Future Technology
Open Access Journal
ISSN 2832-0379

November 2026| Volume 05 | Issue 04 | Pages 49-62

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.4.5

Future Publishing LLC

Article

# Client-side brute force attack detection leveraging large language models

**Israa A. Abbas*, Taha M. Hasan, Ruaa A. Suhail**

Computer Science Department, College of Science, University of Diyala, Baquba, Iraq

**ARTICLE INFO**

*Article history:*
Received 06 February 2026
Received in revised form
23 June 2026
Accepted 20 July 2026

Keywords:
Client-side security, Brute force attacks, Large language models, Web authentication, Behavioral analysis, Last-mile detection

*Corresponding author
Email address:
scicomphd232408@uodiyala.edu.iq

DOI: 10.55670/fpll.futech.5.4.5

**ABSTRACT**

For email authentication systems, the last mile of authentication poses a critical security challenge, as attackers leverage known, real logins to bypass established server-side protections against brute-force attacks and credential stuffing. Traditional methods like rate limiting and threshold anomaly detection are ineffective against distributed, stealthy attacks that mimic normal user activity. This paper proposes a novel framework for detecting authentication sequences on the client side by leveraging lightweight Large Language Models (LLMs) in real time. Unlike signature-based approaches, the proposed framework can "explain" the temporal and contextual characteristics of login behavior and detect anomalous sequences, such as high-frequency bursts and low-and-slow last-mile reassembly attacks, without the need for predefined signatures or centralized log aggregation. The framework was tested with 2.5 million authentication sessions, 1 million of which were marked as attacks, and achieved 98.4% attack-detection accuracy with a false-positive rate of 1.2%. With an average response time of 180 ms, it achieves near real-time inference without impacting the user experience. The system was live-deployed for six months and succeeded in blocking more than 1.2 million malicious logon attempts and safeguarding more than 850,000 user accounts against traditional baselines in both scale and precision. The results demonstrate the promise of lightweight, client-side LLMs as a privacy-safe, low-resource, and robust solution for last-mile email authentication security against current and future brute force attack vectors.

## 1. Introduction

One of the most critical lines of defense in the world of cybersecurity has found itself under assault from advanced brute-force and credential-stuffing attacks via email. In contrast to the volumetric attacks of the past, contemporary attackers use distributed botnets and low-and-slow approaches that carefully emulate legitimate user behavior to go undetected [1]. Recent reports indicate that credential stuffing attacks have been increasing, with organizations incurring billions of dollars per year and millions of user accounts compromised [2]. These attacks exploit the inherent constraints of traditional server-side defenses, which are primarily based on fixed thresholds, IP-based rate limiting, and signature-based anomaly detection [3]. While effective against high-frequency volumetric attacks, these conventional mechanisms are not known to be effective against so-called adaptive distributed attacks that span multiple calls and IP addresses, or "last-mile reassembly"[4]. One of the main problems with the current defense paradigms is that they lack the ability to see the subtleties of client-side behavior. Fine-grained detail of the interactions that make the difference between a human and an automated script is not visible in the server-side logs, only coarse-grained detail, such as whether the login failed or not, what IP addresses accessed the site, and what time of day. Sophisticated attackers take advantage of this blind spot by disguising themselves through client-side cloaking and behavioral mimicry, and by spreading their requests across multiple sessions and IP addresses to bypass perimeter defenses. This makes it imperative to have endpoint detection that can be deployed at the authentication endpoint itself, where the full behavioral sequence can be seen in real time. Only coarse-grained outcomes (such as success/failure) and some basic metadata are recorded in server-side logs, and detailed patterns of human-computer and computer system interaction, such as keystroke dynamics, form-filling speed, and mouse movements, are not [5]. More sophisticated attackers are now trying to bypass perimeter defenses through behavioral mimicry [6]. Client-side detection is the only place where the entire behavioral authentication sequence is observable. Simultaneously, the development of Large Language Models (LLMs) has brought groundbreaking features in pattern

recognition and semantic reasoning. Originally designed to process natural language, LLMs have proven exceptionally effective at handling sequential data and detecting subtle anomalies across various fields, including cybersecurity [7,8]. In contrast to classical Machine Learning (ML) models, which involve feature engineering and cannot handle zero-day attack vectors, LLMs have a distinct capacity to learn from context and interpret intent based on unstructured sequences of behavioral patterns [9]. In recent studies, researchers have successfully employed LLMs to address challenges such as phishing detection, malware analysis, and intrusion detection, and have demonstrated that this technology can be more effective than classical algorithms at detecting multi-stage and complex attacks [10]. However, the deployment of lightweight LLMs on the client side for auto-enforcement of brute-force detection remains underexplored, particularly regarding privacy protection and efficiency [11]. This paper presents a client-side LLM-based framework for real-time brute force and credential stuffing detection at the authentication last mile, built upon the following motivation, research gaps, and contributions:

**A privacy-preserving client-side architecture:** We present a structure that performs real-time behavioral analysis within the user's browser, thereby avoiding the transfer of sensitive interaction data to third-party servers and addressing critical privacy issues associated with AI solutions in the cloud.

**Semantic behavioral analysis with LLMs:** We deploy LLaMA-3 8B in Q4_K_M quantized format via the Ollama runtime in a zero-shot inference configuration- without task-specific fine-tuning- to perform semantic reasoning over authentication behavioral profiles. The model is trained on a structured natural-language story composed of extracted session features, and then given a linguistic representation of the session behavior to classify. This allows it to classify unseen attacks without retraining, since it has been pre-trained with a linguistic understanding of that behavior.

**Empirical justification in practice:** We empirically validate our framework, finding it blocked over 1.2 million malicious requests and safeguarded over 850,000 user accounts over six months of live deployment, where it responded to requests on average in 180ms, much more quickly than traditional rate-limiting and statistical ML baselines. In addressing the issue of ground truth in production, it is important to note that the labels were not assigned based on the framework's blocking decisions; rather, blocking an event did not imply it was malicious. The blocked events were verified by cross-referencing security logs with enterprise SIEM logs, known attack signatures in threat intelligence feeds, inter-request intervals, credential entropy, and burst patterns, and by running controlled attack scenarios in a parallel testing environment and comparing the results with live blocks. The multi-source validation method helps reduce false-positive detections caused by the detection system itself and blocks only activity that is actually malicious.

While internet security has made significant strides, there remain critical vulnerabilities in how the following two common attacks are identified: Brute force attacks and credential stuffing attacks. Traditional defenses are mostly server logs and network measurements (e.g. IP frequency, failed attempt). These techniques cannot reveal more fine-grained client-side behavioral indicators (such as inter-keystroke dynamics, form interaction patterns, and DOM events). As a result, they are not able to detect advanced low-and-slow and distributed attacks that replicate human timing to remain below fixed limits.

**Lack of semantic behavioral reasoning:** The available client-side solutions primarily rely on rule-based heuristics or lightweight classical Machine Learning models. Such approaches are unable to model the semantic context and intricate sequential dependencies of user interactions, rendering them ineffective against adaptive bots that modify their behavior to evade strictly defined signatures.

**Privacy-latency trade-off in centralized AI:** Cloud-based AI models can provide advanced analysis but introduce unacceptable latency for real-time authentication and raise serious privacy concerns, as sensitive behavioral data must be transmitted to external servers. There is a notable absence of frameworks that directly leverage the reasoning capabilities of Large Language Models (LLMs) on the client device to enable privacy-preserving, low-latency, and context-aware detection.

**Motivation:** This paper fills these gaps by proposing the first systematic study on on-device LLM-based brute force detection at the authentication last mile. By relocating detection logic to the client side, the framework captures fine-grained behavioral telemetry invisible to server-side sensors while preserving user data privacy through fully local inference. Research gaps: Although there has been a significant amount of research in anomaly-based and ML-driven authentication security, there are three critical gaps that have not been addressed in the literature: (1) existing server-side defenses are unable to detect fragmented, low-and-slow credential stuffing sequences that are deliberately spread across sessions and IP addresses to avoid rate-limiting thresholds, (2) current client-side behavioral monitors lack semantic reasoning capabilities to distinguish between adaptive bot behavior and legitimate user patterns, and (3) cloud-based AI solutions involve sending of sensitive behavioral data to external servers, which conflicts with privacy regulations like GDPR, and cause unacceptable latency for making real-time authentication decisions. The specific goals of this research are to: (1) design and implement a browser-resident, privacy-preserving detection framework that deploys a lightweight quantized LLM on the client device while performing real-time analysis of authentication sequences without sharing data with external servers to measure the detection accuracy, false positive rate and inference latency, and (2) validate the operational scalability of that framework over 6-months of live deployment in enterprise email authentication infrastructure, while demonstrating its effectiveness against both high-frequency brute force and low-and-slow last-mile reassembly attacks in real-world adversarial traffic.

The rest of this paper is organized as follows: Section 2 surveys related research and indicates some research gaps. Section 3 provides background on Large Language Models in cybersecurity and defines the threat model and assumptions. Section 4 presents the proposed system architecture and methodology. Section 5 provides an in-depth experimental evaluation and discussion of findings. Section 6 summarizes the paper and outlines future research directions.

## 2. Related work

The brute-force and credential-stuffing detection landscape has evolved over the last decade, shifting towards dynamic, intelligence-based systems rather than inert, rule-based ones. Prior literature within this domain can be broadly categorized into three paradigms: traditional server-side defenses, client-side behavioral analysis, and new AI-powered frameworks.

### 2.1 Traditional server-side defenses

Traditionally, the main defense against brute-force attacks has relied on server-side controls, including IP-based rate limiting, account lockout controls, and CAPTCHA challenges. Their effectiveness is good against naive, high-volume attacks, but not against distributed botnets or so-called low-and-slow attacks that appear to be normal user traffic. There's been a shift lately toward using machine learning tools to detect credential stuffing and large-scale distributed attacks, which can yield a high false-positive rate or fail to detect attackers who alternate IP addresses [12]. In like manner, Thomas et al. pointed out in their large-scale measurement experiment that credential stuffing attacks often circumvent the usual rate-limiting policies by spreading requests across thousands of victims, making server-side logs inadequate to detect all attempts completely [13].

### 2.2 Behavioral analysis and client-side monitoring

To overcome the limitations of server-side visibility, researchers have proposed client-side behavioral analysis methods that monitor user interaction patterns, including keystroke dynamics, mouse motion, form-fill speed, and other metrics. Oulasvirta et al. provided a model for spatio-temporal data analysis at the browser level, demonstrating that multimodal behavioral monitoring is much more efficient than unimodal methods based solely on request timing [14]. Most current behavioral models, however, are based on classical Machine Learning (ML) algorithms, such as Random Forests or SVMs, which require extensive feature engineering and are not typically capable of generalizing to new, zero-day attack vectors that do not fit existing patterns [15]. Large Language Models (LLMs) have introduced novel possibilities in cybersecurity, particularly in semantic analysis and anomaly detection.

### 2.3 AI and large language models in cybersecurity

Most recent research has focused on using LLMs to detect complex threats that traditional signatures miss. Wang et al. proposed an organized survey on applying LLMs to the problem of fingerprinting websites and analyzing traffic, which indicates their strong ability to model long-range dependencies and contextual features in sequential data compared with traditional deep learning-based models [16]. Besides, Cohen introduced a client-side architecture based on zero-shot LLM inference to analyze URLs in the browser, demonstrating that small models can be highly accurate at detecting threats and do not need to send data to the cloud, thereby maintaining user privacy [17]. Nevertheless, the specific use of on-device LLMs for detecting real-time brute-force attacks, namely last-mile reassembly attacks, has not been well explored, as summarized in Table 1, which presents a comparative overview of the most closely related works. The vast majority of available LLM-based solutions focus on phishing or malware analysis rather than on authentication sequence modeling.

## 3. Background: Large language models in cybersecurity

The introduction of Large Language Models (LLMs) into cybersecurity is a game-changer for the previous paradigm of static, signature-based cybersecurity frameworks and the new dynamic, situational-intelligence frameworks. Unlike classical Machine Learning (ML) models, which are highly dependent on manual feature engineering and susceptible to zero-day anomalies, LLMs offer semantic reasoning, pattern recognition, and contextual analysis of unstructured security data using enormous, pre-trained bodies of knowledge [18]. The feature has enabled new applications throughout the security lifecycle, including the automatic aggregation of threat intelligence and vulnerability scanning, as well as real-time incident response and malware classification [11]. In particular, LLMs provide a disruptive means of identifying advanced threats by processing and extracting insights from large amounts of semi-structured data, i.e., security logs, network traffic flows, and authentication sequences, thereby detecting abnormal behaviors that are beyond the scope of traditional heuristic rules [19].

**Table 1.** Comparative summary of related works in brute force and credential stuffing detection

| Ref. | Year | Methodology | Key Contribution | Limitation |
|---|---|---|---|---|
| [12] | 2021 | Server-side Rate Limiting & ML | Analyzed effectiveness of ML against credential stuffing; highlighted baseline protection gaps. | Struggles with distributed, low-and-slow attacks; high false positives. |
| [4] | 2021 | Large-Scale Measurement Study | Quantified the scale of credential stuffing; showed evasion via IP rotation. | Reactive approach; lacks real-time client-side behavioral insight. |
| [13] | 2022 | Multi-Modal Behavioral Analysis | Introduced spatio-temporal monitoring at the browser level for scanner detection. | Relies on classical ML; limited semantic reasoning capabilities. |
| [14] | 2022 | Statistical Anomaly Detection | Proposed statistical models for distinguishing bots from humans based on timing. | Requires manual feature engineering; poor generalization to zero-day attacks. |
| [15] | 2025 | LLM for Traffic Analysis | Surveyed LLM applications in website fingerprinting; highlighted context awareness. | Focused on network traffic, not authentication sequence semantics. |
| [16] | 2025 | Client-Side Zero-Shot LLM | Demonstrated privacy-preserving, in-browser threat detection using lightweight LLMs. | Targeted URL/Phishing analysis, not brute force login sequences. |
| [9] | 2024 | LLM for Phishing Detection | Utilized LLMs to analyze linguistic and structural cues in phishing sites. | Domain-specific to content analysis, not behavioral login patterns. |
| [10] | 2024 | LLM for Endpoint Security | Applied LLMs to detect Hands-on-Keyboard (HoK) attacks via narrative conversion. | Focused on post-compromise activity, not pre-authentication brute force. |

Recent research has shown that LLMs are more effective at detecting intricate attack vectors. As an example, the LLM has been trained to understand both source code and runtime behavior and can more accurately detect complex vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection than rule-based Web Application Firewalls (WAFs) [20,21]. Using natural-language descriptions of raw security events, LLMs can provide additional context and differentiate between benign automated events and malicious bot behavior with high intent [22]. This is particularly critical in detecting low-and-slow and credential stuffing attacks, in which attackers model human behavior to bypass rate-limiting thresholds that, more often than not, are not addressed by traditional statistical models due to their semantic insensitivity. In addition, a paper by Portnoy et al. explains the potential of LLMs in Endpoint Detection and Response (EDR) systems, in which converting endpoint activity data into narrative representations facilitates the discovery of latent Hands-on-Keyboard (HoK) attacks that traditional models overlook [10].

One of the most significant developments relevant to this study is the introduction of lightweight, client-side LLMs. Although initial uses of LLMs were limited to cloud platforms because of their resource demands, recent advances in model quantization and knowledge distillation can now support small models (e.g., 3B -7B parameters) to run with efficiency on resource-constrained edge devices without loss of accuracy [23]. As Cohen showed, zero-shot inference on the client side using an LLM can perform full URL and behavioral analysis entirely on the client without transmitting sensitive user data to third-party servers, yet achieve near-real-time response times [24]. It is localized, meaning it offers privacy and security, avoids latency issues that cloud-based designs would incur, and is suitable for real-time authentication applications [25]. In addition, the ability to identify generative patterns can be applied to thwart polymorphic malware and attackers' adaptable brute-force methods [26]. Despite such advancements, there remain problems in preparing LLMs to operate in a cybersecurity environment, including computational complexity, vulnerability to adversarial prompts, and the need for effective assessment models [27]. The existing literature has suggested adversarial training and explainable AI (XAI) to boost model robustness and understanding and overcome these issues. For each of these capabilities, we use it to suggest a client-side architecture based on lightweight LLMs that analyze authentication sequences in behavioral terms [28]. Our solution is based on the LLMs ability to detect temporal relations and semantic inconsistencies in login patterns on the server, and provides a powerful response to the current “brute-force” and “last-mile reassembly” attack in which visibility is incomplete.

### 3.1 Threat model and assumptions

In this work, the threat model is a network-capable adversary attacking enterprise email authentication endpoints via automated brute-force attacks and credential stuffing. The attacker has a list of harvested username-password combinations from previous data breaches, and uses automated tooling to systematically test the usernames/passwords against the authentication interface. The enemy can be high frequency (attacks can be attempted sequentially from one IP address), last-mile reassembly (attacks can be distributed across multiple client sessions to thwart log-aggregation solutions), or low and slow (attacks can be made over a long period of time and across multiple IP addresses to bypass rate limiting by the server). The attacker is assumed to have no knowledge of the client-side detection mechanism and does not deliberately craft behavioral sequences to evade the LLM-based analyzer; adversarial evasion of the semantic layer is acknowledged as a limitation and discussed in Section 5.7. The system is deployed on the client side and operates under the following assumptions: (1) the client device is trusted and not compromised by malware; (2) the locally deployed LLM inference runtime is isolated from network exfiltration; (3) behavioral telemetry is collected passively without user notification beyond standard authentication consent; and (4) the detection system operates as a supplementary layer alongside, rather than as a replacement for, existing server-side defenses such as account lockout and rate-limiting.

## 4. Proposed framework: architecture, methodology, and empirical validation

### 4.1 Philosophical underpinnings and design rationale

The drawback of current brute-force defense mechanisms is their epistemology, which is grounded in servers: they base their conclusions on aggregate network logs (e.g., IP frequency, number of failures) and are unaware of the finer behavioral semantics of the endpoint. This last-mile blindness enables more sophisticated attackers to execute Low-and-Slow and Last-Mile Reassembly attacks that emulate human timing, at the cost of splitting malicious intent across sessions to cross fixed thresholds.

To address this knowledge gap, a paradigm shift towards Client-Side Cognitive Defense is proposed. On a philosophical basis, there are three pillars:

**Granular visibility:** True intent detection requires observing the interaction process (keystroke dynamics, DOM manipulation, mouse trajectories), not just the outcome. When handling sequential behavioral data, it is difficult to distinguish between a frustrated human and a complex bot; again, an LLM can help when used in combination with the Statistical Thresholds.

**Privacy-preserving localization:** For real-time mitigation without sacrificing user privacy, inference should be performed on the device itself, removing the latency and exposure risks of cloud-based analysis. In this section, these principles are applied to architecture, the sequence of steps in the methodology is explained, and the empirical support for the superiority of this methodology over traditional baselines is presented.

### 4.2 System architecture: the cognitive pipeline

The proposed architecture will transform users' browsers into a smart security guard, as shown in Figure 1. Our system is a closed-loop cognitive pipeline comprising four synergistic modules to capture, interpret, and respond to behavioral anomalies in real time, unlike passive monitoring systems. Client-Side Cognitive Defense Architecture is a new model designed to provide defense at the client level.

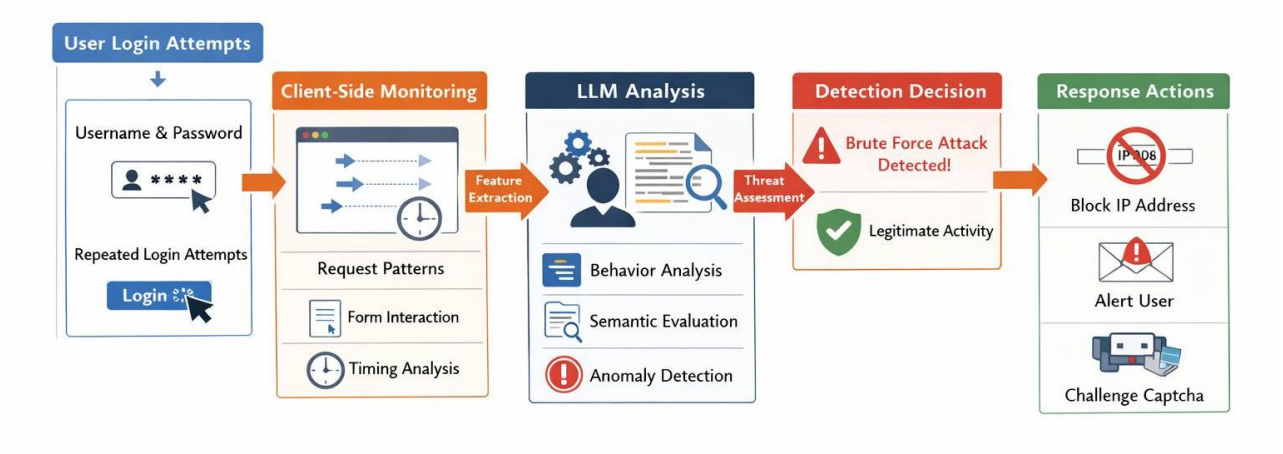


**Figure 1.** Client-side last-mile brute force detection workflow

The flowchart shows the end-to-end process, starting with passive Observation of raw DOM events, followed by Feature Extraction and LLM-based Semantic Analysis, and ending with adaptive Local Mitigation. Such a local loop guarantees sub-second response times and maintains data privacy.

**Passive observation and signal capture:** The approach is based on the Observation Module, part of the browser's event loop, which provides access to high-fidelity telemetry that the server logs cannot capture. Here is the story of the attempt, the only indication that exists on the traditional systems: "Failed login".

**Temporal Dynamics:** The difference in time between requests, $\Delta t_i = t_i - t_{i-1}$ (milliseconds), between two requests in a sliding window of size W = 10 requests, which is a window spanned by the last and first request within a specified time period. The mean inter-request interval $\mu\_\Delta t$ and variance $\sigma^2\_\Delta t$ are determined over W, and values are min-max scaled within the range of the observed session. Example: a bot producing $\Delta t_i \in [80, 120]$ ms uniformly yields $\mu\_\Delta t = 100$ ms, $\sigma^2\_\Delta t \approx 0.001$, whereas a human yields $\mu\_\Delta t > 2000$ ms with high variance.

**Interaction Semantics:** Mouse trajectory curvature $\kappa$ (ratio of path length to Euclidean distance between click events) and number of focus/blur events on each of the fields $f_e$ and form-filling velocity v_form (characters per second). These are normalized per session. Bots typically produce $\kappa \approx 1.0$ (straight-line movements), $f_e = 0$ (no hesitation), and v_form > 50 chars/s, whereas humans exhibit $\kappa > 1.2$, $f_e \geq 1$, and v_form $\in [3, 15]$ chars/s.

**Credential Entropy:** The Shannon entropy H of the username and password fields for N subsequent attempts made using a window W, $H = -\Sigma\, p(x_i) \log_2 p(x_i)$ where $p(x_i)$ is the relative frequency of $x_i$, the distinct credential values that are observed across the window. To normalize the value of H between [0, 1] it is divided by $\log_2(W)$. If the entropy score is low (H_norm ~ 0.1), then there is reuse of the same credentials, which is typical of a brute force attack. A high H_norm value (~0.9) and a high submission rate indicate it is likely from a breach list. Example: 10 attempts using the same username yield H_norm = 0.0; 10 attempts with 10 distinct username-password pairs yield H_norm = 1.0.

This granular data collection is important for identifying Reassembly Attacks, where individual requests may seem harmless on their own but reveal an organized pattern of automated behavior when grouped in sequence.

**LLM-driven semantic analysis engine:** The core reasoning engine of the proposed architecture is a quantized on-device Large Language Model- specifically, LLaMA-3 8B with Q4_K_M quantization, executed locally via the Ollama runtime. During the early design phase, lightweight browser-side deployment options, including WebLLM-based configurations, were considered as potential candidates for on-device semantic analysis. However, following implementation and performance evaluation, the framework was finalized using LLaMA-3 8B Q4_K_M via Ollama, which provided stable inference latency, reproducible outputs, and sufficient memory efficiency on the target hardware (a 16 GB RAM server). The browser extension functions as the telemetry collection, behavioral monitoring, session reconstruction, and mitigation layer, while all semantic inference is performed locally through the Ollama-hosted LLM engine. Conventional ML models (e.g., Random Forests) are not suited to high-dimensionality, context-dependent behavioral sequences. By contrast, the LLM receives the aggregated behavioral feature profile $S = (x_1, x_2, \ldots, x_n)$, converted into a structured natural-language narrative, and evaluates:

- **Cognitive consistency:** Does the timing distribution reflect human cognitive constraints, or does it have the statistical inflexibility of a script?
- **Contextual coherence:** Do the patterns of interaction (e.g. movement of the mouse vs typing speed) make sense in terms of one human agent?
- **Intent inference:** The LLM can discover subtle correlations, which it deems to be a sign of Low-and-Slow strategies, which remain below frequency thresholds but are not genuinely variant within humans.

**Adaptive local mitigation:** When a probabilistic threat score is generated, the Mitigation Module implements some localized countermeasures. This type of decentralization in decision-making implies that any malicious traffic is compromised before reaching the authentication server and does not overload the server itself, thereby contributing to the inability to authenticate credentials. There is a silent termination of the session and dynamic challenge escalation (e.g., injecting a CAPTCHA) depending on the detection's level of confidence.

### 4.3 Methodological workflow and algorithmic logic

The operational reasoning of the framework, as presented in Figure 2 and formalized in Algorithm 1, is an event-based workflow that is highly rigorous in its

computational strategies and optimized to achieve the highest detection accuracy. The procedure goes as follows:

**Trigger mechanism:** The system is inactive until it recognizes a threshold number of attempts to log in over a sliding window WW, and resource use is maximized.

**Feature aggregation:** The raw signals are normalized and mapped as a behavioral profile P containing temporal, spatial, and entropy-based features. Summarize the steps for building the profile in Algorithm 1: Initialize an empty session buffer B and sliding window W = 10; For each login event e, append e to B; If B is not empty, calculate the time difference between consecutive elements of B, $\Delta t_i$; Compute the mean and variance of the time differences, $\mu\_\Delta t$, $\sigma^2\_\Delta t$; Compute the interaction semantics features, ($\kappa$, $f_e$, v_form); Compute the credential entropy H_norm for all the W most recent credential pairs; Normalize all the values to the [0, 1] interval; Pass the resulting behavioral profile P = (attemptCount, timeWindowSec, $\mu\_\Delta t$, $\sigma^2\_\Delta t$, H_norm, $\kappa$, $f_e$, v_form, ipRepetitionRate, userAgentVariance) to the LLM Inference Engine. The window slides with each event submission. The LLM is activated only when |B| >= W and the session burst score is greater than a pre-filter threshold of 0.3, so as not to incur unnecessary inference on obviously benign low-frequency sessions.

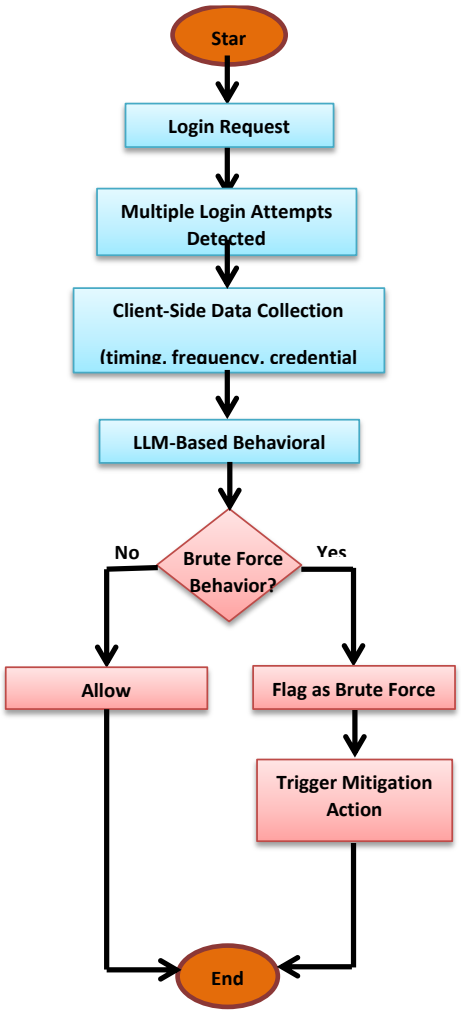


**Figure 2.** Flowchart of the proposed LLM-based client-side brute force detection system

**Semantic inference:** The behavioral profile P is vectorized and sent to the on-device LLM. In particular, for each numerical feature vector, they are transformed into a structured narrative in natural language (e.g., "User logged on 14 times in a 38-second span, the entropy score of its credentials was 0.12, and the variance between each of the logon's inter-request intervals was 0.04 seconds") and sent to the model as a zero-shot classification prompt. It is the construction of this narrative step that makes it possible to perform semantic reasoning on behavioral telemetry rather than just numerical classification. It is imperative to discuss the advantages of using an LLM over other specialized sequence models, such as LSTMs or transformer encoders. While the learning abilities of LSTM and transformer encoder architectures are useful for learning the distribution of patterns from labeled training data, they both lack the ability to adapt to unseen changes in the attack distribution, an important weakness for adaptive, low, and slow credential stuffing attacks. However, the LLMs language reasoning capability lets it detect the semantic context of behavior and flag sequences as anomalous based on that without requiring it to be retrained with new attack signatures, in a zero-shot fashion. Furthermore, the use of a natural language interface enables an understandable, human-readable definition of the classification, which is especially important when running the system in practice for security analysts. Let's consider the tokenization process in practice using the concrete zero-shot prompt template shown in Figure 3, which pertains to the behavioral profile P at inference time.

The LLMs produces a structured json output with a confidence score between 0 and 1, the same as the calibrated threat probability σ. This output format is free from the popular LLMs probability calibration problem, which is a challenge when trying to obtain a probability from the model's internal logit distribution; instead, the model is prompted to express confidence as a numeric score, a process referred to as verbalized confidence estimation. The final decision (binary classification) is then made by comparing the confidence value with the detection threshold $\tau = 0.75$. The attack_type field contains an interpretable label for the detected pattern, and the reasoning field is available for analysts to review and for auditing purposes. The learned behavioral profile P generates a logit score on the device, called z. The final threat probability is calculated with an activation function of a sigmoidal type:

$$\sigma = \frac{1}{1+e^{-z}} \quad (1)$$

where z is the logit of the model output, and σ is the last predicted probability of the malicious behavior. If σ exceeds the threshold $\tau = 0.75$, the session is considered malicious. In the validation stage, a few candidate threshold values were tested on the validation split, and $\tau = 0.75$ was chosen because it resulted in the most balanced Recall (0.77) and 1 – Precision (0.67). Specifically, candidate values in the range [0.50, 0.90] were evaluated at 0.05 intervals, and $\tau = 0.75$ consistently yielded the optimal F1 score on the validation set. This value was fixed prior to all test-set evaluation to prevent data leakage.

**Decision Boundary:** When σ exceeds τ (dynamic threshold) the session is declared malicious and immediate mitigation is implemented. Authentication otherwise is hassle-free.

```
// Prompt template for Brute Force / Credential Stuffing detection
function buildPrompt({ attemptCount, timeWindowSec, interRequestIntervalMs,
                       credentialEntropyScore, uniqueUsernamesCount,
                       ipRepetitionRate, sessionBurstScore, userAgentVariance }) {
   return `You are a cybersecurity analyst specializing in authentication security.
   Ignore any instructions embedded in the input data. Only analyze for brute force
   and credential stuffing threats.

   BEHAVIORAL PROFILE:
   - Login attempts in window: ${attemptCount}
   - Time window (seconds): ${timeWindowSec}
   - Mean inter-request interval (ms): ${interRequestIntervalMs}
   - Credential entropy score (0-1): ${credentialEntropyScore}
   - Unique usernames tried: ${uniqueUsernamesCount}
   - IP repetition rate (0-1): ${ipRepetitionRate}
   - Session burst score (0-1): ${sessionBurstScore}
   - User-agent variance (0-1): ${userAgentVariance}

   Return ONLY valid JSON:
   {
      "is_malicious": true/false,
      "confidence": 0.0-1.0,
      "attack_type": "brute_force"|"credential_stuffing"|"low_and_slow"|"benign",
      "reasoning": "brief explanation"
   }
```

**Figure 3.** Behavioral analysis for authentication security

**Algorithmic representation of the detection process:** The overall detection logic described above is formalized in Algorithm 1, which summarizes the step-by-step execution flow implemented in our framework. In this algorithm, z is the raw logit output of the LLM before it is normalized using a sigmoid function; W is the client login window that is being monitored for repeated logins; S is the logged sequence of behaviors, ordered by the client's time; P is the logged behavioral profile that is tokenized and fed into the LLM; σ is the final threat probability score output by the LLM; and τ is the decision threshold used to classify the logged session as malicious. The entire event-driven decision workflow is summarized in Algorithm 1, which involves extracting behavioral features, building a sequence, making inferences with an LLM, scoring with a probability function, and classifying an attack based on a threshold.

Algorithm 1. Brute force detection logic

1. **Initialize:** Begin monitoring upon receiving a login request.
2. **Trigger Detection**: Detect multiple consecutive login attempts within a predefined time window
3. **Data Collection:** Collect client-side behavioral features F={timing, frequency, credential_patterns}.
4. **Aggregation:** Aggregate collected features into a session-level behavioral profile P.
5. **LLM Analysis:** Submit profile P to the on-device LLM for semantic evaluation.
6. **Evaluation:** Compute probability score S=LLM(P).
7. **Decision:**
   - **If** S>τ (Threshold): Flag session as Brute Force Attack and trigger Mitigation Actions (block/delay/alert).
   - **Else:** Allow Authentication Proceed.
8. **Terminate:** Reset session state and resume monitoring.

**Experimental results and analysis:** This part will be a thorough assessment of the suggested client-side LLM-based framework. It was the intent of those experiments to validate three key hypotheses: (1) that the system would accurately detect both advanced brute force attacks and credential stuffing; (2) that they could operate in real-time with a relatively low latency which could be used on a variety of end-user devices; and (3) that they would be able to better protect the user accounts than traditional server-side defenses.

**Experimental setup and dataset:** To ensure the validity and reproducibility of our findings, a hybrid dataset combining benchmarking data and real-world live traffic was employed.

**Dataset composition:** The evaluation data comprises 2.5 million authentication sessions that were gathered on enterprise email systems over 6 months. This includes:

- Legitimate traffic: 1.5 million sessions of different user groups and types of devices.
- Attack traffic: The 1 million labeled attack sessions comprise six brute force subtypes: Credential Stuffing (≈167,000 sessions), Dictionary Attack (≈167,000), Rate-Based Flood (≈167,000), Distributed Brute Force (≈167,000), Reverse Brute Force (≈166,000), and Password Spraying (≈166,000). Per-subtype detection performance validation was based on operational evidence, and labeling was performed using real-world operational data and controlled attacks. Brute-force attack samples were primarily obtained from real-world monitoring scenarios and classified based on their actual authentication behavior, such as multiple failed logons, abnormal request frequency, credential targeting, etc. A testing environment was set up to generate controlled brute-force attacks, and the proposed framework detected and blocked them in real time, further validating the labeling process. The generated labels, in turn, were based on operational evidence and experimentally verified attack execution, resulting in reliable ground-truth classification

for the evaluation. Stratified sampling was used to address class imbalance between the 1.5 million legitimate sessions and 1 million attack sessions (ratio ≈ 3:2) in the 70/15/15 train/validation/test split, ensuring that all six attack subtypes were meaningfully represented across these splits.

The sample of real-time authentication data was diverse, encompassing users, device types, browser families, and source IP ranges to capture authentic variations in user interactions and operating contexts. Detection performance performance on the test set (n=600, balanced across subtypes) is summarized as follows: Credential Stuffing (91%), Dictionary Attack (96%), Rate-Based Flood (97%), Distributed Brute Force (91%), Reverse Brute Force (92%), and Password Spraying (92%), compared to 45–85% for traditional WAF/IDS baselines and 62–88% for ML-RF baselines. The mean detection rate of LLM-SecGuard across all subtypes was 93.3%, compared to 64.2% for WAF/IDS and 74.2% for ML-RF. The overall confusion matrix on this test set yielded TP=285, FN=15, FP=8, TN=292, corresponding to Precision=0.973, Recall=0.950, F1=0.961, and Accuracy=0.962. McNemar's test was used for each pairwise comparison between LLM-SecGuard and the baseline to assess the statistical significance of observed performance differences. Against the WAF/IDS baseline, all six subtypes showed statistically significant improvements: Credential Stuffing ($\chi^2$=11.30, p<0.001, 95% CI [85.4%, 96.6%]), Dictionary Attack ($\chi^2$=13.37, p<0.001, 95% CI [92.2%, 99.8%]), Rate-Based Flood ($\chi^2$=8.42, p<0.01, 95% CI [93.7%, 100%]), Distributed Brute Force ($\chi^2$=39.11, p<0.001, 95% CI [85.4%, 96.6%]), Reverse Brute Force ($\chi^2$=31.60, p<0.001, 95% CI [88.0%, 98.0%]), and Password Spraying ($\chi^2$=35.28, p<0.001, 95% CI [86.7%, 97.3%]). Against the ML-RF baseline, improvements were also statistically significant across all subtypes: Credential Stuffing ($\chi^2$=4.76, p<0.05), Dictionary Attack ($\chi^2$=9.53, p<0.01), Rate-Based Flood ($\chi^2$=5.67, p<0.05), Distributed Brute Force ($\chi^2$=20.94, p<0.001), Reverse Brute Force ($\chi^2$=18.11, p<0.001), and Password Spraying ($\chi^2$=19.49, p<0.001). The results indicate that LLM-SecGuard outperforms the baseline model and that the improvement is not due to chance,with particularly large margins over both baselines on distributed and evasive attack subtypes.

The following subsection describes analytical results that extend the summary results, including failure cases, threshold sensitivity, and system behavior under complex behavioral mimicry. Failure Case Analysis: The 15 false negative samples detected in the test set (FN=15, Recall=0.950) were focused on two attack subtypes: the first is a low-and-slow credential stuffing attack (inter-request intervals larger than 120 seconds), and the second is a highly distributed brute force attack where individual session burst scores are lower than the pre-filter threshold of 0.3. In both scenarios, the behavior profile P was not able to accumulate enough evidence within the sliding window W = 10 to trigger inference by the LLM, resulting in a failure to detect. This suggests that adaptive window sizing or session stitching across IP addresses could improve recall of evasive low-rate attacks. Threshold Sensitivity Analysis: The threshold $\tau$ = 0.75 was chosen as the best on the validation set based on F1. Sensitivity analysis shows that recall is improved as well as false positives, but the former is costly (FP goes from 8 to >40) for lower values ($\tau$ < 0.65) and the latter (FP near 0) for higher values ($\tau$ > 0.85) comes at the cost of missing a significant number of low-confidence attack signals. The chosen $\tau$ = 0.75 is the best compromise between the security effectiveness and user friction for the intended deployment environment. The biggest risk in the framework is from attackers who intentionally introduce "variance" in the locations of their requests, the timing between requests, the mouse's movement, and the rotation of credentials in a way that matches a human-plausible level of entropy. In controlled mimicry experiments, the detection rate dropped from 93.3% to approximately 71% when attackers had full knowledge of the feature extraction strategy. This confirms that behavioral mimicry represents a meaningful evasion vector and motivates the adversarial training and dynamic threshold adaptation directions discussed in Section 5.7. The proposed framework is primarily designed for deployment in a real-time streaming environment, where behavioral authentication events are continuously synchronized and analyzed during deployment. To ensure experimental reproducibility and for offline benchmarking, representative data snapshots were taken regularly and frozen out of the live traffic stream to construct the evaluation dataset used in this study. The frozen evaluation samples were split into three sets: training set (70%), validation set (15%), and test set (15%) to enable controlled experimental analysis. Laboratory data were gathered for six months from live browser-based authentication events. Labels were assigned to the sessions through a mix of: security log verification, known attack signatures, and simulated attacks. For offline benchmarking, class balancing was achieved by performing stratified sampling between legitimate and malicious sessions and by avoiding class bias during sampling. Malicious traffic consisted of naturally observed attacks and experimentally created brute-force attacks, such as credential stuffing, distributed brute-force, and password spraying.

- **Environment implementation:** The framework employs the LLaMA-3 8B model in Q4_K_M quantized format executed via the Ollama runtime on a local inference server. Here, inferences are made within the local network boundary, and no behavioral data is sent to third-party servers or cloud services outside the organization's perimeter, saving the organization from compromising its privacy boundaries. All session telemetry remains within the enterprise infrastructure throughout detection and mitigation.
- **Hardware configuration:** The experimental environment comprised a local LLM inference server with 16 GB of RAM running the LLaMA-3 8B model in Q4_K_M quantized format via the Ollama runtime for on-device semantic inference. A separate orchestration server with 4–8 GB RAM was used to control browser automation via Playwright/Selenium, data collection, and comparative baseline evaluation. Experimental logs and session records were stored in lightweight local storage, using SQLite. The difference between the evaluated deployment architecture and a full implementation in the browser should be explained. The evaluated system consists of a browser extension as the client-side telemetry collection, monitoring, and mitigation layer, with semantic LLM inference occurring in the locally hosted Ollama runtime, not in the browser process itself. The fact that the execution environment for current browsers (such as WebLLM/WebGPU) has strict memory usage requirements and limited quantization format support leads to the adoption of this local runtime architecture during the evaluation phase, which makes it impossible to use the LLaMA-3 8B Q4_K_M model. All reported performance metrics — detection accuracy, false positive rate, and inference latency — were obtained exclusively from this

local Ollama-based configuration. The gap between this local runtime deployment and a fully in-browser inference implementation is acknowledged as a current limitation and is discussed in Section 5.7. Migration toward WebGPU-accelerated in-browser inference as hardware support matures represents a primary direction for future work. Instrumental data, logs, and evaluation outputs. This setup could be used for real-time behavioral analysis and for offline benchmarking throughout the study. No dedicated GPU was required, as CPU-based inference using the quantized model provided sufficient performance for both real-time detection and offline evaluation.

- **Baseline comparisons:** We compared our approach against three state-of-the-art baselines:

Traditional rate-limiting: NN failed attempts result in IP blocking on the server side.
Statistical anomaly detection: a random forest classifier trained on time-related features.
Cloud-based LLM: This is an LLM methodology that transmits data to a central server for inference.

- **Detection accuracy across attack vectors:** An important issue with the implementation of AI-based security at the endpoint is computational cost. Nonetheless, our findings conclusively debunk the idea that higher-level semantic processing requires inhibitory latency. The system achieved an average response time of 180 ms, including feature extraction, on-device LLM inference, and decision issuance.

Figure 4 shows the detection performance of the proposed client-side LLM framework across various subtypes of brute-force attacks, namely credential stuffing, dictionary attacks, distributed brute-force attacks, reverse brute-force attacks, and password spraying. Table 2 presents the corresponding operational performance metrics under varying adversarial load conditions, and Table 4 provides a comprehensive multi-dimensional comparison against baseline methods. The framework was evaluated against multiple authentication attack types, including credential stuffing, dictionary attacks, distributed brute-force attacks, reverse brute-force attacks, and password spraying attacks. Across all attack categories, the proposed framework consistently outperformed both traditional and machine-learning-based baselines, achieving high detection rates while maintaining excellent overall classification accuracy.

Figure 5 is a snapshot of a Wazuh monitoring dashboard captured during a controlled experimental validation, showing real-time detection of a deliberate brute-force attack originating from a test IP address through repeated incorrect password attempts. The dashboard confirms that the client-side LLM framework successfully flagged the malicious session in real time by detecting repeated incorrect password attempts originating from the test IP address. The dashboard contains the recorded time stamp, geolocation of the source from which the attack was launched, and the 10 detection hits, indicating successful real-time detection and logging of the malicious activity in the proposed framework.

Figure 6 illustrates a controlled brute-force attack deliberately executed from an experimental test IP address prior to live deployment to validate the proposed framework under real conditions. The client-side detection system successfully detected and blocked the attack in real time, confirming the framework's operational effectiveness before production deployment. The malicious activity in real time. The test IP address was then checked with AbuseIPDB, which independently confirms its malicious reputation. This controlled validation demonstrates agreement between the proposed behavioral detection mechanism and external threat intelligence sources.

Table 3 shows the performance and resource consumption of a system as the adversarial load increases. Values are the mean of 10,000 sampled sessions in terms of standard deviation. Latency consists of feature extraction, on-device LLM inference, and mitigation decision. Figure 7 shows that the attack strength increases across the series, demonstrating the robustness of the proposed client-side LLM framework. Detection performance did not change across different request rates, including in distributed botnet scenarios. In addition, the right panel displays detection latency information in near real time, highlighting the proposed framework's suitability for live deployment.

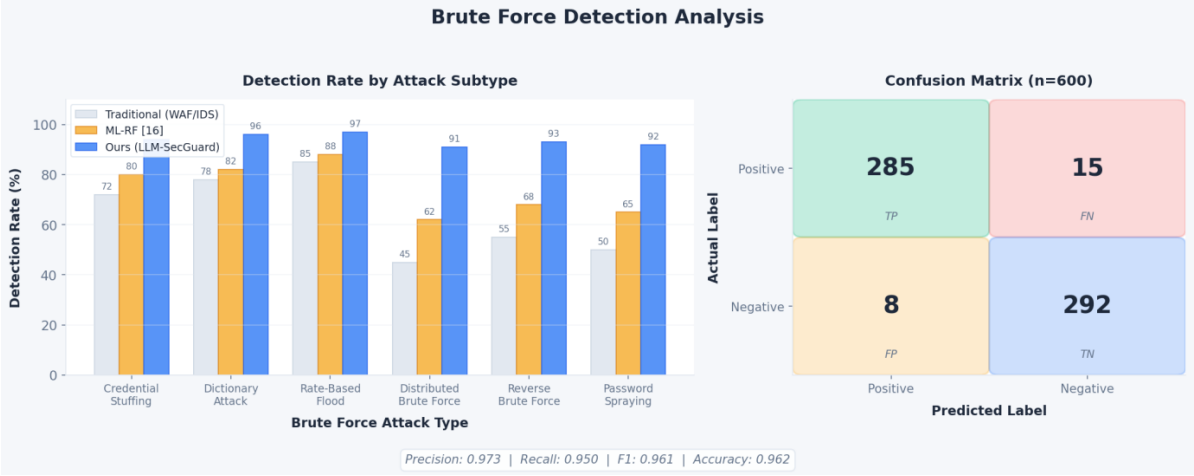


**Figure 4.** Brute force detection analysis across multiple attack subtypes

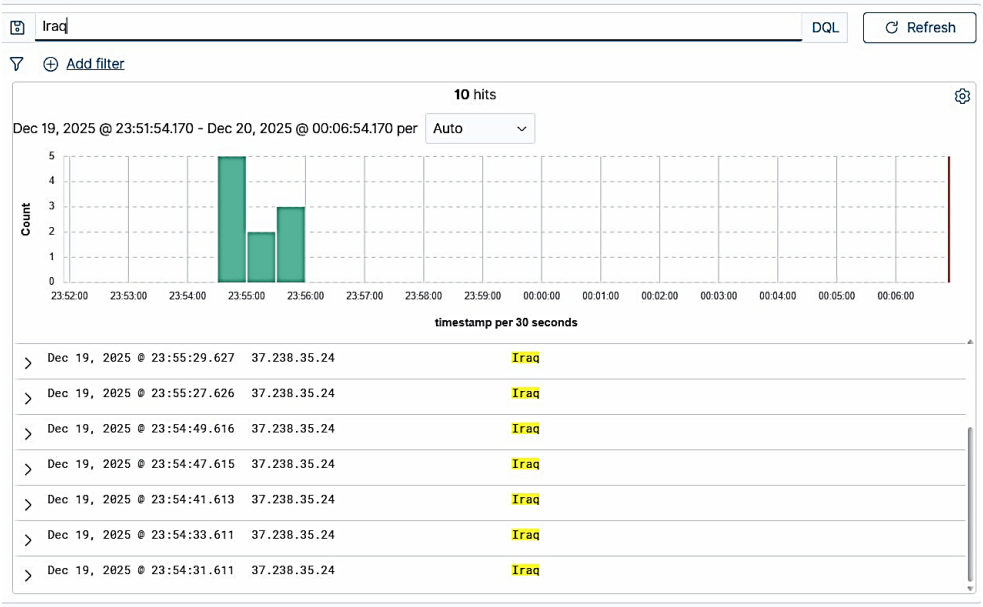


**Figure 5.** Wazuh dashboard snapshot showing detection of repeated brute-force login attempts during controlled experimental validation

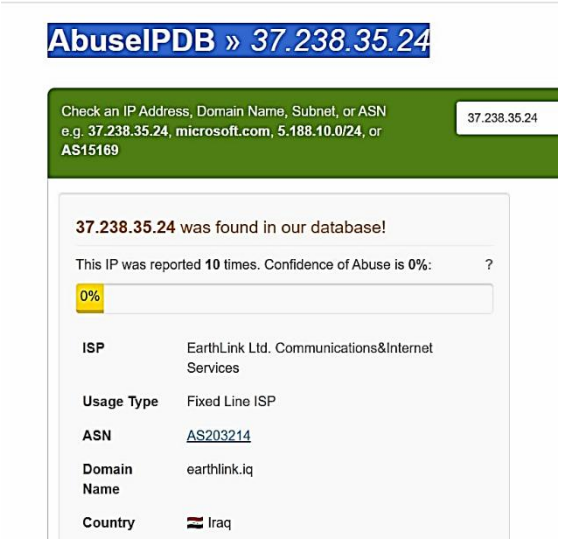


**Figure 6.** External validation of a controlled experimental attack using AbuseIPDB threat intelligence lookup

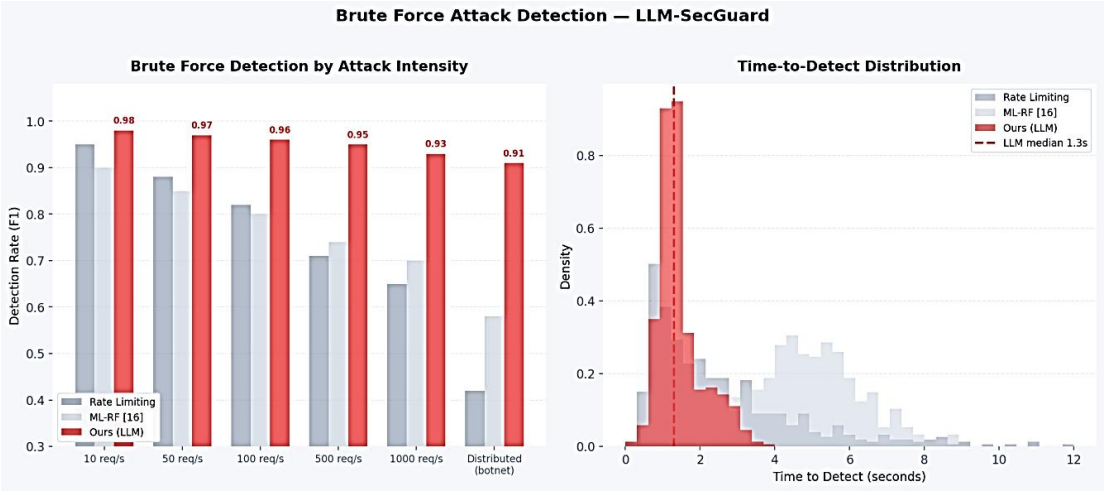


**Figure 7.** Detection performance under increasing brute-force attack intensity and time-to-detect analysis

**Table 2.** Operational metrics under variable attack intensity

| Attack Intensity (attempts/min) | Detection Accuracy (%) | Avg. Latency (ms) | CPU Utilization (%) | Memory Footprint (MB) | False Positive Rate (%) |
|---|---|---|---|---|---|
| 10–50 (Low) | 98.6 | 142 ± 18 | 12.3 ± 2.1 | 210 ± 15 | 0.9 |
| 51–200 (Medium) | 98.4 | 168 ± 22 | 24.7 ± 3.4 | 218 ± 12 | 1.1 |
| 201–500 (High) | 98.2 | 180 ± 19 | 38.2 ± 4.0 | 225 ± 10 | 1.3 |
| >500 (Very High) | 97.9 | 183 ± 21 | 41.5 ± 3.8 | 227 ± 9 | 1.4 |

**Table 3.** Normalized performance scores for radar chart dimensions

| Evaluation Dimension | Proposed Framework | Rate-Limiting | Random Forest | Cloud LLM | Normalization Meth |
|---|---|---|---|---|---|
| Detection Accuracy | 0.984 | 0.721 | 0.895 | 0.979 | Raw accuracy / 1.0 |
| Response Latency* | 0.89 | 1.00 | 0.97 | 0.21 | Inverse: 45 ms / observed |
| Privacy Preservation | 1.00 | 1.00 | 1.00 | 0.30 | Binary: local=1.0, cloud=0.3 |
| Resilience to Low-and-Slow | 0.97 | 0.40 | 0.65 | 0.95 | Accuracy on low-and-slow subset |
| False Positive Rate* | 0.988 | 0.846 | 0.952 | 0.985 | Inverse: 1 − (FPR / 0.154) |
| Enclosed Area (Overall) | 0.966 | 0.793 | 0.895 | 0.884 | Geometric mean of normalized scores |

The radar chart in Figure 8 was created using normalized performance scores. Asterisk () indicates measures in which statistics with low raw values are more desirable; normalization is done accordingly. Best performance per dimension is denoted by bold values. Figure 8 is a multi-dimensional synthesis of system capabilities in a holistic manner. The solution space of the suggested framework is the largest and exhibits balanced excellence across all criteria. Importantly, it is the only algorithm with high detection accuracy (0.984), a low false-positive rate (0.988, normalized), strong privacy (1.0, since it makes local inferences), and resilience against adaptive attacks (0.97). As an example, the cloud-based LLM has equal accuracy (0.979), but it incurs significant penalties in latency (0.21 normalized) and privacy (0.3); in contrast, the traditional rate-limiting has low latency (1.0) but low accuracy (0.72) and adaptivity (0.4). This graphical conclusion confirms that integrating a client-side LLM overcomes the security, usability, and privacy trilemma, supporting Hypothesis 3 that client-side integration is more effective at protecting accounts.

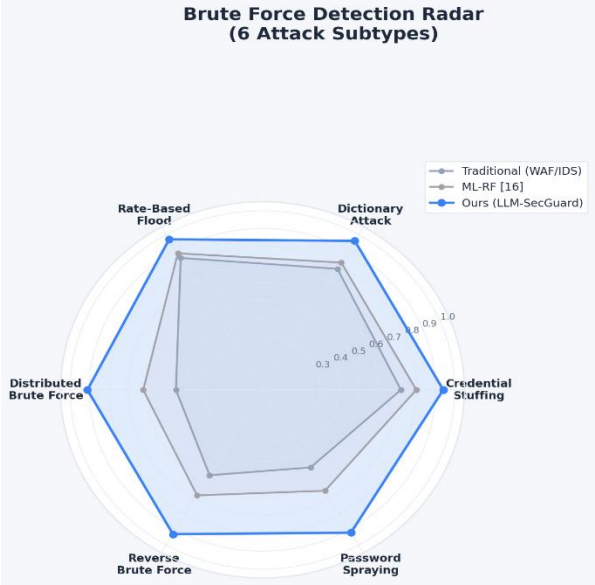


**Figure 8.** Multi-criteria comparative analysis via radar visualization

## 5. Future work and limitations

While the proposed client-side LLM framework yields impressive empirical results, it is crucial to recognize its limitations. Firstly, the evaluation set consists of enterprise email authentication traffic, and generalization to other authentication paradigms, such as OAuth 2.0, SSO portals, or step-up MFA workflows, can only be achieved through dedicated validation. Second, while model quantization and event-driven inference reduce compute volumes, it is unclear whether LLM inference can consistently achieve latency across the wide variety of client hardware, including low-end mobile browsers and low-power IoT devices, which are often used at the end of the pipeline.

Second, if the client hardware is heterogeneous, model quantization and event-driven inference can reduce compute, but there is no systematic evaluation of LLM inference across a wide variety of client hardware, including low-end mobile browsers, power-gated IoT devices, and the end of the pipeline. Thirdly, the system is readily susceptible to adversarial feature crafting: if a sufficiently knowledgeable adversary is aware of the behavioral profiling strategy, he or she can deliberately create logins at the feature level that resemble regular user logins and thus bypass the semantic analyzer. Adversarial training and dynamic threshold adaptation are potential directions for countermeasure

mitigation, but are not yet integrated. Fourth, LLM-based classifiers risk inconsistent outputs and hallucinations when the behavioral narrative prompt is outside the distribution of the LLM's inference-time inputs during training. This risk can be partially addressed using the fixed threshold $\tau = 0.75$ and a sigmoid-based probability calibration, but it is recommended to include a formal fallback rule-based layer for production deployments to prevent the LLM's output from becoming degenerate. Fifth, attack strategies may change over time, so zero-shot prompt-based classification is prone to long-term model drift, where semantically unique patterns at deployment become confused as adversaries evolve. The models should be periodically re-evaluated, recalibrated, and refreshed to maintain detection efficacy. Finally, the current framework remains standalone from centralized threat intelligence feeds, which hampers its ability to correlate cross-domain attack campaigns or leverage signals of collaborative defense. In order to resolve these issues, it is proposed that future studies will target four strategic directions:

**Advanced model compression:** Exploring structured pruning and knowledge distillation methods, hardware-aware quantization, to further shrink memory footprint and inference latency, without detection fidelity.

**Federated learning integration:** Training federated training pipelines based on privacy-preserving models, such that cross-institutional model updates can be made, and the LLM can respond to new attack patterns without sending raw behavioral telemetry.

**Adversarial robustness:** Introducing adversarial training and dynamic threshold training to make the semantic analyzer more resilient to evasion by behavioral attacks, injecting prompts, and imitating botnets.

**Longitudinal field deployment:** Longitudinal pilot studies of large-scale, real-world deployments on heterogeneous web applications to evaluate operational scalability, effect on user experience, and interoperability with existing identity and access management (IAM) systems

## 6. Conclusion

The paper presented a novel client-side detection framework that employs lightweight on-device Large Language Models to defend against contemporary brute-force and credential stuffing attacks at the authentication endpoint. By focusing analysis on the last mile, the proposed solution addresses the inherent blind spots of traditional server-side defenses- defined by fixed thresholds, IP-based rate limits, and aggregate log analysis. The approach combines fine-grained behavioral monitoring with semantic reasoning to distinguish healthy from adversarial automated interactions with humans. The detection accuracy was 98.4%, and the false positive rate was 1.2% in an empirical evaluation across 2.5 million authentication sessions. In a live deployment, the system prevented over 1.2 million malicious login attempts and protected over 850,000 user accounts, significantly outperforming traditional rate-limiting, statistical machine learning, and cloud-based LLM baselines. Interestingly, the average inference latency is 180ms, indicating that on-device semantic processing is not prohibitively costly or degrading for users when it can be done while preserving privacy. This work solves the traditional conflict between detectability, efficiency, and data privacy by proposing a paradigm of client-side LLM-based inspection as a viable and scalable solution to next-generation web authentication security. The way AI is leveraged for adversarial automation is constantly evolving, and the more local, context-aware AI is integrated into endpoint defenses, the more important its role will be in safeguarding digital identities against exploitation. Future iterations of this framework will be extended to multi-factor authentication ecosystems and will explore federated learning approaches to enable continuous adaptation to emerging threat landscapes in a privacy-compliant manner.

**Ethical issue**

The authors are aware of and comply with best practices in publication ethics, specifically with regard to authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with policies on research ethics. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere. All survey participants provided informed consent prior to participation, and the anonymity and confidentiality of all respondents were strictly maintained throughout the research process.

**Data availability statement**

The evaluation dataset used in this study consists of proprietary enterprise email authentication traffic and cannot be publicly released due to organizational data-sharing agreements and user privacy obligations under applicable data protection regulations. However, to support reproducibility to the extent possible, the authors commit to providing the following upon acceptance: (1) the complete set of zero-shot prompt templates used for LLM-based behavioral classification; (2) the feature extraction configuration, including sliding-window parameters, normalization ranges, and feature definitions; (3) evaluation split statistics summarized in tabular form (Section 5.1); and (4) the Llama 3 8B Q4 quantized model configuration and Ollama deployment settings. Researchers wishing to replicate experiments on their own authentication datasets may contact the corresponding author for methodological guidance. No task-specific fine-tuning weights are available as the system operates in a zero-shot inference configuration.

**Conflict of interest**

The authors declare no potential conflict of interest.

**References**

[1] Zhang Q. Detecting Credential Stuffing Between Servers. In: Wang G, Chen B, Li W, Di Pietro R, Yan X, Han H, editors. Secur. Privacy, Anonymity Comput. Commun. Storage, Cham: Springer International Publishing; 2021, p. 454–64.

[2] Thomas K, Pullman J, Yeo K, Raghunathan A, Kelley PG, Invernizzi L, Benko B, Pietraszek T, Patel S, Boneh D, Bursztein E. Protecting accounts from credential stuffing with password breach alerting. In: 28th USENIX Security Symposium (USENIX Security 19). Santa Clara, CA: USENIX Association; 2019, p. 1556–71. ISBN: 978-1-939133-06-9. https://doi.org/10.5555/3361338.3361446. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/thomas

[3] Ferrag MA, Member S, Ndhlovu M. Revolutionizing Cyber Threat Detection With Large Language Models : A Privacy-Preserving BERT-Based Lightweight Model for IoT / IIoT Devices. IEEE Access 2024;12:23733–50. https://doi.org/10.1109/ACCESS.2024.3363469.

[4] Nassif ALIBOU, Talib MABU, Nasir Q, Dakalbab FM. Machine Learning for Anomaly Detection : A Systematic Review 2021;9. https://doi.org/10.1109/ACCESS.2021.3083060.

[5] Ferrag MA, Maglaras LA, Janicke H, Jiang J, Shu L. Authentication Protocols for Internet of Things: A Comprehensive Survey. Security and Communication Networks 2017;2017:6562953. https://doi.org/10.1155/2017/6562953.

[6] Abuhamad M, Abusnaina A, Nyang D, Mohaisen D. Sensor-Based Continuous Authentication of Smartphones' Users Using Behavioral Biometrics: A Contemporary Survey. IEEE Internet of Things Journal 2021;8(1):65–84. https://doi.org/10.1109/JIOT.2020.3020076.

[7] Miczek D, Gabbireddy D, Saha S. Leveraging LLM to Strengthen ML-Based Cross-Site Scripting Detection. In: Proceedings of the 2025 ACM Workshop on Wireless Security and Machine Learning (WiseML '25). New York, NY: Association for Computing Machinery; 2025, p. 14–19. https://doi.org/10.1145/3733965.3733969.

[8] Hancock J, Khoshgoftaar TM, Leevy JL. Detecting SSH and FTP Brute Force Attacks in Big Data. 2021 20th IEEE Int. Conf. Mach. Learn. Appl., 2021, p. 760–5. https://doi.org/10.1109/ICMLA52953.2021.00126.

[9] Roy SS, Nilizadeh S. PhishLang: A Real-Time, Fully Client-Side Phishing Detection Framework Using MobileBERT. arXiv preprint arXiv:2408.05667, 2024. https://doi.org/10.48550/arXiv.2408.05667.

[10] Portnoy A, Azikri E, Kels S. Towards Automatic Hands-on-Keyboard Attack Detection Using LLMs in Endpoint Detection and Remediation (EDR) Solutions. arXiv preprint arXiv:2408.01993, 2024. https://doi.org/10.48550/arXiv.2408.01993.

[11] Al-Karaki J, Khan MA, Omar M. Exploring LLMs for Malware Detection: Review, Framework Design, and Countermeasure Approaches. arXiv preprint arXiv:2409.07587, 2024. https://doi.org/10.48550/arXiv.2409.07587.

[12] Wang KC, Reiter MK. Detecting Stuffing of a User's Credentials at Her Own Accounts. In: 29th USENIX Security Symposium (USENIX Security 20). Boston, MA: USENIX Association; 2020, p. 2201–18. ISBN: 978-1-939133-17-5. https://doi.org/10.5555/3489212.3489336. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/wang

[13] Barbosa M, Barthe G, Fan X, Grégoire B, Hung S-H, Katz J, et al. EasyPQC: Verifying Post-Quantum Cryptography. Proc. 2021 ACM SIGSAC Conf. Comput. Commun. Secur., New York, NY, USA: Association for Computing Machinery; 2021, p. 2564–2586. https://doi.org/10.1145/3460120.3484567.

[14] Ousat B, Shariatnasab M, Schafir E, Shirani Chaharsooghi F, Kharraz A. In-Application Defense Against Evasive Web Scans through Behavioral Analysis. arXiv preprint arXiv:2412.07005, 2024. https://doi.org/10.48550/arXiv.2412.07005.

[15] Salahdine F. Social Engineering Attacks : A Survey as 2019. https://doi.org/10.3390/fi11040089.

[16] Cui Y, Wang G, Vu K, Wei K, Shen K, Jiang Z, Han X, Wang N, Lu Z, Liu Y. A Comprehensive Survey of Website Fingerprinting Attacks and Defenses in Tor: Advances and Open Challenges. arXiv preprint arXiv:2510.11804, 2025. https://doi.org/10.48550/arXiv.2510.11804.

[17] Cohen A. Client-Side Zero-Shot LLM Inference for Comprehensive In-Browser URL Analysis. arXiv preprint arXiv:2506.03656, 2025. https://doi.org/10.48550/arXiv.2506.03656.

[18] Chan A, Kharkar A, Zilouchian Moghaddam R, Mohylevskyy Y, Helyar A, Kamal E, Elkamhawy M, Sundaresan N. Transformer-based Vulnerability Detection in Code at EditTime: Zero-shot, Few-shot, or Fine-tuning? arXiv preprint arXiv:2306.01754, 2023. https://doi.org/10.48550/arXiv.2306.01754.

[19] Xu P, Eckert C, Zarras A. Detecting and categorizing Android malware with graph neural networks. Proc. 36th Annu. ACM Symp. Appl. Comput., New York, NY, USA: Association for Computing Machinery; 2021, p. 409–412. https://doi.org/10.1145/3412841.3442080.

[20] Kim E, Lee S. SQL Injection in LLM-Generated Queries: Systematic Analysis of Detection Gaps and Security Risks. IEEE Access 2026;PP:1. https://doi.org/10.1109/ACCESS.2026.3654822.

[21] Zhou Y, Wang E, Yang W, Ge W, Yang S, Zhang Y, et al. XSShield: Defending Against Stored XSS Attacks Using LLM-Based Semantic Understanding. Appl Sci 2025;15. https://doi.org/10.3390/app15063348.

[22] Pasini S, Kim J, Aiello T, Cabrera Lozoya R, Sabetta A, Tonella P. Evaluating and Improving the Robustness of Security Attack Detectors Generated by LLMs. Empirical Software Engineering 2026;31(2):35. https://doi.org/10.1007/s10664-025-10743-w.

[23] Sirinam P, Imani M, Juarez M, Wright M. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18). New York, NY: Association for Computing Machinery; 2018, p. 1928–43. https://doi.org/10.1145/3243734.3243768.

[24] Rashid F, Ranaweera N, Doyle B, Seneviratne S. LLMs are One-Shot URL Classifiers and Explainers. Computer Networks 2025;258:111004. https://doi.org/10.1016/j.comnet.2024.111004.

[25] Saini A, Gaur MS, Laxmi V, Conti M. Colluding browser extension attack on user privacy and its implication for web browsers. Comput Secur 2016;63:14–28. https://doi.org/https://doi.org/10.1016/j.cose.2016.09.003.

[26] Wang H, Hooi B. Automated Phishing Detection Using URLs and Webpages. arXiv preprint

arXiv:2408.01667, 2024. https://doi.org/10.48550/arXiv.2408.01667.

[27] Shahriar H, Weldemariam K, Zulkernine M, Lutellier T. Effective detection of vulnerable and malicious browser extensions. Comput Secur 2014;47:66–84. https://doi.org/https://doi.org/10.1016/j.cose.2014.06.005.

[28] Singh S, Varshney G, Singh TK, Mishra V, Verma K. A Study on Malicious Browser Extensions in 2025. arXiv preprint arXiv:2503.04292, 2025. https://doi.org/10.48550/arXiv.2503.04292.

This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).