Detecting Stealthy Web Bots: A Behavioral Analysis Framework for OpenWPM Automation

Shakil Ahmed, Md. Mahedi Hasan Rigan, Md. Shohrab Hossain, *Member, IEEE*, and Anupam Das, *Senior Member, IEEE*

***Abstract*—Nowadays, Web bots are used extensively for tasks like search engine indexing and security assessments, but they can also facilitate malicious activities such as ad fraud and data theft and many more. However, existing approaches are unable to detect more advanced bots, such as those driven by Selenium or OpenWPM, which can conceal their browser fingerprint and imitate human browsing behaviors. In this paper, we propose a novel technique for identifying advanced bots, specifically those using OpenWPM, through behavioral analysis. Our approach considers four browsing behaviors, including mouse movement, mouse click, keystroke, and scrolling. We employ an ensemble of lightweight classification models trained on behavioral features, which are augmented using unsupervised clustering in a novel way to enhance detection performance. The detection system is designed in a modular fashion, making it resilient to missing behavioral data and independent of platform-specific features or tasks, enabling generalizability across diverse web platforms. The proposed approach achieves an F1-score of 98.8%, presenting a promising solution for detecting OpenWPM bots and other human-mimicking bots with improved precision.**

***Index Terms*—Bot detection, behavioral analysis.**

I. Introduction

In RECENT years, a significant portion of web traffic has been generated by automated web bots [1], often developed using automation frameworks such as Selenium [2], OpenWPM [3] or Puppeteer [4]. There are various types of bots with legitimate applications, including search engine bots, copyright enforcement bots, site monitoring bots, and commercial bots, each serving distinct purposes [1], [5]. While many of these bots are benign, there also exist a growing number of malicious bots that engage in harmful activities such as distributed denial-of-service (DDoS) attacks, spam, digital ad fraud, data scraping, credential stuffing, and more [6], [7]. These bots are capable of mimicking human browsing behavior [1], [8], making it increasingly difficult for websites to detect automated bot activity. For example, the WebTAP framework [9] leverages OpenWPM to track and analyze web behavior at scale, helping businesses stay transparent, comply with privacy regulations, and monitor advertising practices. However, the same tools that enable valuable research and compliance efforts can also be leveraged by malicious parties to automate web interactions at scale. Consequently, detecting and mitigating bot activity has become essential to ensuring web security.

Despite significant progress, bot detection remains a critical challenge. Traditional methods primarily focus on analyzing web session logs, using HTTP request features (e.g., GET/POST counts, error rates) and session-level patterns [8], [10], [11], [12], [13], [14]. However, these approaches struggle to detect sophisticated, human-mimicking bots. While some studies achieve promising results using browser fingerprinting [15], [16], [17] or graph-based website navigation analysis [18], [19], these methods are often circumvented by bots that spoof fingerprints [17] or imitate human-like navigation patterns. Mouse dynamics have emerged as a promising behavioral biometric, with studies extracting trajectory features for binary classification or employing deep learning on time-series or image-based representations [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. However, many of these studies rely on constrained data collection or synthetic datasets, lack validation against real-world bots, and are often limited to specific contexts such as blog platforms. These limitations highlight the need for a more generalizable and robust behavioral analysis framework capable of detecting advanced, human-mimicking bots across diverse web environments.

This work addresses the shortcomings of existing methods for detecting human-mimicking bots by proposing a modular approach that leverages behavioral features extracted from mouse movements, mouse clicks, mouse scroll, and keystroke dynamics. Each module is designed to operate independently, enabling robust performance even when partial data is available. The approach integrates unsupervised clustering with supervised classification to improve detection accuracy in a flexible and extensible manner. To that end, we aim to answer the following research questions:

RQ1: To what extent can browsing behavior data effectively distinguish bots from humans, and which behavioral characteristics offer the most discriminative power? We develop a model that leverages mouse dynamics, including movement and clicking, as well as scrolling patterns and keystroke activity, to effectively identify bots. Our analysis reveals that certain

Received 1 June 2025; revised 26 November 2025; accepted 28 November 2025. Date of publication 3 December 2025; date of current version 12 March 2026. *(Shakil Ahmed and Md. Mahedi Hasan Rigan contributed equally to this work.) (Corresponding author: Anupam Das.)*

Shakil Ahmed, Md. Mahedi Hasan Rigan, and Md. Shohrab Hossain are with the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka 1000, Bangladesh (e-mail: shakilahmedndc@gmail.com; rigan5031@gmail.com; mshohrabhossain@cse.buet.ac.bd).

Anupam Das is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (e-mail: anupam.das@ncsu.edu).

Digital Object Identifier 10.1109/TDSC.2025.3639982

behavioral characteristics consistently differ between humans and bots, offering strong discriminative signals for robust detection.

RQ2: In what ways can the performance of classification-based machine learning models be improved for detecting human-mimicking bots through behavioral signals? To address this question, we propose leveraging a diverse ensemble of classification models whose predictions are aggregated to enhance overall accuracy. Additionally, we integrate clustering techniques within each model to capture latent structures in behavioral signals, such as mouse movement patterns and keystroke dynamics, thereby refining detection capabilities.

RQ3: Which combinations of behavioral features contribute most to accurate bot detection, and how robust and generalizable is the system to missing interaction types? We evaluate different combinations of behavioral actions to identify which ones contribute most to accurate bot detection. Building on this, we design a modular detection framework where each action type is analyzed independently, allowing the system to remain effective even when some data types are missing. By avoiding platform-specific actions (e.g., drag-and-drop, timing entropy), the approach generalizes well across web contexts. The outputs of each module are combined through an ensemble-based mechanism, enabling reliable detection even under incomplete or noisy behavioral input.

The key contributions of this work are as follows.

- *OpenWPM bot detection via behavior:* We are the first to detect automation tools like OpenWPM based solely on browsing behavior. To support this, we developed a dataset with 150 human participants in an unconstrained, naturalistic setting, augmented with sessions from OpenWPM-driven bots. This provides insight into behavioral differences between humans and bots. For example, we observe that the mouse move action exhibits the most distinguishable behavioral patterns, particularly in features like path efficiency and slope variance.
- *Clustering-assisted classification:* We propose a novel technique that integrates unsupervised clustering of browsing patterns with supervised classification, enabling improved detection performance, even with lightweight classification models.
- *Enhanced behavioral features:* We introduce a new mouse movement feature and leverage the relatively underutilized mouse scrolling behavior to better distinguish between human and bot interactions. From our evaluations, the mouse scrolling action type alone achieves an average F1-score of 84.6%, and up to 98.8% when combined with other interaction types, demonstrating its strong effectiveness in behavioral-based detection.
- *Modular design for adaptability:* Our system is implemented in a modular fashion, allowing each interaction type such as mouse movements, clicks, keystrokes, and scrolling to be analyzed independently. This ensures robustness even when certain signals are absent. Notably, when excluding the most commonly used signal in prior work—mouse movement, our system still achieves an F1-score of 90.1% using the remaining interaction types. We also demonstrate generalizability by training and testing the model across different websites.

The rest of the paper is organized as follows. In Section II, we review background and related work on bot detection, highlighting distinctions from prior approaches. Section III describes the proposed approach, outlining the system architecture and its key components. Section IV details the experimental setup, the dataset used, and presents the performance results of the proposed detection system. Section V discusses the limitations of our approach and provides a broader perspective on its practical implications and potential areas for improvement. Finally, Section VI concludes with a summary of contributions.

II. Background and Related Work

A. Existing Works

Early bot detection methods primarily relied on web session log analysis [8], [10], [11], [12], [13], [14], extracting simple statistics such as request counts, error rates, referrers, and session durations. These approaches effectively identify basic automation but struggle against sophisticated bots that mimic human browsing. Browser fingerprinting [15], [16], [42], [43] has also been widely used, leveraging device-level attributes like screen resolution, fonts, and rendering inconsistencies. However, such methods are increasingly circumvented by evasion techniques, including OpenWPM-specific fingerprint spoofing [17], which undermines their robustness. Network-based approaches [44], [45] offer complementary signals but show limited accuracy and scalability.

In contrast, mouse dynamics have emerged as a promising alternative, offering behavioral cues difficult for bots to replicate [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]. Previous studies [22], [41] used engineered features from mouse movements, clicks, and keystrokes with traditional classifiers like C4.5 which demonstrated strong performance in controlled settings such as blog platforms. However, these datasets are often constrained, platform-specific, and limited in realism. Later studies adopted deep learning by converting trajectories into images or sequences. 1D-CNN and LSTM-based models [21], [23], [26] achieved high reported accuracies (up to 99.91%), though typically under repetitive tasks in login forms which limit their generalizability to unconstrained browsing. Neuromotor and GAN-based trajectory generation [25] produced more human-like patterns, but Random Forest models still detected them due to underlying artifacts. Importantly, handcrafted statistical features [22] often outperformed neuromotor features, especially against synthetic bots. ReMouse [28] investigated session replay bot detection by analyzing intra-user variability with Dynamic Time Warping and deep embeddings.

Graph-based approaches have been widely explored for social bot detection [18], [19], [32], [33], [34], [35], [36], [37], [46], [47], [48], leveraging the rich information available in social networks including textual content, user attributes, and interaction relations. Session graphs model pages as nodes and transitions as edges, enabling Graph Convolutional Networks to identify abnormal navigation. Multiscale and hypergraph models [35], [47] improve detection through structural and

TABLE I
SUMMARY OF COMPARISON WITH THE PRIOR TECHNIQUES (✓: INDICATES SUPPORTED/APPLICABLE; ×: INDICATES NOT SUPPORTED/NOT APPLICABLE; ◧: NOT PASSIVE DETECTION; †: CREDENTIAL STUFFING BOTS; –: NO EXPLICIT RESULT)

Literature	Type of Data	Techniques	Bot Type	Detection Rate	Emulates Human Behavior	Cross-Platform Applicability	Model Complexity
[8], [11]	User agent, HTTP features	Classification model	Simple, Moderate	96.5%	×	×	Lightweight
[12]	Session & Http log	Clustering model	Known bots	91%-92%	×	×	Lightweight
[31]	Session & Http log	Positive-unlabeled Learning	Simple, Moderate	–	×	×	Moderate
[24]	Mouse kinematics	Clustering, Fractional Derivatives	Clicker bots	–	×	×	Moderate
[18], [19]	Site navigation graph	CNN, Graph CNN	page repetitive bots	–	×	✓	Heavy
[28]	Mouse Dynamics	DTW, VGG16(DL Model)	Session Replay bots	–	×	×	Heavy
[26], [27]	Mouse Trajectory	LSTM, ResNet	Credential stuffing bots	99.5%-99.91%	†	×	Heavy
[25]	Mouse trajectory, Neuromotor feature	Classification, LSTM	GAN based bot	99.1%	◧	×	Heavy
[23]	Mouse Trajectory	CNN, LSTM	Statistical attack bots	96.2%	✓	✓	Heavy
[21]	Mouse trajectory, Web session log	CNN, classification, ensemble	Moderate, Advanced	93%	✓	✓	Heavy
[32]–[37]	Text, User metadata	Graph, GNN, GCNN	Social bot	-	✓	×	Heavy
[38]–[40]	Text, User metadata	Transformer, Graph	Social bot	-	✓	×	Heavy
[22], [41]	Mouse & Keystroke dynamics	Classification model	Blog bots	99.45%	✓	×	Lightweight
Ours	Mouse & Keystroke dynamics	Classification, Clustering	OpenWPM-based bots	98.8%	✓	✓	Lightweight

semantic features. Earlier flow-based graph models [48], [49] faced scalability issues and were less effective on encrypted traffic. Despite strong performance, graph-based approaches face several limitations. They are often platform-dependent, with most validation limited to Twitter-like datasets, leaving cross-platform robustness uncertain. Their computational cost makes them less practical in low-resource environments. Finally, most assume static graphs, overlooking the dynamic nature of social networks where bots change connections, adopt dormant "sleeper" strategies, or evolve posting behaviors over time.

Transformer-based methods [38], [39], [40] further exploit semantic and sequential cues. Heterogeneity-aware graph transformers [38] and pretrained language models [50], [51] enhance detection of sophisticated or coordinated bots, often by integrating relational and multimodal features [40]. However, their computational cost and platform dependency limit broad deployment. These findings collectively demonstrate that handcrafted features remain highly competitive and may even rival deep models when supported by thoughtful feature engineering. Table I provides an overview of related works, summarizing their approaches, techniques, and key limitations relevant to our study.

B. Distinction From Existing Work

Prior research on bot detection has largely focused on single interaction types—primarily mouse movement [23], [25], [26], [27] or on browser fingerprinting. While effective in controlled settings, these approaches often depend on platform-specific signals, degrade when interaction data is incomplete, and struggle to generalize across diverse browsing environments. Many deep learning–based methods also require high computational costs and lack interpretability, making them less practical for real-world deployment. In contrast, we introduce a novel feature for mouse movement that enhances the separation between human and bot behavior, improving classification effectiveness. Additionally, our approach combines unsupervised clustering of browsing patterns with machine learning based classification in a novel, lightweight framework. This integration facilitates robust detection of OpenWPM-driven bots, even with minimal computational overhead. By evaluating detection performance across multiple interaction types and their combinations, we demonstrate that our method generalizes effectively across varied types of user behavior, not just platform-specific tasks.

III. PROPOSED APPROACH

The goal of our system is to distinguish between human users and bots based on their browsing behavior, specifically by analyzing mouse dynamics and keystrokes. While other browsing behaviors, such as HTTP features, can provide some insight, they are not particularly effective on their own against advanced bots that emulate human behavior [8]. Combining these features with mouse dynamics often leads to a simple aggregation of scores, which does not significantly improve detection performance [21]. Therefore, we propose using a comprehensive set of mouse related features, including mouse movements, clicks, scrolling, and keystrokes, to enhance detection accuracy. To ensure that the model is adaptable to a wide range of websites, we employ separate modules dedicated to each of these four types of actions. Fig. 1 illustrates the overall workflow of the proposed detection system. The detection system consists of four components:

1) *Frontend Logger:* A lightweight JavaScript module embedded in webpages to passively capture raw user interaction events (mouse, scroll, keypress), without adversely affecting user experience or consuming significant bandwidth. A more detailed analysis of the associated overhead and resource consumption is provided in Section IV-G.
2) *Backend Analyzer:* Processes raw UI events into high-level behavioral actions by aggregating and compressing sequential inputs for efficient representation.
3) *Clustering and Classification:* Enhances features via class-wise K-Means clustering and trains multiple classifiers (SVM, RF, XGBoost) on interaction-specific data.
4) *Prediction Fusion:* Combines predictions from multiple classifiers using a meta-classifier, which learns to aggregate their outputs and produce a more accurate final human/bot decision per session.

A. Frontend Logger

The frontend logger in our system is modeled after the approach described by Chu et al. [22], [41]. Implemented in JavaScript and embedded across all pages of the test website, the logger operates passively in the background, requiring no

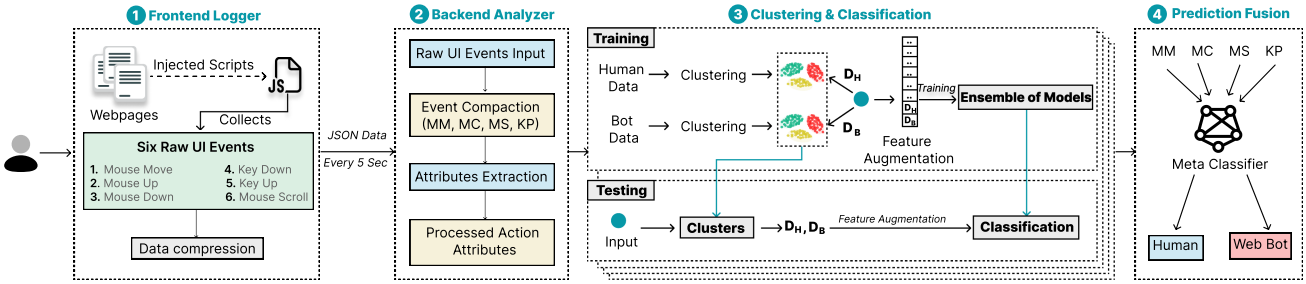

Fig. 1. Our approach consists of: (1) a lightweight frontend logger that passively captures raw UI events without impacting user experience, (2) a backend analyzer that compacts and maps events to high-level behaviors, (3) a feature-augmented clustering and ensemble classification module for human/bot detection, and (4) a fusion unit that aggregates classifier outputs across interaction types (MM = Mouse Move, MC = Mouse Click, KP = Key Press, MS = Mouse Scroll) for robust session-level decisions.

TABLE II
USER ACTIONS AND THEIR BEHAVIORAL ATTRIBUTES

Action	Description	Attributes
Mouse Move	The movement of the mouse in any direction	Distance, Displacement, Duration, Velocity, Path efficiency, Net path slope, Mean path slope, Variance of slope
Mouse Click	The press and release of the left or right mouse button	Duration, Inter arrival time
Mouse Scroll (Scrolling)	The action of moving the scroll wheel on a computer mouse up or down	Duration, Average scroll velocity
Key Press (Keystroke)	The press and release of a keyboard button	Key press duration, Inter arrival time

explicit user interaction. It captures a range of user behaviors, including mouse movements, keypresses, and scrolling activity, whenever the user engages with any webpage element. Consistent with the referenced work, our logger records six core UI events: Mouse Move, Mouse Down, Mouse Up, Key Down, and Key Up, with the addition of a Scroll event. Each event is captured using JavaScript event listeners, which are triggered upon user interaction and log the event data in JSON format. The logger temporarily buffers these events and periodically transmits them in batches to the backend for further processing.

To ensure minimal impact on user experience and bandwidth, we adopt the client-side cursor data compression techniques proposed by Leiva et al. [52], which significantly reduce the volume of transmitted data without compromising behavioral fidelity. By applying a combination of lossy and lossless methods optimized for web analytics, our system achieves efficient data transmission while preserving the resolution necessary for accurate bot detection. Additionally, asynchronous data collection and event buffering are employed to avoid blocking the main execution thread, further ensuring that user interactions remain unaffected.

B. Backend Analyzer

The Backend Analyzer processes the raw user interface data received from the frontend logger, compacting it into higher-level interaction events, following an approach partially inspired by Chu et al. [41]. These high-level events include Mouse Move, Mouse Click, Keystroke, and Mouse Scroll. Continuous mouse movements are segmented into multiple move events based on pauses during the motion. Similarly, Keydown and Keyup events are aggregated into a single Keystroke event, while Mouse Button Down and Mouse Button Up are merged into a Mouse Click event. Unlike the approach in [41], our Backend Analyzer introduces scrolling as a distinct event type and extracts additional features from mouse scrolling data. A detailed breakdown of these aggregated actions is provided in Table II. The attributes were selected based on established findings in behavioral biometrics [22], [25], [26]. Features such as distance, velocity, duration, and inter-arrival time are known to capture natural irregularities in human interactions that automated bots struggle to reproduce. In Section IV-B, we further justify this choice by showing how these properties consistently differentiate between human and bot behaviors. As a result, these properties serve as strong discriminative signals for building effective bot detection models.

For the mouse move action, we compute *Distance* as the total length of the mouse trajectory and *Displacement* as the straight-line distance between its start and end points. *Duration* refers to the time span of the mouse move event. *Velocity* is derived by dividing the total distance by the corresponding duration. We calculate *Path Efficiency* as the ratio of displacement to distance, where lower values indicate more erratic motion. Additionally, the *Net Path Slope* captures the slope of the direct line between the start and end points, while the *Mean Path Slope* and *Variance of Slope* provide insights into directional

consistency by evaluating the average and variation of slopes along the movement trajectory. Also, if the time between two consecutive mouse events exceeds 200 milliseconds, they are considered part of separate mouse movements.

For discrete input events such as mouse clicks and key presses, we compute *Click/Press Duration* as the time interval between the down and up actions of a button or key, and *Click/Press Inter-arrival Time* as the time gap between the end of one click or key press and the start of the next. *Mouse Scroll Duration* captures the total time of a scrolling episode, segmented by pauses in scrolling activity, while *Mouse Scroll Average Velocity* is computed as the total scroll distance (measured in pixels) divided by the scroll duration. The resulting high-level action features are then separated and routed to their respective clustering and classification models for further analysis.

C. Clustering and Classification

1) Clustering-Based Feature Augmentation: To distinguish between human and bot behaviors, we first collect real-world interaction data from both human users and automated bots (more details in Section IV). We apply clustering techniques (K-Means or Agglomerative clustering) separately to the human and bot training sessions in the feature space. Specifically, human-class sessions are grouped into K_H clusters and bot-class sessions into K_B clusters, resulting in prototypical centroids that represent typical human and bot behavior, respectively. We compute the euclidean distance between each input data point and the nearest centroid from both the human and bot clusters, resulting in two distance values: D_H (distance from the nearest human cluster) and D_B (distance from the nearest bot cluster). These two distances are then appended to the original feature, and the augmented data is used to train an ensemble of models. This clustering step embeds unsupervised structure into the feature representation, enriching the raw behavioral metrics.

2) Classification Models: After feature engineering, we train supervised classifiers to predict human versus bot on the enriched feature vectors. To exploit the heterogeneous signals, separate models are trained for each action type (mouse, scroll, click, keystroke). We train three standard classifiers: a Support Vector Machine (SVM), a Random Forest, and an XGBoost gradient-boosted tree model. These algorithms are widely used in behavioral classification tasks and have shown high accuracy: for example, RF and XGBoost achieved top scores on keystroke/mouse dynamics data [25], [53], and SVM/RF have yielded more than 90% accuracy on similar biometric data [54]. To further enhance prediction performance, we combine the outputs of these models using an ensemble strategy. The goal of ensemble learning is to leverage the complementary strengths of diverse algorithms and models, achieving higher predictive accuracy than individual models alone through their combined decision-making [8], [21], [53].

D. Prediction Fusion

At inference time, each trained model (across all action types) produces a probability of bot prediction for the current session. These predictions are then fed into a meta-classifier, which learns to weigh and combine the outputs to form the final decision. The meta-classifier is included in the proposed approach primarily to learn the optimal weights for different action types instead of assigning the same weight to each action type. In our implementation, the meta-classifier is a logistic regression model trained on the probability outputs from the four interaction-specific models (mouse movement, mouse click, mouse scroll, and key press). Logistic regression is chosen over other neural networks to avoid adding additional complexity to the model. Learning-based fusion of diverse behavioral classifiers can significantly boost overall accuracy. The final system thus produces a single classification indicating "human" or "bot."

IV. Evaluation

This section presents a concise overview of our experimental setup and key findings. We begin by describing the datasets used for training and evaluation. Next, we analyze behavioral differences between human users and bots across interaction types. We then report the performance of our proposed detection approach using several classification models, including individual and ensemble methods. The impact of clustering-based feature augmentation is also evaluated. Finally, we present ablation and cross-validation results to evaluate model robustness, followed by an analysis of system overhead in terms of computational efficiency, latency, and bandwidth usage.

A. Experimental Setup

1) Data Collection Framework: To evaluate the effectiveness of our bot detection framework, we constructed a dataset comprising both human and bot-generated browsing sessions. The data was collected using a custom-built interactive website designed to emulate a real-world e-commerce site. This website allowed users to perform typical shopping-related activities, including browsing through a collection of products, viewing detailed descriptions, adding items to a shopping cart, and submitting textual feedback for individual products. The user interface included scroll capabilities on the homepage to encourage natural scrolling behavior, alongside dynamic content loading to better reflect realistic interaction patterns. Mouse movements, clicks, scroll events, and keystroke inputs were continuously tracked across all pages. These rich behavioral traces were recorded for both human users and scripted bot agents.

To ensure that our detection system is broadly applicable across different types of web environments, we designed the interaction flows to represent generalized behavior patterns not limited to e-commerce but common across a variety of modern content-driven websites. Actions like browsing content, interacting with elements, and submitting forms are common across news, educational, and social media platforms. To evaluate the generalizability of our approach, we collected a small-scale dataset from a different type of web platform, a social media site called PostIt [55]. The platform resembles popular social networks, allowing users to browse and interact with posts, engage through likes and comments, view profiles, and search or sort content by various attributes.

Moreover, while all four types of interaction data were collected, our detection model is designed to be agnostic to interaction types, so it can operate effectively even when only a subset of interaction types is available. This ensures deployment flexibility in environments where, for example, keystroke logging may be restricted for privacy reasons or certain interactions are not applicable (e.g., read-only content).

2) Human Data Collection: For human data collection, 150 participants interacted with the website, with each participant visiting the site two to three times. This resulted in a total of 320 distinct web sessions. Each session lasted approximately 5 minutes, yielding over 25 hours of total interaction data. Interaction events were recorded using embedded frontend logging scripts that transmitted data in real-time to a backend storage system for analysis. For social site data collection, we used 15 new human subjects different from the e-commerce dataset, each performing two browsing sessions of approximately five minutes.

All data collection procedures were reviewed and approved by the Institutional Review Board (IRB) of the primary author's institution. Participants were informed of the study goals and their rights, including the ability to withdraw at any time by the study coordinators. All participants voluntarily took part in the study and provided their informed consent, acknowledging that their data would be used solely for research purposes. No financial compensation was offered for participation.

3) Bot Data Collection: For our data collection, we utilize OpenWPM, which leverages Selenium WebDriver to simulate human interactions. Standard Selenium actions such as mouse movements, clicks, typing, and scrolling are often linear, instantaneous, and easily distinguishable from genuine human behavior. To mitigate this, the Human-Like Interaction Selenium API (HLISA) [56] is employed, which enhances Selenium and OpenWPM by introducing more human like patterns into automated interactions. HLISA replaces Selenium's default ActionChains with augmented versions that incorporate realistic delays, variability, and behavioral subtleties.

Mouse movements are designed to follow curved, jittered paths with variable speed and correction phases, rather than straight, uniform trajectories. Clicking actions replicate human distributions, including dwell times and occasional imprecision. Scrolling is performed in natural tick intervals with intermittent pauses and simulated finger repositioning, while typing incorporates realistic dwell and flight times, pauses at natural linguistic boundaries, and appropriate use of modifier keys. These improvements slow down and diversify automated interactions, making them less uniform and reducing the likelihood of detection by bot-detection systems.

In our experiments, all bot agents employ HLISA, and additional randomness is incorporated for mouse clicks, keystrokes, and scrolling to further mimic human behavior. Some agents are designed to perform only a single type of action—such as mouse movements, clicks, or typing—to emulate specific behaviors like browsing, clicking, or form filling. Other agents combine multiple action types to replicate a broader range of genuine human interactions. Bot sessions are generated using OpenWPM version 0.21.1 in a Conda 23.3.0 environment on Ubuntu 22.04 LTS. A total of 20 unique OpenWPM browsing scripts were developed and each script was executed 10 times, producing bot sessions with an average duration of 8 minutes, resulting in approximately 26 hours of session data. This ensured a balanced dataset in terms of total volume between human and bot interactions. The bot scripts were adapted to the social media platform by implementing continuous scrolling, engaging with features such as likes, comments, and profile views, and simulating realistic browsing through adaptive hover durations over posts. A similar amount of bot session data was collected to match the human social site dataset.

4) Evaluation Criteria: We evaluate performance using standard metrics—Accuracy, Precision, Recall, and F1 score. Accuracy reflects overall correctness, Recall measures the model's ability to detect true bots, Precision indicates the proportion of correctly identified bots, and F1 balances Precision and Recall to capture overall detection effectiveness.

B. Behavioral Differences Between Humans and Bots

This subsection addresses *RQ1* by analyzing the behavioral differences between human users and bots across various interaction types. Behavioral data collected from both human users and OpenWPM-driven bots served as the foundation for analyzing differences in interaction patterns. To explore these behavioral distinctions, we conducted a comprehensive analysis of key user actions—namely mouse movement, clicking, scrolling, and key press behavior. The distribution plots in Fig. 2 illustrate the comparative behavior across 14 interaction features.

Notably, bots tend to exhibit higher values in mouse movement metrics, as shown in subplots (a), (b), (c), and (d), which highlight greater displacement, distance, velocity, and duration. These elevated values suggest scripted or mechanically efficient behavior, characterized by more consistent values and a tighter distribution. Trajectory related features such as path slope (e), mean slope (f), variance of slope (g), and path efficiency (h) indicate that for bots the peak of the distribution lies within a shorter range. While both groups show similar movement patterns, their distributions differ in values.

Mouse clicking behavior, illustrated in subplots (i) and (j), shows that both click duration and inter-arrival time are generally lower for bots compared to humans. Key press features, shown in subplots (k) and (l), follow a similar trend: human users exhibit greater variability and temporal inconsistency, whereas bot inputs are more uniform and repetitive. Scroll duration (m) and average scroll velocity (n) further reveal distinct distribution patterns between the two groups.

These findings provide compelling evidence that while bots can superficially imitate certain aspects of human interaction, they generally lack the nuanced temporal and spatial irregularities found in genuine behavior. The differences across features—especially in timing variability, trajectory randomness, and movement fluidity—can be leveraged as discriminative signals for building robust bot detection models. By modeling these subtle behavioral cues, it is possible to distinguish automated activity from human engagement with high confidence.

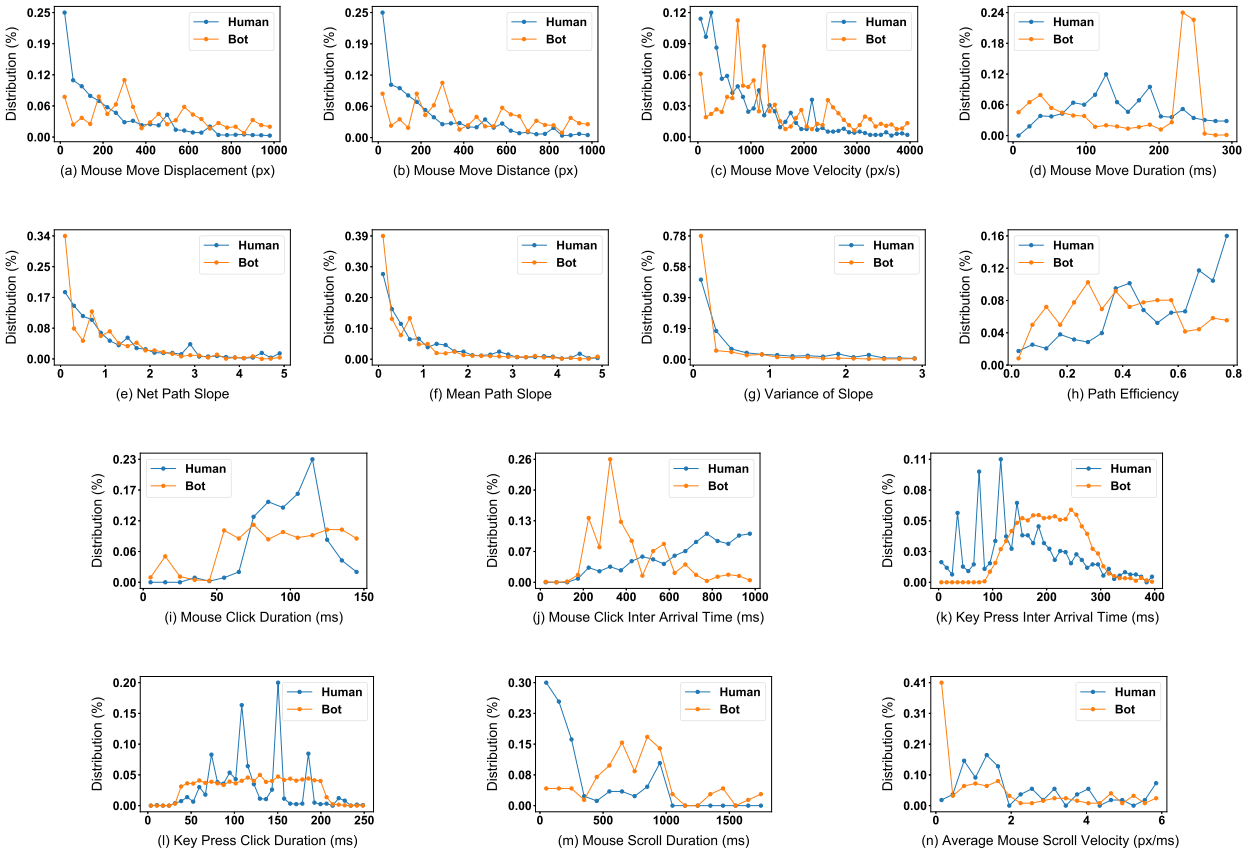

Fig. 2. Comparative analysis of human versus bot behavioral patterns in mouse and keyboard interactions.

C. Performance of Baseline Classification Models

This subsection addresses *RQ2* by evaluating the performance of our detection approach using multiple classification models, including both individual and ensemble methods. To evaluate the effectiveness of different machine learning models in distinguishing between human and bot sessions, we conducted extensive experiments across all four interaction types: mouse movement, clicks, scrolling, and keystrokes. We evaluate the performance of three widely used classifiers: Support Vector Machine (SVM), Random Forest, and XGBoost, along with an ensemble model that combines their predictions to enhance classification accuracy.

Each model was evaluated under two settings:

- *Raw Features Only:* Utilized only the original interaction features without any augmentation.
- *Augmented Feature Set:* Employed an extended set of features that included clustering-based distance metrics derived from either K-Means or Agglomerative Clustering, as detailed in Section III-C1.

Table III demonstrates the impact of clustering based feature augmentation on model performance. Across all models, incorporating clustering based distance features leads to improved accuracy compared to using only raw interaction features. For example, the ensemble model without clustering achieves an accuracy of 96.4%, while augmenting it with Agglomerative clustering improves the accuracy to 97.2%. Notably, using K-Means clustering results in the highest accuracy of 98.8%. This trend is consistent across individual models as well, with each showing improved performance when enhanced with clustering features, demonstrating the benefit of combining multiple classifiers and enhancing the feature space through unsupervised clustering. It is worth noting that the ensemble model with Agglomerative clustering shows a noticeable recall drop to 91.2%. This anomaly occurs because Agglomerative clustering can produce highly uneven cluster sizes, which sometimes position bot data closer to human clusters. As a result, some bot interactions are misclassified as human, thereby reducing recall despite otherwise strong overall performance.

D. Effectiveness of Cluster-Augmented Features

This subsection continues addressing *RQ2* by evaluating how clustering-based feature augmentation influences the effectiveness of bot detection.

1) Impact of Cluster-Distance Features: To enhance our classifier's ability to capture subtle, non-linear behavioral differences between human and bot interactions, we augmented our feature space with cluster-distance metrics derived from unsupervised K-Means clustering. The motivation behind this step is rooted in the observation that different user types (humans versus bots) exhibit distinguishable structural groupings in the behavioral feature space—particularly in mouse movement

TABLE III
PERFORMANCE COMPARISON OF BASELINE MODELS WITH AND WITHOUT CLUSTERING-BASED FEATURES

Model	Clustering	Accuracy	Precision	Recall	F1 Score
SVM	None	92.5	90.6	94.2	92.4
Random Forest		94.1	94.5	93.1	93.8
XGBoost		94.7	94.3	94.5	94.4
Ensemble (SVM+RF+XGB)		96.4	95.3	97.8	96.5
SVM	Agglomerative	95.4	94.9	95.5	95.2
Random Forest		96.3	97.9	95.8	96.8
XGBoost		96.5	97.3	95.4	96.3
Ensemble (SVM+RF+XGB)		97.2	98.4	91.2	94.7
SVM	K means	94.1	94.6	94.3	94.4
Random Forest		98.1	96.5	99.7	98.1
XGBoost		98.1	98.0	98.4	98.2
Ensemble (SVM+RF+XGB) (Ours)		**98.8**	**98.5**	**99.1**	**98.8**

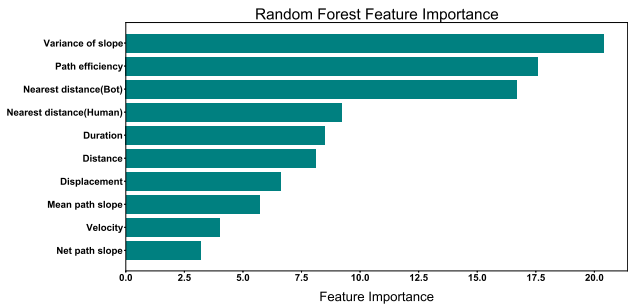

Fig. 3. Feature importance of mouse movement module classification.

dynamics. Fig. 4 illustrates the spatial separation of human and bot sessions in the cluster space.

By applying K-Means clustering independently to the human and bot training data, we generated separate centroids for each class. Then, for every new data point, we computed its euclidean distance to the closest human cluster centroid and the closest bot centroid. These two distance values were appended to the original feature vector as additional features.

The intuition behind using cluster distances is twofold. First, behavioral data lies in a high-dimensional space where proximity to learned behavioral prototypes (i.e., centroids) provides a strong discriminative signal. Second, these distance metrics encode how typical or anomalous a session is relative to known human or bot clusters, which is especially effective when raw features overlap.

Empirical results clearly show the advantage of this augmentation. As seen in Table III, integrating cluster-distance features consistently improves classification accuracy across all base classifiers. Our ensemble model, in particular, improves from 96.4% (raw features only) to 98.8% with K-Means augmented features. Notably, this approach also enhances precision and recall.

To assess the impact of augmented distance-based features on classification, we plot the feature importance graph from the Random Forest model trained on the mouse movement module (Fig. 3). Feature importance is computed based on the reduction in Gini impurity across decision trees. The results show that slope variance is the most influential feature, followed closely by path efficiency and nearest distance to bot clusters, highlighting their strong role in distinguishing human and bot behavior.

2) Optimal Cluster Count (K) in K-Means: Determining the appropriate number of clusters (K) is critical for ensuring that the centroids used in our cluster-distance feature augmentation reflect meaningful and representative behavioral patterns. To this end, we performed a detailed clustering evaluation using three well-established internal validation metrics: *Silhouette Score*, *Davies-Bouldin Index*, and *Calinski-Harabasz Index*. This analysis was conducted separately for human and bot datasets across each of the four action types (Mouse Move, Mouse Click, Scroll, and Keystroke), resulting in eight total K-Means clustering tasks.

We focus here on the Mouse Move action type to illustrate the evaluation process. Fig. 5 presents the clustering evaluation metrics for both bot and human sessions across a range of K values. For bot sessions, the silhouette score peaked at $K = 15$–18, suggesting that the data exhibits strong cohesion and separation at these cluster counts. The Davies-Bouldin index achieved its lowest (most favorable) values at $K = 8, 10$, and 15, while the Calinski-Harabasz index peaked at $K = 11, 15$, and 16. Based on the consistent performance across all three metrics, we selected $K = 15$ as the optimal value for bots, balancing robustness and stability. For human sessions, the silhouette score was highest for $K = 10$ and 11, with the Davies-Bouldin index again favoring $K = 8, 10$, and 11. While the Calinski-Harabasz index showed strong results for $K = 11$ and 15–17, we chose $K = 10$ to align with the consensus of the other two indices and maintain consistency with behavioral interpretability.

To ensure comprehensive feature augmentation across all behavior types, we repeated this optimization process for each of the four action types—Mouse Move, Click, Scroll, and Keystroke—for both human and bot datasets. The optimal K values used for feature generation are summarized in Table IV.

E. Impact of Multi-Action Integration

In pursuit of *RQ3*, this subsection examines the robustness and generalizability of our models through ablation studies and cross-validation. To evaluate the contribution of each interaction type, we train the ensemble model on different combinations of the four action types using the e-commerce training data and test its performance on both the e-commerce test set and the collected social media site dataset, which was not used during training. This setup evaluates *generalizability across platforms* and robustness to missing interaction

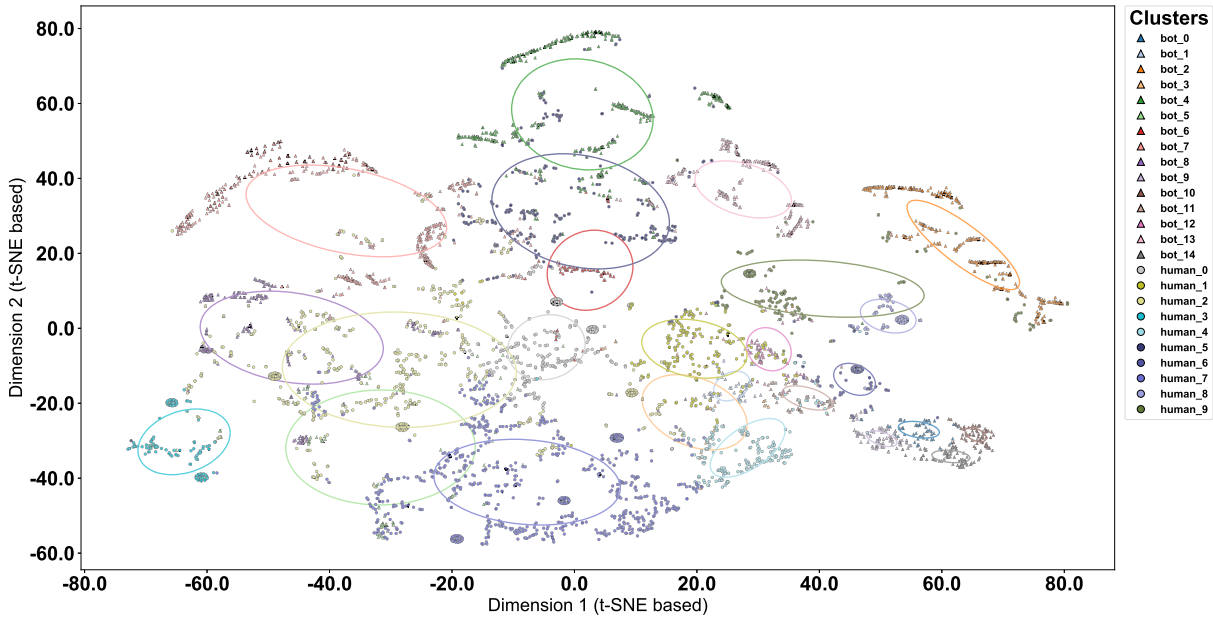

Fig. 4. t-SNE visualization of human and bot clusters based on mouse movement behavior. Human clusters (circles) and bot clusters (triangles) exhibit distinct spatial groupings with visible separation in many regions. Each cluster is outlined with a fitted ellipse representing one standard deviation from the cluster center, capturing the dominant orientation and general spread of the group. This separation supports the intuition that the distance of a point to its nearest human and bot cluster can serve as a discriminative feature. Points that fall closer to bot-dense regions tend to reflect automated behavior, while those closer to human clusters align with natural user interaction patterns.

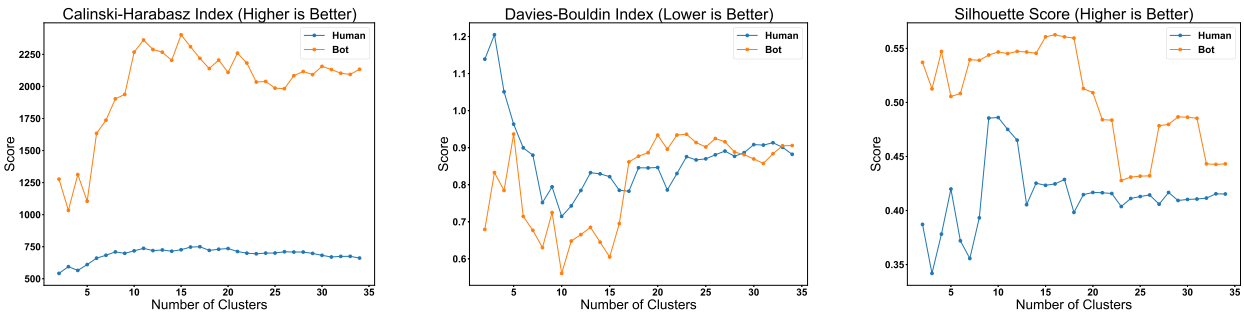

Fig. 5. Optimal cluster count evaluation using three clustering metrics for human and bot sessions based on mouse movement behavior.

TABLE IV
OPTIMAL CLUSTERS FOR MOUSE MOVE, MOUSE CLICK, KEYPRESS AND SCROLLING

Action Type	Category	Optimal Clusters	Silhouette	Davies-Bouldin	Calinski-Harabasz
Mouse Move	Human	10	0.49	0.71	717.61
	Bot	15	0.56	0.605	2403.15
Mouse Click	Human	7	0.47	0.75	312.73
	Bot	5	0.51	0.71	456.19
Key Press	Human	5	0.62	0.59	304.28
	Bot	6	0.58	0.55	387.42
Scrolling	Human	8	0.57	0.61	246.48
	Bot	6	0.62	0.53	328.83

types. Table V presents the accuracy for each subset, where MM, MC, MS, and KP represent Mouse Move, Mouse Click, Mouse Scroll, and Key Press, respectively. The full combination yields the highest accuracy of 98.8% for the e-commerce test dataset and 97.4% for the social site dataset. Interestingly, while MM+MC+KP gives the second-best performance (97.2%) on e-commerce data, MM+MC+MS achieves 96.2% on the social site dataset, suggesting that scrolling may be more discriminative than keystrokes in that context. Among the three-action subsets, excluding keystrokes results in minimal performance loss, indicating their redundancy when other signals are present. The weakest performance is observed with scroll-only interactions. Overall, combining multiple interaction types improves detection accuracy, with mouse movement, especially velocity and path efficiency, emerging as the most discriminative feature.

TABLE V
PERFORMANCE METRICS FOR DIFFERENT ACTION COMBINATIONS

Action Included	Accuracy		Precision		Recall		F1 Score	
	E-comm.	Social	E-comm.	Social	E-comm.	Social	E-comm.	Social
MM	95.1	93.8	96.4	94.9	95.5	93.5	95.9	94.2
MC	85.2	83.9	86.1	84.7	84.3	82.8	85.2	83.7
MS	84.8	86.3	83.7	84.9	85.6	86.0	84.6	85.4
KP	87.9	85.5	87.5	85.6	87.7	85.2	87.6	85.4
MM+MC	95.2	93.7	93.7	92.5	97.4	95.6	95.5	94.0
MM+MS	95.3	94.1	94.6	93.2	95.3	94.0	95.0	93.6
MM+KP	96.5	94.0	96.7	94.2	96.2	94.8	96.4	94.5
MC+MS	86.6	85.7	86.9	85.5	88.1	86.8	86.5	86.1
MC+KP	88.6	85.2	87.2	85.0	89.5	86.8	88.4	85.9
MS+KP	88.2	86.5	88.9	86.3	88.1	86.2	88.5	86.2
MM+MC+MS	96.4	96.2	95.9	95.4	96.3	96.7	96.1	96.0
MM+MC+KP	97.2	95.6	96.3	94.9	98.7	96.8	97.5	95.8
MM+MS+KP	96.8	95.3	93.5	92.1	98.2	96.5	95.8	94.2
MC+MS+KP	90.4	89.7	90.0	88.4	90.2	89.6	90.1	89.0
MM+MC+MS+KP	98.8	97.4	98.5	97.0	99.1	97.7	98.8	97.3

F. Cross-Validation Stability

To evaluate the robustness and generalizability of our models, we performed stratified 10-fold cross-validation on the dataset. In each fold, data was split into 70% for training, 20% for validation, and 10% for testing. Session data from the same participant or bot script was kept separate across splits to prevent data leakage and ensure evaluation integrity. We achieved an average accuracy of 98.1%, precision of 97.9%, recall of 98.4%, and F1-score of 98.0% across the 10 folds, demonstrating consistent, reliable detection performance with minimal variation across user and bot interaction distributions.

G. Overhead Analysis

To ensure the feasibility of real-time deployment, we evaluate the overhead of our behavioral bot detection system across three key dimensions: computational resource consumption, processing latency, and bandwidth usage.

Computational Resource: The system is lightweight, with minimal CPU usage on both client and server. The frontend logger runs asynchronously, with no impact on user interaction. On the backend, the pipeline involves simple statistical feature extraction, centroid distance calculation, and classification using lightweight models (SVM, Random Forest, XGBoost). These operations are computationally cheap and require no GPU acceleration. To validate this, the backend was deployed on Google Cloud using an e2-micro instance with 1 vCPU (shared) and 1 GB RAM. Despite limited resources, the system maintained efficient performance, confirming its suitability for resource-constrained environments.

Latency: Client-side interactions are transmitted in 5-second windows, collected, and compressed into batches for backend processing. This design lets the detection system analyze data incrementally rather than waiting for an entire session. For example, with a 20-second window, the system considers only the most recent 20 seconds of activity—a much smaller slice than a full 5-minute session—and can produce a detection verdict in roughly 5–12 ms depending on interaction frequency. In contrast, processing a complete 5-minute session, including feature extraction, clustering-based augmentation, and ensemble classification, takes about 120–150 ms, still enabling near real-time detection.

Bandwidth (Log Transmission & Processing): Raw UI events (mouse movements, clicks, scrolls, keystrokes) are compressed client-side using efficient compaction and buffering strategies. For a typical 5-minute browsing session, the total data sent is approximately 100–120 KB, resulting in an average bandwidth usage of < 0.5 KB/s. This minimal bandwidth footprint enables scalable deployment, even on bandwidth-constrained networks.

V. DISCUSSION

Our detection system achieves high accuracy in distinguishing advanced OpenWPM based web bots from human users. It effectively utilizes a wide range of behavioral data, including mouse movements, clicks, keypresses, and scrolling, either individually or in combination, based on information at hand. This adaptability allows the model to be deployed across various types of websites.

Handling Partial Interactions and Performance Trade-offs: A key advantage of our approach lies in its resilience to missing data. Since each interaction type is modeled independently, the system can still perform well even when certain input sources are unavailable. For example, our model still achieves F1 scores above 90% without using keystroke action types data. This modularity also supports flexible deployment under different resource constraints. While combining all four interaction types with the meta-classifier yields the highest accuracy of 98.8%, relying solely on mouse movement eliminates the need for meta-level fusion, reducing overhead with only a modest drop to 95.1% accuracy (3.7%). This trade-off allows the system to balance performance and efficiency, making it adaptable to both high-performance and lightweight environments.

Cross-Site Adaptability and Integration: We developed and tested our system using a purpose-built website that incorporates a variety of interaction patterns commonly found across domains such as e-commerce, social media, and news platforms. A lightweight JavaScript frontend passively captures user events with minimal impact, while the backend pipeline including the clustering and ensemble classification can be readily integrated into existing web analytics or security infrastructures. The detection process runs in real time and remains transparent to users.

Limitations and Future Directions: Our method has certain limitations. We did not incorporate browser fingerprinting, as advanced bots can effectively mimic legitimate fingerprints, rendering it unnecessary for our purpose. However, for bots that have difficulty mimicking fingerprints, such as those using headless browsers, fingerprinting could provide an additional layer of detection.

Another limitation concerns scalability in high-traffic environments. While our system achieves near real-time detection with latencies of 5–12 ms for incremental windows and 120–150 ms for full-session analysis, these measurements were obtained on a lower-end single-core e2-micro instance, which is not suitable for high-traffic workloads. For high-traffic scenarios, it is necessary to use powerful multi-core vCPUs, which will automatically improve the performance of our tree-based model. Additionally, leveraging multi-threading and parallel processing via Python's *multiprocessing* module can further distribute tasks across multiple worker processes. A thorough scalability study under varying hardware and traffic conditions remains an important direction for future work.

Future work should focus on expanding the dataset to better generalize browsing behaviors across different user groups and environments. Furthermore, integrating all browsing behavioral

features with HTTP based features [8], [21] could enhance detection capabilities. While previous studies have explored combining only mouse movement behavior with web log analysis [21], but they rely on heuristic based thresholding or simple averaging of scores from separate modules, which may limit effectiveness. Further research is needed to develop and evaluate more advanced integration techniques. Another important direction is benchmarking our framework against commercial bot management solutions such as Cloudflare Bot Management [57] and Imperva [58]. While these tools are widely deployed in practice, they are proprietary and operate as black-box services, which makes direct comparisons challenging. Our current work therefore focused on developing a transparent, research-oriented framework. In the future, indirect benchmarking approaches can be explored, for instance by deploying our system in environments where such commercial tools are already active, and by studying how our behavioral modules can complement these existing solutions.

VI. Conclusion

In this paper, we present a robust and modular behavioral analysis framework for detecting stealthy web bots, particularly those using OpenWPM. By leveraging a combination of mouse movements, clicks, scrolling, and keystrokes, our system effectively differentiates between human and bot interactions. The integration of clustering-based feature augmentation with ensemble classifiers yields a high detection accuracy of 98.8%. Notably, the system remains resilient in the presence of partial or missing action data, ensuring adaptability across diverse web environments. Our findings demonstrate the potential of browsing behavior as a powerful tool to counter sophisticated human-mimicking web automation.

References

[1] Imperva, "2024 imperva bad bot report," Imperva, Tech. Rep., 2024, Accessed: 13, 2024. [Online]. Available: https://www.imperva.com/resources/resource-library/reports/2024-bad-bot-report

[2] Selenium, Seleniumhq browser automation, 2025. Accessed: Jun. 01, 2025. [Online]. Available: https://www.selenium.dev/

[3] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1388–1401.

[4] Puppeteer, Puppeteer: Headless browser automation, 2025. Accessed: Jun. 01, 2025. [Online]. Available: https://pptr.dev/

[5] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 1589–1605.

[6] M. Aljabri and R. M. A. Mohammad, "Click fraud detection for online advertising using machine learning," *Egyptian Informat. J.*, vol. 24, no. 2, pp. 341–350, 2023.

[7] P. Kumari and A. K. Jain, "A comprehensive study of DDoS attacks over IoT network and their countermeasures," *Comput. Secur.*, vol. 127, 2023, Art. no. 103096.

[8] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and Y. Kompatsiaris, "Towards a framework for detecting advanced web bots," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, New York, NY, USA, 2019, doi: 10.1145/3339252.3339267.

[9] WebTAP, "Webtap–web transparency and accountability platform," 2025. Accessed: Jun. 01, 2025. [Online]. Available: https://webtap.ai/

[10] S. Rovetta, A. Cabri, F. Masulli, and G. Suchacka, "Bot or not? a case study on bot recognition from web session logs," in *Quantifying and Processing Biomedical and Behavioral Signals 27*. Berlin, Germany: Springer, 2019, pp. 197–206.

[11] T. Tanaka, H. Niibori, L. Shiyingxue, S. Nomura, H. Kawashima, and K. Tsuda, "Bot detection model using user agent and user behavior for web log analysis," *Procedia Comput. Sci.*, vol. 176, pp. 1621–1625, 2020.

[12] M. Zabihimayvan, R. Sadeghi, H. N. Rude, and D. Doran, "A soft computing approach for benign and malicious web robot detection," *Expert Syst. Appl.*, vol. 87, pp. 129–140, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417304116

[13] D. Doran and S. S. Gokhale, "An integrated method for real time and offline web robot detection," *Expert Syst.*, vol. 33, no. 6, pp. 592–606, 2016.

[14] G. Suchacka and M. Sobkow, "Detection of Internet robots using a Bayesian approach," in *Proc. IEEE 2nd Int. Conf. Cybern.*, 2015, pp. 365–370.

[15] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc, "FP-Crawlers: Studying the resilience of browser fingerprinting to block Crawlers," in *Proc. NDSS Workshop Measurements, Attacks, Defenses Web*, 2020, doi: 10.14722/madweb.2020.23010.

[16] H. Jonker, B. Krumnow, and G. Vlot, "Fingerprint surface-based detection of web bot detectors," in *Proc. 24th Eur. Symp. Res. Comput. Secur.*, Luxembourg, Springer, 2019, pp. 586–605.

[17] B. Krumnow, H. Jonker, and S. Karsch, "Analysing and strengthening openwpm's reliability," 2022, *arXiv:2205.08890*.

[18] Y. Luo, G. She, P. Cheng, and Y. Xiong, "BotGraph: Web bot detection based on sitemap," 2019, *arXiv:1903.08074*.

[19] J. Kadel, A. See, R. Sinha, and M. Fischer, "BOTracle: A framework for discriminating bots and humans," in *Proc. Eur. Symp. Res. Comput. Secur.*, Springer, 2024, pp. 49–67.

[20] N. Afanaseva and P. Lozhnikov, "Bot detection using mouse movements," in *Proc. Dyn. Syst., Mechanisms Mach.*, 2023, pp. 1–4.

[21] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and I. Kompatsiaris, "Detection of advanced web bots by combining web logs with mouse behavioural biometrics," *Digit. Threats: Res. Pract.*, vol. 2, no. 3, pp. 1–26, 2021.

[22] Z. Chu, S. Gianvecchio, and H. Wang, "Bot or human? a behavior-based online bot detection system," in *From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday*. Berlin, Germany: Springer, 2018, pp. 432–449.

[23] A. Wei, Y. Zhao, and Z. Cai, "A deep learning approach to web bot detection using mouse behavioral biometrics," in *Proc. Biometric Recognition: 14th Chin. Conf.*, Zhuzhou, China, Springer, 2019, pp. 388–395.

[24] I. P. Malashin, V. S. Tynchenko, A. P. Gantimurov, V. A. Neluyb, and A. S. Borodulin, "Detecting of robotic imitation of human on-the-website activity with advanced vector analysis and fractional derivatives," *IEEE Access*, vol. 12, pp. 56707–56718, 2024.

[25] A. Acien, A. Morales, J. Fierrez, and R. Vera-Rodriguez, "Becaptcha-mouse: Synthetic mouse trajectories and improved bot detection," *Pattern Recognit.*, vol. 127, 2022, Art. no. 108643.

[26] H. Niu, J. Chen, Z. Zhang, and Z. Cai, "Mouse dynamics based bot detection using sequence learning," in *Proc. Biometric Recognition: 15th Chin. Conf.*, Shanghai, China, Springer, 2021, pp. 49–56.

[27] H. Niu, A. Wei, Y. Song, and Z. Cai, "Exploring visual representations of computer mouse movements for bot detection using deep learning approaches," *Expert Syst. Appl.*, vol. 229, 2023, Art. no. 120225.

[28] S. Sadeghpour and N. Vlajic, "Remouse Dataset: On the efficacy of measuring the similarity of human-generated trajectories for the detection of session-replay bots," *J. Cybersecurity Privacy*, vol. 3, no. 1, pp. 95–117, 2023.

[29] W. Kaixin, L. Hongri, W. Bailing, H. Shujie, and S. Jia, "A user authentication and identification model based on mouse dynamics," in *Proc. 6th Int. Conf. Inf. Eng.*, 2017, pp. 1–6.

[30] K. L. Meidenbauer, T. Niu, K. W. Choe, A. J. Stier, and M. G. Berman, "Mouse movements reflect personality traits and task attentiveness in online experiments," *J. Pers.*, vol. 91, no. 2, pp. 413–425, 2023.

[31] S. Dhamnani, R. Sinha, V. Vinay, L. Kumari, and M. Savova, "Botcha: Detecting malicious non-human traffic in the wild," 2021, *arXiv:2103.01428*.

[32] S. Feng, H. Wan, N. Wang, and M. Luo, "BotRGCN: Twitter bot detection with relational graph convolutional networks," in *Proc. 2021 IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2021, pp. 236–239.

[33] Y. Li et al., "Relevance-aware anomalous users detection in social network via graph neural network," in *Proc. Int. Joint Conf. Neural Netw.*, 2021, pp. 1–8.

[34] K. Wang, X. Wang, K. Chen, Z. Wang, and K. Zheng, "Multi-stage self-training social bot detection based on graph neural network," *Eng. Appl. Artif. Intell.*, vol. 152, 2025, Art. no. 110816.

[35] F. Liu, B. Jiang, and R. Ma, "Multiscale group behavior perception for intelligent social bot detection," *Expert Syst. Appl.*, vol. 297, 2025, Art. no. 129315.
[36] Q. Lin and J. Zhou, "BotBR: Social bot detection with balanced feature fusion and reliability-enhanced graph learning," in *Proc. 48th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2025, pp. 392–402.
[37] M. Gao, Q. Duan, B. Liu, Y. Xiao, X. Wang, and Y. Chen, "Higher-order information matters: A representation learning approach for social bot detection," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2025, pp. 675–685.
[38] S. Feng, Z. Tan, R. Li, and M. Luo, "Heterogeneity-aware Twitter bot detection with relational graph transformers," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 3977–3985.
[39] A. Sallah et al., "Fine-tuned understanding: Enhancing social bot detection with transformer-based classification," *IEEE Access*, vol. 12, pp. 118250–118269, 2024.
[40] J. Luo and C. Jin, "Fusing content and social relationships: A multi-modal heterogeneous graph transformer approach for social bot detection," *EPJ Data Sci.*, vol. 14, no. 1, 2025, Art. no. 68.
[41] Z. Chu, S. Gianvecchio, A. Koehl, H. Wang, and S. Jajodia, "Blog or block: Detecting blog bots through behavioral biometrics," *Comput. Netw.*, vol. 57, no. 3, pp. 634–646, 2013.
[42] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet Detection Based on Network Behavior," in *Botnet Detection: Countering the Largest Security Threat*. Berlin, Germany: Springer, 2008, pp. 1–24.
[43] F. Alejandre, N. Cortés, and E. Aguirre, "Botnet detection using clustering algorithms," *Res. Comput. Sci.*, vol. 118, pp. 65–75, 2016.
[44] S. S. Ahmad, M. D. Dar, M. F. Zaffar, N. Vallina-Rodriguez, and R. Nithyanand, "Apophanies or epiphanies? how crawlers impact our understanding of the web," in *Proc. Web Conf.*, 2020, pp. 271–280.
[45] M. Schwarz, F. Lackner, and D. Gruss, "Javascript template attacks: Automatically inferring host information for targeted exploits," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, doi: 10.14722/ndss.2019.23155.
[46] Y. Yang et al., "Rosgas: Adaptive social bot detection with reinforced self-supervised gnn architecture search," *ACM Trans. Web*, vol. 17, no. 3, pp. 1–31, 2023.
[47] S. A. Alhosseini, R. Bin, P. Tareaf Najafi, and C. Meinel, "Detect me if you can: Spam bot detection using inductive representation learning," in *Proc. World Wide Web Conf.*, 2019, pp. 148–153.
[48] A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "A graph-based machine learning approach for bot detection," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage.*, 2019, pp. 144–152.
[49] W. Wu, J. Alvarez, C. Liu, and H.-M. Sun, "Bot detection using unsupervised machine learning," *Microsystem Technol.*, vol. 24, no. 1, pp. 209–217, 2018.
[50] Q. Guo, H. Xie, Y. Li, W. Ma, and C. Zhang, "Social bots detection via fusing BERT and graph convolutional networks," *Symmetry*, vol. 14, no. 1, 2021, Art. no. 30.
[51] H. G. Shah and H. Joshi, "Spam bot detection on Twitter platform using positional attention based dense convolutional neural network," *Appl. Soft Comput.*, vol. 184, 2025, Art. no. 113725.
[52] L. A. Leiva and J. Huang, "Building a better mousetrap: Compressing mouse cursor activity for web analytics," *Inf. Process. Manage.*, vol. 51, no. 2, pp. 114–129, 2015.
[53] B. Aldrich, Y. Liu, M. Almousa, and M. Anwar, "Translating keystroke and mouse dynamics data to classify human mood," in *Proc. 5th Int. Conf. Transdisciplinary AI*, 2023, pp. 79–86.
[54] A. Barkworth, R. Tabassum, and A. H. Lashkari, "Detecting IMAP credential stuffing bots using behavioural biometrics," in *Proc. 12th Int. Conf. Commun. Netw. Secur.*, 2022, pp. 7–15.
[55] ihtasham42, Postit: A social media app, 2025. Accessed: Oct. 01, 2025. [Online]. Available: https://github.com/ihtasham42/social-media-app?tab=readme-ov-file
[56] D. Goßen, H. Jonker, S. Karsch, B. Krumnow, and D. Roefs, "HLISA: Towards a more reliable measurement tool," in *Proc. 21st ACM Internet Meas. Conf.*, 2021, pp. 380–389.
[57] Cloudflare, "Cloudflare's bot management," 1999. [Online]. Available: https://developers.cloudflare.com/bots/
[58] imperva, "Imperva advanced protection," 1999. [Online]. Available: https://www.imperva.com/products/advanced-bot-protection-management/

Shakil Ahmed received the BSc degree in computer science and engineering from the Bangladesh University of Engineering and Technology (BUET), in 2023. He is currently working as a software development engineer with IQVIA, Dhaka, Bangladesh, and is preparing to pursue PhD opportunities beginning in Fall'26. He has focused his work on computer vision, natural language processing, and security, with a focus on applying AI methods to real-world, security-relevant problems. His research interests span machine learning, NLP, and the application of LLMs in security contexts. He is also interested in generative models such as GANs for both vision and security-related applications, as well as exploring how reinforcement learning can contribute to the design of privacy-preserving and adaptive secure systems.

Md. Mahedi Hasan Rigan received the BSc degree in computer science and engineering from the Bangladesh University of Engineering and Technology (BUET), in 2023. He is currently working as a software engineer with Samsung R&D Institute, Dhaka, Bangladesh. He will begin his PhD degree in Fall 2025 with the Department of Computer Science, University of Texas at Arlington, USA. His primary research interests lie in computer security, computer vision, and machine learning, with a specific focus on exploring vulnerabilities in multi-modal systems, attacks on large language model (LLM) agents, and developing privacy-preserving techniques in machine learning.

Md. Shohrab Hossain (Member, IEEE) received the BSc and MSc degrees in computer science and engineering from the Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2003 and 2007, respectively, and the PhD degree from the School of Computer Science, University of Oklahoma, Norman, OK, USA in December 2012. During his PhD, he worked under NASA-funded projects related to survivability, scalability, and security of space networks. He is currently serving as a professor with the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. His research interests include cyber security, mobile malware detections, software-defined networking (SDN), security of mobile and ad hoc networks, and the Internet of Things. He has published more than 90 technical research papers in leading journals and conferences including the *Journal of Computers & Security*, *Ad Hoc Networks*, *IEEE Access*, *Journal of Network and Computer Applications*, *IEEE Transactions of Mobile Computing*, *Wireless Personal Communication*, *PLoS One*, IEEE GLOBECOM, IEEE ICC, IEEE MILCOM, IEEE WCNC, IEEE HPCC, etc.

Anupam Das (Senior Member, IEEE) received the BS and MS degrees in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2008 and 2010, respectively, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign (UIUC), Illinois, USA, in 2016. He was a recipient of a Fulbright Science and Technology fellowship during his PhD degree. He was also a postdoctoral fellow with the School of Computer Science at Carnegie Mellon University (CMU) from 2016 to 2018. He is currently an assistant professor with the Computer Science Department, North Carolina State University (NCSU), Raleigh, North Carolina, USA. His research interests lie in security and privacy, with a special focus on designing secure and privacy-preserving technologies. He has published more than 60 technical research papers in leading journals and conferences. He has been awarded more than U.S. $2.5 million in research grants from the U.S. National Science Foundation (NSF), NCSU, Meta, and Amazon. He has also received two ACM Distinguished Paper Awards (ASIACCS 2014, MMSys 2017). His projects have been covered by media outlets such as Wired, Forbes, ZDNet, MotherBoard, and FastCompany.