistence estimation problem using payload-free, flow-level behavioral metadata extracted from live network streams. Both offline benchmark datasets and traditional DPI fail in live environments when the payload gets encrypted. While this framework assumes that botnet activities, such as C2 polling, scanning, and propagation, create predictable and repeating patterns, human-generated network traffic is generally more variable and exhibits less temporal consistency. The proposed method computes an online lag-1 autocorrelation analysis over contemporaneous network data to evaluate this automated persistence in a non-intrusive way. A statistically high autocorrelation coefficient is essentially a hallmark of automated communication—it's what lets the framework distinguish malicious behavior from the traffic patterns of everyday human activity.

3.2. Traffic capture and streaming architecture

The framework taps into live network traffic continuously through the jNetPcap library, using its native libpcap interface for lightweight packet capture. Instead of inspecting packet payloads, only flow-level metadata is extracted, so that it works even when communication is encrypted. For every observed network packet passing through the monitoring interface, the framework extracts a lightweight, fixed-size tuple consisting of the following metadata attributes:

- 1) Packet length (bytes)
- 2) Source IP address
- 3) Destination IP address
- 4) Source port number
- 5) Destination port number
- 6) Transport protocol (TCP, UDP, or ICMP)
- 7) Application protocol inferred via standardized port mapping

Table 1
Well Known protocols and their corresponding ports Under Surveillance

Protocol	Port	Operational Threat Mapping & Attack Vector
SMTP	25	Spam distribution, data exfiltration loops
FTP	20, 21	Automated payload dropping, credential abuse
DNS	53	C2 command tunneling, malicious domain fluxing
HTTP	80	Web-based bot profiling, scanning, flooding
HTTPS	443	Encrypted C2 communication, stealth beacons
IMAP	143	Remote control propagation, secondary abuse
POP3	110	Unauthorized information gathering
Telnet	23	Automated brute-force IoT propagation vectors

As traffic is generated, metadata comes in, which is immediately aggregated on a per-host basis using static IP association, building out dynamic temporal behavioral profiles for each active device on the enterprise network. The framework prioritizes the well-known protocols to flag suspicious activity. Table 1 provides a detailed description of these well-known protocols and the corresponding port mappings.

3.3. Continuous streaming sliding-window strategy

To support continuous monitoring, the framework employs a rolling sliding-window strategy instead of offline batch aggregation. The active observation window is updated every 5 minutes while preserving a 7-day behavioral history. At each update, obsolete observations are removed, newly captured traffic is incorporated, and the behavioral statistics are recalculated. Let W_t denote the active host evaluation window at time step t . The framework maintains a fixed historical observation duration T (e.g., a 7-day lookback horizon) and updates continuously at a fine-grained sliding interval Δt (e.g., every 5 min):

$$W_{t-\Delta t} \rightarrow W_t \rightarrow W_{t+\Delta t} \quad (1)$$

With every sliding step, old metadata drops out of the buffer automatically, and newly generated traffic statistics are incorporated into the active host profile.

3.4. Behavioral feature extraction

The framework captures seven distinct, lightweight flow-level behavioral features, chosen to reflect communication persistence, traffic asymmetry, temporal regularity, and the kind of anomalous transmission behavior typical of botnet-controlled hosts. Since it relies only on metadata, it can be applied to any type of network.

Figure 1 illustrates the architecture of the proposed system, which depicts the flow of the anomaly detection system.

3.4.1. Outgoing traffic volume

$x_t^{(1)}$ = Outgoing packets volume on the monitored ports of a host during W_t

When a host is infected by bots and used to send spam emails, join DDoS attacks, or spread malicious commands quickly, one of the clearest signs is a consistently high level of outbound communication traffic.

3.4.2. Incoming traffic volume

$x_t^{(2)}$ = incoming packets volume on the monitored ports of a host during W_t

Unusual inbound communication patterns often suggest C2 reactions, scanning responses, organized P2P synchronization traffic, or unauthorized remote interactive connections.

3.4.3. Outgoing-to-incoming traffic ratio

Structural traffic asymmetry is quantified directly at each sliding step as follows:

$$x_t^{(3)} = \frac{x_t^{(1)}}{x_t^{(2)}} \quad (2)$$

A significant disparity between outgoing and incoming traffic patterns is a crucial structural sign of flooding events, asymmetric scanning cycles, or automated noninteractive harmful data transfers.

3.4.4. Mean inter-connection delay

Let Δt indicate the temporal interval between consecutive network connections that are initiated by the host inside the rolling observation window W_t . The mean inter-connection delay is formally defined as:

$$x_t^{(4)} = \frac{1}{M_t - 1} \sum_{i=1}^{M_t - 1} \Delta t_i \quad (3)$$

where M_t indicates the total number of connections observed within the window.

3.4.5. Median inter-connection delay

The framework calculates the median inter-connection delay as follows:

$$x_t^{(5)} = \text{median}(\Delta t_1, \dots, \Delta t_{M_t}) \quad (4)$$

3.4.6. ICMP request rate

$x_t^{(6)}$ = Total no of ICMP requests carried out during W_t

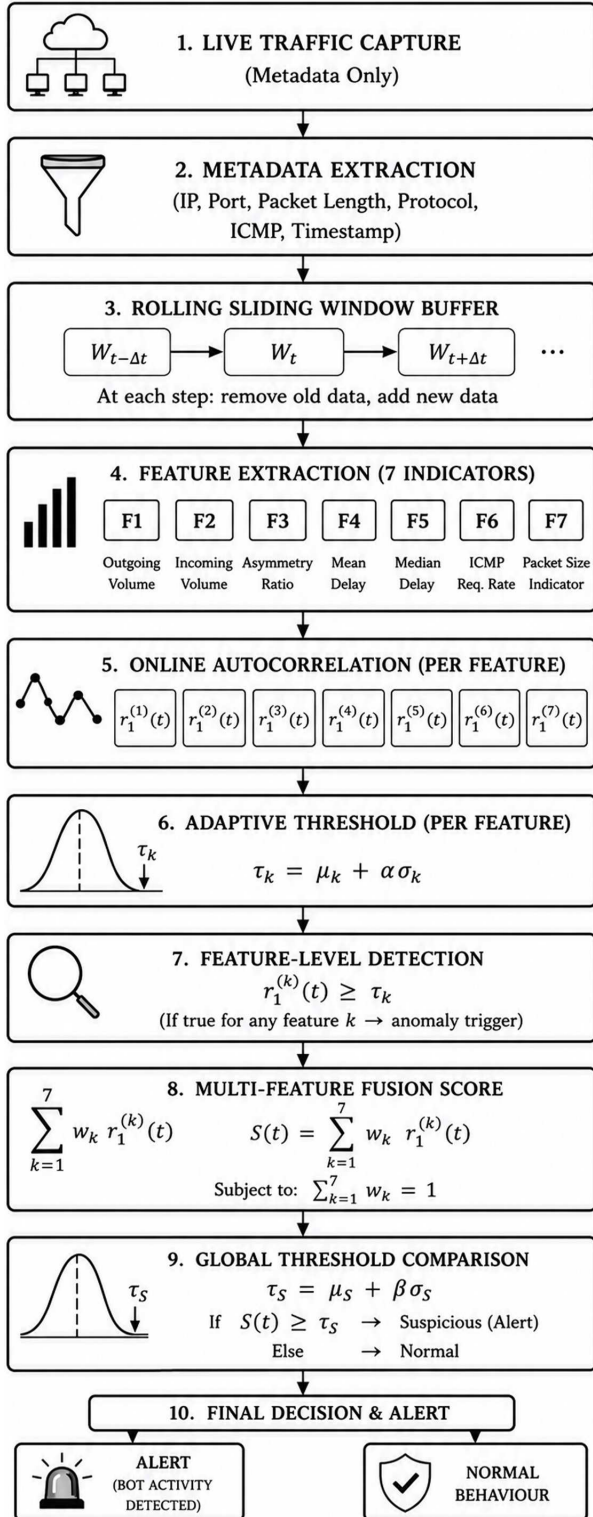
3.4.7. Packet size indicator

$x_t^{(7)}$ = This metric calculates the average byte volume of packets observed inside the current observation window W_t

3.4.8. Behavioral significance of extracted features

All the selected features create a multifaceted temporal representation of network activity. Working across these multiple dimensions, the framework can detect both fast, high-volume attacks and subtler, low-rate C2 communication behaviors.

Figure 1
Pipeline of the proposed hybrid behavioral-temporal framework



3.5. Online autocorrelation-based behavioral persistence estimation

To find the first-lag autocorrelation coefficient is computed dynamically over an ongoing historical lookback duration that has lasted H consecutive rolling-window steps. For every

extracted behavioral characteristic k , a real-time discrete sequence is persistently upheld within a sliding statistical buffer:

$$X_t^{(k)} = \{x_{t-H+1}^{(k)}, x_{t-H+2}^{(k)}, \dots, x_t^{(k)}\} \quad (5)$$

The instant temporal persistence estimator $r_1^{(k)(t)}$ at the current sliding step t is calculated mathematically over the active historical lookback horizon as follows:

$$r_1^{(k)}(t) = \frac{\sum_{\tau=t-H+1}^{t-1} (x_{\tau}^{(k)} - \bar{x}_{1,t}^{(k)}) (x_{\tau+1}^{(k)} - \bar{x}_{2,t}^{(k)})}{\sqrt{\left[\sum_{\tau=t-H+1}^{t-1} (x_{\tau}^{(k)} - \bar{x}_{1,t}^{(k)})^2 \right] \left[\sum_{\tau=t-H+1}^{t-1} (x_{\tau+1}^{(k)} - \bar{x}_{2,t}^{(k)})^2 \right]}} \quad (6)$$

where $\bar{x}_{1,t}^{(k)}$ and $\bar{x}_{2,t}^{(k)}$ represent the localized running sample means calculated exclusively over the sliding lookback horizon elements:

$$\begin{aligned} \bar{x}_{1,t}^{(k)} &= (1/(H-1)) \sum_{\tau=t-H+1}^{t-1} x_{\tau}^{(k)} \\ \bar{x}_{2,t}^{(k)} &= (1/(H-1)) \sum_{\tau=t-H+1}^{t-1} x_{\tau+1}^{(k)} \end{aligned} \quad (7)$$

By keeping statistical aggregation confined to the rolling lookback horizon H , the framework estimates temporal persistence online without having to repeatedly process the whole traffic history. The proposed framework has a computational complexity of $(O(H))$ per update interval. Because the computational cost scales linearly with the lookback window rather than the full traffic history. The framework stays well-suited for real-time analysis of streaming network traffic without increasing the overhead.

3.6. Interpretation of temporal persistence

The streaming autocorrelation coefficient matches irregular, non-deterministic human network dynamics by satisfying the conventional boundary constraint: $0 \leq |r_1^{(k)}(t)| \leq 1$. Values that are very near to zero suggest a total lack of temporal dependency, whereas values that are close to one indicate a strong serial reliance and also the existence of repetitive, automated communication patterns that are typical of botnet control modules.

3.7. Statistical detection and multi-feature behavioral fusion

3.7.1. Baseline behavior modeling

The framework establishes a statistical behavioral baseline from validated benign traffic observations and shows a way to separate automated malicious activity from normal network operations.

The mean autocorrelation (μ_k) and standard deviation (σ_k) for each monitored feature k are calculated as follows:

$$\begin{aligned} \mu_k &= (1/M) \sum_{i=1}^M r_{1,i}^{(k)} \\ \sigma_k &= \sqrt{[(1/(M-1)) \sum_{i=1}^M (r_{1,i}^{(k)} - \mu_k)^2]} \end{aligned} \quad (8)$$

where $r_{1,i}^{(k)}$ is the lag-1 autocorrelation value of feature k for observation i and M indicates the number of benign observations used for baseline calculation. These characteristics serve as the reference model for anomaly identification and describe the anticipated temporal stability of acceptable communication patterns.

3.7.2. Adaptive threshold estimation

An adaptive anomaly threshold τ_k is calculated dynamically for each individual feature track as follows:

$$\tau_k = \mu_k + \alpha \sigma_k \quad (9)$$

where μ_k and σ_k represent the running localized mean and standard deviation of feature k and α denotes the empirical sensitivity parameter controlling the boundary width of the normal profile. To statistically justify the selection of $\alpha = 2.0$, a parametric validation sweep was executed across $\alpha \in [1.0, 3.0]$ using the validation subset (D_{val}). Lower bounds ($\alpha \leq 1.5$) increased tracking sensitivity but yielded an unacceptable false positive rate (FPR > 4.2%) that induces live alert fatigue, whereas conservative upper bounds ($\alpha \geq 2.5$) overly constricted the detection envelope and suppressed the true positive detection rate down to 74.1%. The configuration of $\alpha = 2.0$ was selected because it mathematically minimized the geometric distance to the ideal receiver operating point (0, 1), establishing an optimized, defensible equilibrium that yields a robust 92.1% true positive detection rate alongside a tightly bound 1.5% false positive rate.

3.7.3. Feature-level behavioral detection

An individual host is flagged as behaviorally anomalous at time step t if its instantaneous autocorrelation satisfies the following condition:

$$r_1^{(k)} \geq \tau_k \quad (10)$$

for at least one of the monitored behavioral features k . This early signal suggests a monitored traffic feature has been behaving anomalously long enough to drift from the expected temporal pattern of legitimate traffic. That drift can indicate a potential automated C2 activity.

3.7.4. Composite multi-feature behavioral fusion

The aggregation of all seven behavioral autocorrelation tracks into a singular, unified composite anomaly score:

$$S(t) = \sum_{k=1}^7 w_k r_1^{(k)}(t) \quad (11)$$

where w_k represents the explicit weight assigned to feature track k , which must satisfy a strict normalization constraint.

$$\sum_{k=1}^7 w_k = 1 \quad (12)$$

The fusion of these multidimensional behavioral correlation engines created by the framework, which is intended to target specialized attacker evasion strategies, makes the approach stronger to find Bots. The weights are mapped to address distinct attack vectors and evasion mechanisms across three critical operational layers:

- 1) Volumetric and Asymmetric Attack Resilience: This layer tracks the outgoing volume and incoming volume and traffic asymmetry to uncover bulk data exfiltration, active scanning loops, and DDoS attacks.
- 2) Jitter-Evasive Timing Protection: This layer depends more on mean delay and the outlier-resilient median delay. Even when a sophisticated attacker injects random timing jitter to mask the periodic signature that frequency-domain tools usually catch, these temporal stability metrics will shift in response. That shift gives it away—the underlying automated behavior still

shows up through simultaneous surges in the related structural metrics ($x_t^{(3)}$ or $x_t^{(7)}$).

- 3) Multi-Layer Protocol Signaling: This layer evaluates using ICMP requests and structural variations in the packet size indicator to detect stealthy low-rate discovery sweeps.

A host is designated as a bot infected when the consolidated fusion score is greater than or equal to the global behavioral threshold: $S(t) \geq \tau_S$.

3.7.5. Multi-layer detection interpretation

This framework has a two-tier approach to find the bots. The feature tier checks for any significant deviations from the usual network behavior. In the second tier, all seven temporal indicators are jointly analyzed to generate the final detection decision. This hierarchical approach reduces false alarms by distinguishing transient fluctuations in legitimate traffic from persistent behavioral anomalies associated with botnet activity. By integrating feature-level analysis with multi-feature temporal correlation, the framework improves its ability to discover both high-rate attacks and stealthy low-rate C2 communications that are not usually identified by individual-feature, static threshold approaches.

3.8. Experimental dataset generation and behavioral learning framework

To evaluate the proposed hybrid behavioral-temporal framework, network traffic data were collected using a controlled virtual environment that emulated an organizational network. The experimental setup consisted of three monitored systems $K = 3$, with each system observed for 8 h per day over a 1-week period.

3.8.1. Traffic generation and feature extraction

The data collection process consisted of two distinct operational phases:

- 1) Baseline Phase (Benign Profile): In this phase, the systems operate under normal conditions with activity limited to legitimate workflows such as HTTP/HTTPS, DNS, SSH, SMTP, file transfers, and system updates, without any intended injections of malicious activities to establish the normal baseline profile (B_{normal}).
- 2) Attack Phase (Malicious Profile): Controlled botnet activities, which consist of C2 communication, automated port scanning, ICMP requests, and anomalous outbound traffic bursts, were generated in an isolated network segment to create the bot profile (B_{Bot}).

Live traffic was captured using the jNetPcap library for flow-level analysis. Packets were aggregated within successive 5-minute intervals (I_t) to build a seven-dimensional behavioral feature vector for each monitored host k :

$$x_t^{(k)} = [v_{out}, v_{in}, a_{asym}, \mu_{delay}, \tilde{x}_{delay}, r_{icmp}, s_{packet}]^T \quad (13)$$

Each interval profile was compared against a rolling 7-day historical window. As new observations became available, those falling outside the 7-day window were discarded, allowing the statistical baseline to adapt to legitimate changes in network behavior.

3.8.2. Subsampling and partitioning strategy

For each 5-minute interval, every monitored system generates 12 feature vectors per hour, resulting in 96 feature vectors

during each 8-hour observation session. For 7 days for each host, it produces $96 \times 3 \times 7 = 2016$ feature samples per phase. To keep the evaluation dataset (D) class-balanced, we retained a stratified subsample of roughly 1,000 benign samples (N_{benign}) and 1,000 bot samples (N_{Bot}). Stratifying jointly across host identity and observation day to preserve both temporal and inter-host diversity.

$$D = D_{\text{benign}} \cup D_{\text{bot}}, \quad \text{where } |D_{\text{benign}}| \approx |D_{\text{bot}}| \approx 1,000$$

The dataset D was split 80/10/10 via stratified sampling into disjoint training (D_{train}), validation (D_{val}), and testing (D_{test}) subsets:

$$D = D_{\text{train}} \cup D_{\text{val}} \cup D_{\text{test}}$$

such that $|D_{\text{train}}| = 0.80 \times |D|$, $|D_{\text{val}}| = 0.10 \times |D|$, $|D_{\text{test}}| = 0.10 \times |D|$ (14)

D_{train} was used for model construction, D_{val} for hyperparameter tuning, and the D_{test} was utilized exclusively for final evaluation.

To evaluate the discriminative capability of the proposed temporal features, three baseline methods were assessed using the test dataset D_{test} : static threshold detector, a One-Class Support Vector Machine (OC-SVM), and an RF classifier. Consistent with their learning paradigms, the threshold detector and the OC-SVM were trained exclusively on benign samples from the training dataset D_{train} , whereas the RF classifier was trained using the complete labeled training set. Model performance was evaluated using accuracy, precision, recall, F1-score, Receiver Operating Characteristic–Area Under the Curve (ROC-AUC), and confusion matrices. It is important to clarify that these machine learning models are used solely for diagnostic purposes, specifically to confirm that the extracted feature space is statistically separable. The actual real-time threat detection in the framework runs independently, driven entirely by the temporal persistence estimators and the fuzzy behavioral inference engine.

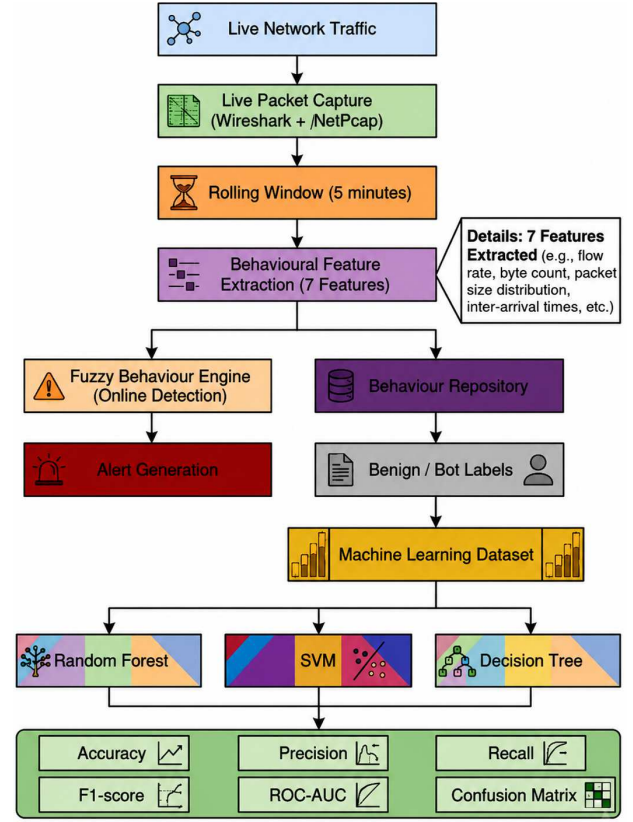
For full transparency, the underlying scripts will be made publicly available in an anonymized repository once the manuscript is published. This includes the traffic monitoring scripts, the jNetPcap feature extraction pipeline, and the validation configurations used in this study. Now the dataset is made available, and its location is shared at the end of the script.

Figure 2 illustrates the data processing architecture, wherein live network traffic is transformed into behavioral features, analyzed in real time via fuzzy logic, and subsequently utilized to construct the labeled dataset for machine learning evaluation.

4. Results and Discussion

The behavioral dataset used for evaluation was generated automatically through the proposed rolling-window monitoring framework described in Section 3.8. Every 5-minute interval generates a seven-dimensional behavioral feature vector that represents the temporal communication characteristics of the host being monitored. Then the labeled data is used to evaluate both the proposed fuzzy behavioral detection framework and a set of conventional ML classifiers, which gives us independent performance validation.

Figure 2
Feature extraction and validation architecture



4.1. Demonstration of limitations of static threshold-based detection

Before we proceed to validating the performance of the proposed method, this subsection illustrates the limitations of the conventional static threshold-based detection techniques, which are commonly used in rule-based and signature-driven IDSs. This analysis backs up the limitations discussed earlier and emphasizes the need for a temporal and adaptive detection technique.

To achieve this, a baseline detector was set up using fixed thresholds on common traffic metrics—packet rate, protocol-specific traffic volume, inbound-outbound ratio, and inter-arrival delay. Traffic was consolidated into weekly intervals using the same preprocessing steps discussed earlier, with an alert triggered whenever a feature exceeded its threshold based on historical averages.

The static threshold-based detector caught roughly 60% of botnet-related behaviors. Stealthy C2 communications and low-rate attacks remained undetected as their traffic stayed within what looked like normal ranges, so they slipped through undetected. At the same time, legitimate variations in user activity occasionally exceeded fixed thresholds, leading to a relatively high false-positive rate.

The inherent limitations of static threshold-based detection can be summarized as follows:

- 1) Inability to capture temporal behavioral dependencies.
- 2) Sensitivity to normal traffic variability.
- 3) Lack of contextual behavioral aggregation.
- 4) Poor robustness against low-rate and stealthy attacks.

This is what motivates the proposed framework, which uses autocorrelation-driven deviation analysis paired with fuzzy aggregation—which we evaluate in the subsections that follow.

Feature vectors extracted from the rolling-window framework were labeled normal (baseline phase) or bot (post-attack phase), providing a dataset for independent ML validation. Conventional classifiers were then trained on the extracted data to see how well the extracted temporal features separate the two classes. These results validate the effectiveness of the feature extraction procedure itself—they do not take the place of the suggested fuzzy inference engine, which remains the actual detection mechanism.

The quantitative performance results obtained across standard evaluation metrics are summarized in Table 2.

Table 2 shows that the framework proposed is outperforming the threshold-based detector and the ML classifiers for every feature. Static thresholding struggles the most, since it can't account for how behavior changes over time.

The proposed method instead tracks behavioral persistence using rolling lag-1 autocorrelation, which lets it pick up both fast-spreading botnet activity and quieter, sustained C2 behavior. It also doesn't depend on large labeled datasets like the supervised classifiers do, making it a better fit for environments where network behavior keeps shifting.

To mathematically verify the performance of the proposed framework, statistical tests were conducted to determine whether the observed improvement over the baseline methods was statistically significant. A paired two-tailed t-test was applied to the prediction results obtained from the held-out test set D_{test} , while McNemar's chi-squared test was used to compare the differences in classification errors. The proposed framework was compared with the strongest baseline, the supervised Random Forest (RF) classifier. The results showed a statistically significant difference between the two methods ($p < 0.01$). This provides additional evidence that the performance improvement is associated with the proposed use of localized temporal persistence in the fuzzy inference framework rather than being due only to variation in the test samples.

4.2. Discussion of detection performance

The results confirm that the system tracks behavioral persistence over time, instead of just relying on single traffic snapshots, to distinguish benign hosts from bot-infected ones. Four components make this work: rolling-window monitoring, autocorrelation-based temporal analysis, adaptive thresholding, and fuzzy inference. Long-term behavioral patterns are captured by the rolling window. It also smooths out the short-lived fluctuations expected from normal enterprise traffic, so an isolated burst or a one-off anomaly doesn't throw off the final decision.

Adaptive thresholding adjusts the decision boundary to reflect the network's real statistical behavior, which reduces the false positives that fixed thresholds typically generate—adding another layer of robustness. The fuzzy inference engine then weighs multiple behavioral indicators together, rather than leaning on any single feature. That makes it better at catching coordinated deviations tied to botnet activity. The ML validation results back this up. The extracted temporal features turned out to be strongly discriminative, reflected in the accuracy, precision, recall, F1-score, and ROC-AUC scores reported earlier. Overall, the proposed framework successfully distinguishes between malicious and benign traffic while remaining lightweight enough for real-time deployment.

4.3. Feature-level temporal persistence analysis

The lag-1 autocorrelation coefficient $r_1(k)(t)$ was continually calculated for each of the seven behavioral features, in order to verify the hypothesis of this study, which suggests that the infected hosts will show greater temporal persistence than the benign hosts. This computation runs entirely within the localized historical lookback period, $H = 7 \times 24 \times 12 = 2016$ intervals, inside the online sliding-window buffer. That keeps the tracking latency deterministic.

The explicit feature-level autocorrelation values and adaptive thresholds found for both benign and bot-infected host execution tracks are shown in Table 3. The autocorrelation coefficients obtained during the initial observation period were

Table 2
Performance evaluation across comparative detection paradigms

Method	Precision	Recall	F1-score	ROC-AUC
Static threshold	0.63	0.61	0.62	0.68
One-Class SVM	0.78	0.75	0.76	0.81
Random Forest classifier	0.84	0.82	0.83	0.88
Proposed method	0.92	0.90	0.91	0.94

Table 3
Statistical autocorrelation properties and adaptive anomalous thresholds (τ_k)

Feature track (k)	Adaptive threshold (τ_k)	Benign host value	Bot-infected host value
F1 – Outgoing Volume	0.31	0.20	0.42
F2 – Incoming Volume	0.27	0.18	0.37
F3 – Traffic Asymmetry	0.36	0.22	0.45
F4 – Mean Delay	0.27	0.19	0.40
F5 – Median Delay	0.30	0.21	0.39
F6 – ICMP Request Rate	0.24	0.17	0.34
F7 – Packet Size Indicator	0.32	0.23	0.41

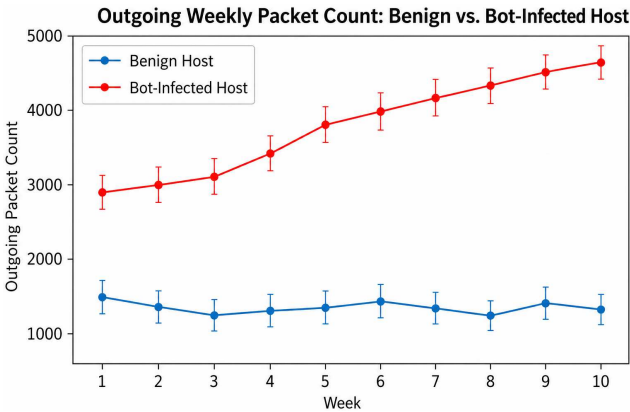
considered indicative of normal host behavior in order to establish a behavioral baseline. Newly calculated coefficients were compared against the baseline during the later monitoring periods. The gap between past and present behavior is used to measure behavioral drift and flag possible botnet activity. This technique enables the framework to continue operating effectively even while a host gradually becomes infected over time.

Table 3 reports representative average autocorrelation values obtained across the evaluation dataset. The results depict that there is a clear separation between benign and infected hosts across all monitored behavioral features. While benign hosts remain within the adaptive threshold range, infected hosts exceed the adaptive thresholds in many feature dimensions, which demonstrates that there is a strong temporal persistence associated with automated network activity.

Figure 3 illustrates the difference between the outgoing packets behavior during normal and infected hosts.

In the same way, the strong persistence observed in the mean and median delay features indicates the presence of regular communication patterns that are consistent with automated processes and differ from the more variable timing characteristics of legitimate user activity. The sustained rise in outgoing packet activity for the bot-infected host also illustrates the temporal persistence captured by the proposed autocorrelation framework. The framework put forth improves its capacity to differentiate automated communication patterns from typical normal network activity by capturing sustained behavioral regularity over time, unlike immediate threshold-based approaches.

Figure 3
Comparison of outgoing packet persistence between benign and bot-infected hosts across consecutive monitoring intervals



4.4. Validation of multidimensional behavioral monitoring

The framework integrates all seven complementary behavioral indicators to represent different characteristics of network traffic, which allows to tackle multiple botnet attack strategies at once and expose automated traces across distinct traffic patterns.

4.4.1. Volumetric and asymmetric attack detection

The features outgoing volume, incoming volume, and traffic asymmetry can effectively detect automated flooding activities, scanning activities, and any attempt at data exfiltration. Hosts that are infected by bots show an elevation in these features.

4.4.2. Jitter-evasive timing detection

Mean delay and median delay provide protection against timing-jitter evasion techniques. Even when attackers deliberately vary connection intervals to disturb periodicity-based detection, automated communication behavior can still be detected through concurrent increases in the persistence of structural traffic features.

4.4.3. Protocol-level signaling detection

The ICMP request rate and packet size indicators keep watch for low-rate reconnaissance activity, host discovery scans, and persistent C2 polling behavior. A consistent rise in either can signal compromised, malicious activity.

4.5. Composite behavioral fusion analysis

To reduce false positives caused by isolated traffic surges or transient user activity, the framework combines the autocorrelation values of all seven behavioral features into a single composite behavioral fusion score.

Using equal feature weights: $w_k = 1/7 = 0.143$

Equal weighting was adopted to avoid feature bias and to evaluate the intrinsic contribution of temporal persistence across all behavioral dimensions.

The following baseline matrix validations were used to evaluate the composite anomaly scores for target configurations:

1) Benign Host Model Evaluation:

$$S_{\text{benign}} = 0.143 * (0.20 + 0.18 + 0.22 + 0.19 + 0.21 + 0.17 + 0.23) = 0.20$$

2) Bot-Infected Host Model Evaluation:

$$S_{\text{infected}} = 0.143 * (0.42 + 0.37 + 0.45 + 0.40 + 0.39 + 0.34 + 0.41) = 0.40$$

The global behavioral decision threshold (τ_S) was derived using verified normal baseline observations:

$$\begin{aligned} \tau_S &= \mu_S + 2\sigma_S \\ \tau_S &= 0.20 + 2(0.04) \\ \tau_S &= 0.28 \end{aligned} \tag{14}$$

The computed composite anomaly scores were shown on a fuzzy behavioral interpretation scale to improve the visualization of the decision boundaries formed by the proposed fusion architecture. The regions of behavioral transition allow for a gradual distinction between legitimate and suspicious communication patterns, avoiding the sudden instability that comes with strict binary classification.

Figure 4 tells us that the normal or benign traffic always stays within the normal or somewhat normal regions. On the other hand, the presence of bot activity makes the score to be moved toward the abnormal zones. Table 4 summarizes the resulting classification outcomes.

Since the benign host remains below the global threshold ($0.20 < 0.28$), it is accurately managed without triggering false alarms. However, the infected host score significantly exceeds the decision boundary ($0.40 > 0.28$), validating the role of the multi-feature fusion architecture in combining behavioral evidence. The difference between composite score (S) of the benign host and infected host clearly depicts how effective the framework proposed enhances the class separability. A larger margin above the decision threshold means clearer separation between benign and botnet traffic—which translates to better detection performance and fewer false-positive classifications.

Figure 4
Fuzzy behavioral interpretation of the composite anomaly score generated by the fusion framework

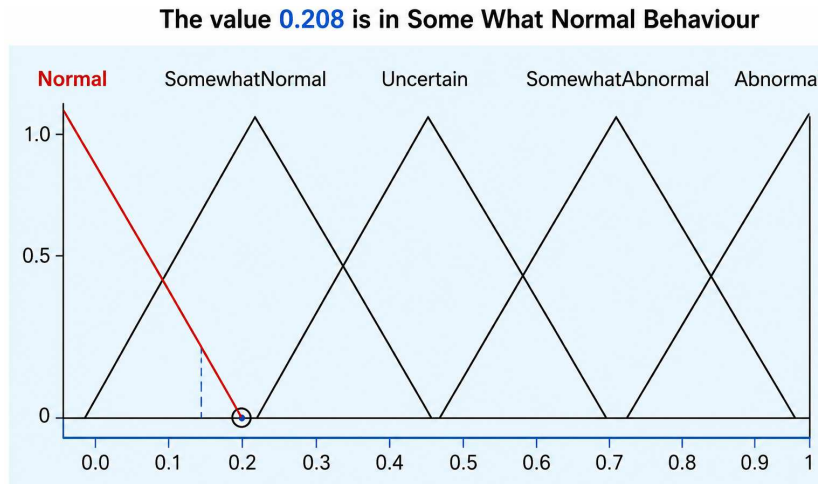


Table 4
Global behavioral fusion verdict outcomes

Host type	Composite score (S)	Global threshold (τ_S)	Classification outcome
Benign host	0.20	0.28	Normal status
Bot-infected host	0.40	0.28	Suspicious/alert prompted

4.6. Validation of the online sliding-window architecture

In order to assess the effectiveness of the architecture, the features were continuously updated within every rolling window as it is been previously discussed in Section 3.3. During each update interval Δt , the metadata of the incoming traffic was added to the active host profile, while the observations falling outside the rolling window were discarded.

The experimental findings demonstrate that the rolling-window approach supports continuous analysis of streaming network traffic while maintaining long-term behavioral characteristics. Over the course of successive update intervals, benign hosts displayed relatively stable autocorrelation values, while bot-infected hosts showed persistent behavioral patterns which was steadily exceeded the adaptive thresholds.

The framework catches both abrupt behavioral changes and gradually evolving C2 activities without requiring offline batch processing as it has a continuously updating mechanism. These results suggest that the proposed system that uses the streaming is well-suited for continuous behavioral monitoring in real time and in operational network environments, all while keeping computational overhead low.

4.7. Computational efficiency and real-time operation

The framework estimates temporal persistence over a rolling historical window of length (H), avoiding repeated processing of the complete historical traffic data. The mathematical processing complexity of the online autocorrelation engine is restricted per-host update step to $O(H)$.

Deployment in real-time monitoring environments with minimal processing overhead is made possible by computational

requirements being bounded and predictable since H only represents the active sliding historical horizon rather than the complete traffic archive.

The framework adapts well even in constant, high-volume network monitoring settings due to its minimal computational complexity. The simple architectural design allows real-time observation of the ongoing traffic and a prompt reaction to new behavioral changes or irregularities. Offline ML methods typically need regular retraining or batch processing, which increases overhead. The proposed framework avoids that—its rolling sliding-window design lets it constantly adjust to new streaming traffic on its own.

4.8. Qualitative comparative evaluation with recent frameworks

To show clearly how this work relates to and improves on existing architectures in the literature, a structural qualitative comparison was carried out against the same state-of-the-art methods referenced earlier in the related work section.

The capability comparison across critical deployment parameters is summarized in Table 5.

Table 5 shows the proposed framework differs from recent state-of-the-art botnet detection methods in several important ways. Deep learning-based approaches generally hit high classification accuracy, but they need large labeled datasets, heavy training, and frequent model retraining to keep up in evolving network environments. Periodicity-based methods work well for regular C2 communication, but they may perform worse when attackers use irregular beaconing or temporal jitter.

Similar to this, offline preprocessing is usually required for sliding-window segmentation methods before behavioral analysis can be performed and carried out. The proposed framework takes

Table 5
Qualitative architectural comparisons with recent state-of-the-art detection frameworks

Framework	Detection principle	Requires labeled training data?	Real-time live capture?	Handles encrypted sessions?	Computational overhead profile	Practical deployment constraint
BADD-CNN (2025)	Deep CNN	Yes	No	Limited	High	Requires labeled datasets and payload-dependent feature extraction
Lomb-Scargle (2025)	Periodicity analysis	No	Partial	Yes	Medium	Effective mainly for periodic C2 communications
Sliding time analysis (2022)	Traffic segmentation	No	Partial	Yes	Medium	Offline preprocessing and segmentation overhead
Proposed framework	Temporal persistence + fuzzy inference	No	Yes	Yes	Low	Requires an initial benign baseline learning phase

a different way; it continuously analyzes temporal behavioral persistence using lightweight flow-level metadata, without looking at packet payloads or requiring supervised training. This makes it suitable for continuous deployment in enterprise environments where communications are encrypted and constantly evolving traffic patterns limit how far conventional signature-based and supervised learning approaches can go. The temporal behavioral modeling strategy at the center of the framework is largely responsible for the reported performance indicators.

Traditional approaches tend to analyze traffic observations in isolation. The rolling-window architecture takes a different route—it captures behavioral persistence over time, which lets the framework tell sustained botnet activity apart from the transient fluctuations typical of normal legitimate traffic. On top of that, the adaptive thresholding mechanism and fuzzy behavioral fusion cut down false-positive detections by pulling information from multiple behavioral indicators into one unified decision process. Together, these pieces drive the improvements seen in accuracy, precision, recall, F1-score, and ROC-AUC during the experimental evaluation.

Overall, the experimental findings demonstrate that temporal behavioral persistence is a useful marker for differentiating botnet activity from typical network activity. The proposed framework successfully detects both persistent and evolving behaviors of botnet while supporting continuous analysis of streaming network traffic by integrating online temporal analysis, adaptive thresholding, and multi-feature behavioral fusion. Additionally, it maintains low processing overhead in encrypted network contexts and under changing communication times. These results support the proposed methodology and also indicate its potential for practical deployment in real-world network monitoring settings.

5. Conclusion

This paper presents a network forensics approach for detecting bots that is built around drift in temporal behavioral patterns, instead of single traffic snapshots. The real-time analysis brings together real-time traffic analysis, autocorrelation-based temporal

modeling, and a fuzzy inference engine to track evolving network activity over time without relying exclusively on predefined decision criteria. The approach detects communication patterns linked to botnet C2 activities by integrating behavioral data with temporal persistence.

In the experimental evaluation, the proposed framework achieved an F1-score of 0.91 and a ROC-AUC of 0.94. Unlike many ML approaches, the detection framework itself does not require labeled training data during operation. The labeled datasets used in this study served only to evaluate how discriminative the extracted behavioral features were, through comparative baseline models—they weren't part of the actual detection process. The autocorrelation-based temporal features consistently captured persistent behavioral patterns that differentiate the abnormal and botnet activity from normal and legitimate network traffic. As a result, the proposed framework achieved better separation between benign and botnet traffic than the threshold-based baseline, while keeping the false-positive rate low.

Even though the results are promising, the system still has limitations that need to be addressed. At first, the framework depends on a completely clean baseline period (B_{normal}). If an attacker has already compromised a host before this baseline is built, the adaptive thresholds risk absorbing that malicious signature as normal user behavior. There's also the question of highly sophisticated attackers, who might still slip through during high-traffic bottlenecks by deliberately injecting massive timing delays or random communication jitter. Future work will look at group-based behavioral profiling across a larger number of machines to catch pre-existing compromises that may not be detectable using a single-host baseline. Another direction is the integration of the fuzzy inference engine with deep generative models to improve robustness against increasingly adaptive botnet behaviors.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in Figshare at <https://figshare.com/s/c5c40c16d0f343a2bf31>.

Author Contribution Statement

Shivanna Doddegowda: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Pooja Mysore Raghupathi:** Conceptualization, Methodology, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

- [1] Wadhwa, P., Moudgil, A., & Bagchi, A. (2024). Advanced techniques in intrusion detection for botnet traffic analysis. In *2024 2nd International Conference on Advances in Computation, Communication and Information Technology*, 1, 1383–1388. <https://doi.org/10.1109/ICAICIT64383.2024.10912148>
- [2] Sandra, A. T., Ashok, A., & Nair, R. T. (2023). Blockchain based peer to peer botnet detection using Louvain algorithm. In *2023 14th International Conference on Computing Communication and Networking Technologies*, 1–6. <https://doi.org/10.1109/ICCCNT56998.2023.10307429>
- [3] Alissa, K., Alyas, T., Zafar, K., Abbas, Q., Tabassum, N., & Sakib, S. (2022). Botnet attack detection in IoT using machine learning. *Computational Intelligence and Neuroscience*, 2022(2), 1–14. <https://doi.org/10.1155/2022/4515642>
- [4] Garg, R., Rayudu Durgarao, V. G., Dewan, L., & Pal, M. (2025). A comparative study of ML classifiers for DDoS botnet detection in UAV-enabled IoT environments. In *2025 International Conference on Emerging Technology in Autonomous Aerial Vehicles*, 1–5. <https://doi.org/10.1109/ETAAV66793.2025.11213251>
- [5] Oškera, D., Koumar, J., Pokorná, A., Jeřábek, K., & Čejka, T. (2025). Botnet detection through periodic patterns in command-and-control network traffic. In *2025 21st International Conference on Network and Service Management*, 1–6. <https://doi.org/10.23919/CNSM67658.2025.11297465>
- [6] Owen, H., Zarrin, J., & Pour, S. M. (2022). A survey on botnets, issues, threats, methods, detection and prevention. *Journal of Cybersecurity and Privacy*, 2(1), 74–88. <https://doi.org/10.3390/jcp2010006>
- [7] Chavan, N., Kukreja, M., Jagwani, G., Nishad, N., & Deb, N. (2022). DDoS attack detection and botnet prevention using machine learning. In *2022 8th international conference on advanced computing and communication systems*, 1, 1159–1163. <https://doi.org/10.1109/ICACCS54159.2022.9785247>
- [8] Ibrahim, Z. A., Razali, R. A., Ismail, S. A., Azhar, I. H. K., Rahim, F. A., & Azilan, A. M. A. (2022). A review of machine learning botnet detection techniques based on network traffic log. In *2022 IEEE International Conference on Computing*, 204–209. <https://doi.org/10.1109/ICOCO56118.2022.10031803>
- [9] Benedictus, F., Putra, M. A. R., Ahmad, T., Za'in, C., & de Souza-Daw, T. (2025). Improving spam botnet detection through convolutional model and geolocation feature enhancement in a novel three-class classification task. *International Journal of Intelligent Networks*, 6, 185–203. <https://doi.org/10.1016/j.ijin.2025.08.001>
- [10] Ramadhan, I. F., Putra, M. A. R., De La Croix, N. J., Ahmad, T., Kannole, V. E., Mtonga, K., . . . & Ijtihadie, R. M. (2025). SB-Net: A novel spam botnet detection scheme with two-stage cascade learner and ensemble feature selection. *IEEE Open Journal of the Communications Society*, 6, 7237–7255. <https://doi.org/10.1109/OJCOMS.2025.3603425>
- [11] Quamer, A., & Kaur, H. (2025). Real-time botnet detection using temporal convolutional networks and self-organizing map. In *2025 2nd International Conference On Multidisciplinary Research and Innovations in Engineering*, 693–698. <https://doi.org/10.1109/MRIE66930.2025.11156544>
- [12] Shinan, K., Alsubhi, K., Alzahrani, A., & Ashraf, M. U. (2021). Machine learning-based botnet detection in software-defined network: A systematic review. *Symmetry*, 13(5), 866. <https://doi.org/10.3390/sym13050866>
- [13] Shivanna, D., & Pooja, M. R. (2024). An empirical assessment of botnets and detection methods. *Grenze International Journal of Engineering and Technology*, 10(1), 2828–2835.
- [14] Tadamalla, N. (2025). Real-time visualization and optimization of peer-to-peer botnets for efficient management and propagation control. In *2025 IEEE 2nd International Conference on Information Technology, Electronics and Intelligent Communication Systems*, 1–6. <https://doi.org/10.1109/ICITEICS64870.2025.11341131>
- [15] Ramesh, R. B., & Thangaraj, S. J. J. (2023). Analyzing and detecting botnet attacks using anomaly detection with machine learning. In *2023 5th International Conference on Inventive Research in Computing Applications*, 911–915. <https://doi.org/10.1109/ICIRCA57980.2023.10220903>
- [16] Ali, M., Shahroz, M., Mushtaq, M. F., Alfarhood, S., Safran, M., & Ashraf, I. (2024). Hybrid machine learning model for efficient botnet attack detection in IoT environment. *IEEE Access*, 12, 40682–40699. <https://doi.org/10.1109/ACCESS.2024.3376400>
- [17] Bonsu, M. A., & Akekudaga, P. (2025). Privacy-preserving lightweight IDS for smart home botnet defense. In *2025 IEEE 16th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*, 0784–0790. <https://doi.org/10.1109/UEMCON67449.2025.11267724>
- [18] Qasim, M., Waleed, M., Um, T. W., Pahlevani, P., Pedersen, J. M., & Masood, A. (2024). Diving deep with botlabds1: A novel ground truth-empowered botnet dataset. *IEEE Access*, 12, 28898–28910. <https://doi.org/10.1109/ACCESS.2024.3367122>
- [19] Yan, Y., Lang, B., Meng, X., & Xiao, N. (2024). Toward generating communication graph datasets for botnet detection in autonomous systems. *IEEE Transactions on Information Forensics and Security*, 19, 7908–7923. <https://doi.org/10.1109/TIFS.2024.3453172>
- [20] Ali, M., Mushtaq, M. F., Akram, U., Aray, D. G., Vergara, M. M., Karamti, H., & Ashraf, I. (2025). Botnet detection in internet of things using stacked ensemble learning model. *Scientific Reports*, 15(1), 21012. <https://doi.org/10.1038/s41598-025-02008-9>

- [21] Lagraa, S., Husák, M., Seba, H., Vuppala, S., State, R., & Ouedraogo, M. (2024). A review on graph-based approaches for network security monitoring and botnet detection. *International Journal of Information Security*, 23(1), 119–140. <https://doi.org/10.1007/s10207-023-00742-7>
- [22] Velasco-Mata, J., González-Castro, V., Fidalgo, E., & Alegre, E. (2023). Real-time botnet detection on large network bandwidths using machine learning. *Scientific Reports*, 13(1), 4282. <https://doi.org/10.1038/s41598-023-31260-0>
- [23] Pierrard, R., Poli, J. P., & Hudelot, C. (2018). Learning fuzzy relations and properties for explainable artificial intelligence. In *2018 IEEE International Conference on Fuzzy Systems*, 1–11. <https://dx.doi.org/10.1109/FUZZ-IEEE.2018.8491538>
- [24] Goyal, M., & Mittal, S. (2024). Review for prevention of botnet attack using various detection techniques in IoT and IIoT. In *2024 2nd International Conference on Disruptive Technologies*, 259–264. <https://doi.org/10.1109/ICDT61202.2024.10489168>
- [25] Putra, M. A. R., Ahmad, T., Ijtihadie, R. M., & Hostiadi, D. P. (2022). Prototyping distributed botnet detection system in computer networks. In *2022 5th International Conference on Computing and Informatics*, 292–297. <https://doi.org/10.1109/ICCI54321.2022.9756127>
- [26] Rabhi, S., Abbes, T., & Zarai, F. (2023). Transfer learning-based Mirai botnet detection in IoT networks. In *2023 International Conference on Innovations in Intelligent Systems and Applications*, 1–5. <https://doi.org/10.1109/INISTA59065.2023.10310379>
- [27] Rao, K. S., & Reddy, D. M. (2024). Comprehensive intrusion detection for investigating network traffic and botnet attacks. In *2024 IEEE 6th International Conference on Cybernetics, Cognition and Machine Learning Applications*, 339–344. <https://doi.org/10.1109/ICCCMLA63077.2024.10871826>
- [28] Alghamdi, A., & Barsoum, A. (2024). A comprehensive IDS to detect botnet attacks using machine learning techniques. In *2024 IEEE 3rd International Conference on Computing and Machine Intelligence*, 1–6. <https://doi.org/10.1109/ICMI60790.2024.10585846>
- [29] Haq, S., & Singh, Y. (2018). Botnet detection using machine learning. In *2018 Fifth International Conference on Parallel, Distributed and Grid Computing*, 240–245. <https://doi.org/10.1109/PDGC.2018.8745912>
- [30] Pradeepthi, K. V., & Kannan, A. (2018). Detection of Botnet traffic by using neuro-fuzzy based intrusion detection. In *2018 Tenth International Conference on Advanced Computing*, 118–123. <https://doi.org/10.1109/ICoAC44903.2018.8939109>
- [31] Prasanth, V. S., Haritha, P. S., Likhitha, B., Mahalakshmi, K., & Akhther, A. P. (2025). Artificial intelligence based supervised learning model for attack detection of botnet and defence in IoT cybersecurity infrastructures using convolutional neural network. In *2025 International Conference on Inventive Computation Technologies*, 1535–1540. <https://doi.org/10.1109/ICICT64420.2025.11004980>
- [32] Hostiadi, D. P., & Ahmad, T. (2022). Sliding time analysis in traffic segmentation for botnet activity detection. In *2022 5th International Conference on Computing and Informatics*, 286–291. <https://doi.org/10.1109/ICCI54321.2022.9756077>

How to Cite: Doddegowda, S., & Raghupathi, P. M. (2026). Leveraging AI and Network Forensics for Smart Botnet Detection. *Artificial Intelligence and Applications*. <https://doi.org/10.47852/bonviewAIA62029953>