International Journal of Basic and Applied Sciences, 14 (7) (2025) 296-304

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/h8f10w38
Research paper

Countering AI-Driven Adversaries: An Adaptive Deception Framework for Resilient Web Security

Sa'eed Serwan Abdulsattar [1] *, Hani Al-Balasmeh [2], Mohammed Majed Mohammed Al-Khalidy [1], Fayzeh Abdulkareem Jaber [3], Rahmeh Abdulkaeem Jaber [4]

[1] Dept of Electrical and Electronics Engineering, University of Bahrain
[2] Dept of Informatics Engineering, College of Engineering, University of Technology, Bahrain
[3] Dept of Computer Studies, University of Technology, Bahrain
[4] Dept of Business Administration, University of Technology, Bahrain
**Corresponding author E-mail: h.albalasmeh@utb.edu.bh*

Received: October 12, 2025, Accepted: November 8, 2025, Published: November 12, 2025

Abstract

This paper presents a novel deception-based cybersecurity framework that redefines web defense through adaptive, embedded traps designed to detect and contain AI-driven automated adversaries. With the rapid advancement of machine learning (ML) and large language models (LLMs), traditional web security measures—such as CAPTCHA, honeypots, and Web Application Firewalls (WAFs)—have become increasingly ineffective. Modern bots can now simulate human browsing, execute JavaScript, and evade detection through the use of adaptive algorithms. The proposed framework introduces invisible, dynamically generated traps within the Document Object Model (DOM) and JavaScript layers of web applications to exploit the behavioral disparities between genuine users and automated systems.
These traps include hidden forms, off-screen anchor links, and randomized decoy endpoints that are imperceptible to human users but tectable by automated bots. Interactions with these traps trigger behavioral analysis routines that calculate a Non-Human Interaction Likelihood (NHIL) score, a novel session-level metric that employs sigmoid activation functions to measure behavioral abnormality across multiple parameters. Based on this score, the system classifies, logs, and isolates non-human activity in real time.
An experimental evaluation in a test web environment demonstrated perfect classification performance, with an F1-score of 1.0, achieving complete detection accuracy without false positives or degradation in user experience. Page load latency increased by less than five milliseconds, confirming the framework's lightweight and seamless integration.
By merging adversarial design, behavioral analytics, and adaptive deception, the proposed system establishes a resilient, intelligence-driven approach to web security. It transforms defensive architecture from reactive to proactive—detecting, engaging, and neutralizing automated adversaries at the interaction layer—offering a scalable model for the next generation of deception-centric cybersecurity.

Keywords: *Cybersecurity; Deception Framework; Behavioral Analytics; Bot Detection; Web Security; Machine Learning.*

1. Introduction

Recent advancements in machine learning (ML) and large language models (LLMs) [1]–[3] have precipitated a profound shift in the cybersecurity landscape, undermining long-established defense mechanisms such as the Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) [4], [5]. Initially, CAPTCHAs were designed to exploit the cognitive limitations of early machine learning algorithms, leveraging tasks like character recognition and semantic interpretation that were once difficult for automated systems to solve [6]. However, as ML capabilities evolved, the balance of power gradually shifted in favor of machines [7]. Modern models now demonstrate superior pattern recognition and feature extraction capabilities, enabling them to outperform humans in solving complex CAPTCHA challenges [8]. These include text-based, image-based, and behavioral variants, all of which have been systematically bypassed by adversarial AI models [9]–[11]. State-of-the-art deep learning architectures, particularly convolutional neural networks (CNNs) and object detection frameworks such as YOLO, achieve near-perfect accuracy in solving CAPTCHA, effectively nullifying their role as reliable human verification tools [12].
This evolution has created a critical paradox between usability and security. As CAPTCHA becomes increasingly complex to deter machine learning attacks, it simultaneously degrades the experience for legitimate users, with human success rates falling below 70% for intricate challenges. Simultaneously, Web Application Firewalls (WAFs), another cornerstone of web protection, are struggling to adapt to new evasion techniques [13]. Reinforcement learning-based black-box testing models have emerged that can bypass WAF defenses with remarkable efficiency [14]. In contrast, evaluations of rule-based systems, such as ModSecurity, reveal significant limitations in detecting injection-based attacks, like SQLi [15].
Equally concerning is the decline in the effectiveness of honeypots—once regarded as vital tools for collecting threat intelligence. Attackers have begun exploiting the inherent design flaws of these systems. For instance, Patel et al. [16] documented cases of cryptojacking

operations that leveraged the Cowrie honeypot to mine cryptocurrency, thereby turning a defensive asset into an attack vector. Although research continues to enhance honeypot resilience [17], such systems remain susceptible to detection and resource exhaustion.

At the network perimeter, even advanced intrusion detection and mitigation systems are increasingly ineffective against traffic generated by LLM-powered bots. These bots can emulate human-like browsing, context retention, and adaptive evasion strategies [18]. Emerging techniques, such as the Substitution-based In-Context Optimization (SICO) method, enable LLMs to generate text that is indistinguishable from human output, thereby bypassing AI text detectors entirely [19]. This convergence of high-capacity automation and adaptive deception creates an environment where traditional rule-based and reactive defenses are rapidly losing relevance.

A recurring pattern emerges among current security mechanisms: they are predominantly external, static, and reactive, focusing on detection after the intrusion rather than dynamic engagement during it. The inevitability of AI-based circumvention is already evident [20]; studies show that autonomous agents can navigate websites, simulate user sessions, and harvest sensitive data without triggering standard detection systems [21], [22]. These developments underscore an urgent need to rethink the architecture of web security from a defensive to a deceptive paradigm.

Rather than relying solely on external barriers, the next generation of web defense must incorporate deception-based mechanisms directly into internal architectures. Embedding invisible traps within web applications introduces a proactive layer that detects and neutralizes automated adversaries in real time. These traps exploit inherent behavioral discrepancies between genuine user interactions—such as event-driven JavaScript execution, delayed mouse movement, and DOM-based interactions—and automated agents that indiscriminately parse HTML structures or send API calls without visual rendering.

By serving misleading payloads such as decoy login pages, fabricated vulnerabilities, or non-existent administrative endpoints, the system lures automated adversaries into controlled environments. Interaction with these traps triggers event-based monitoring systems that log identifiers, redirect malicious traffic, or sandbox sessions for forensic analysis. This method transforms the defensive model from passive detection to active engagement, encouraging adversaries to self-identify through their behavioral patterns.

The novelty of this research lies in its integration of behavioral analytics, adversarial design, and deception within a unified adaptive framework. Unlike static honeypots or conventional CAPTCHA mechanisms, the proposed system continuously evolves its deceptive elements—randomizing identifiers, shuffling traps, and adapting thresholds based on observed interaction data. By embedding these components at the web layer, it achieves resilience without compromising usability or performance.

This paradigm shift marks a transition from perimeter-based protection to embedded resilience, where deception, analytics, and adaptive learning converge to create a living defense mechanism capable of countering AI-driven adversaries in real-time.

1.1. Background and related work

Early developments in cyber deception primarily focused on passive decoy mechanisms, such as honeypots and honeytokens [23]. These systems simulate legitimate services or data to attract attackers, diverting them from actual assets while collecting valuable intelligence on their tactics, techniques, and procedures (TTPs). Javadpour et al. [24] conducted a comprehensive review of enhancement techniques designed to improve honeypot effectiveness by mitigating detectability and extending operational longevity, which remain key challenges in maintaining the credibility of deception. Similarly, Müller et al. [25] developed performance evaluation frameworks for honeypots, emphasizing interaction depth, data authenticity, and false-positive rates as core performance indicators. While these contributions strengthened deception-based intelligence gathering, their inherent reactivity limits their efficacy in rapidly evolving attack environments.

To introduce adaptability and proactive defense, researchers proposed the Moving Target Defense (MTD) paradigm, which continuously alters system configurations—such as IP addresses, ports, and code variants—to increase attacker uncertainty [26]. This approach transforms network predictability into a dynamic variable, complicating reconnaissance and exploitation. In the context of next-generation infrastructures, Soussi et al. [27] extended MTD concepts to Beyond 5G and 6G networks, highlighting their role in securing ultra-reliable low-latency communication (URLLC) and distributed edge systems. These studies emphasize that adaptive configuration changes alone are insufficient; true resilience requires integrating attacker engagement and intelligence extraction into the defensive process.

Pawlick et al. [28] advanced this discussion through a game-theoretic taxonomy of defensive deception, classifying techniques such as MTD, perturbation, and honey-x (including honeypots and honeytokens) according to their strategic interactions with adversaries. Their taxonomy identifies three central principles: (1) adaptive crypsis, in which defenders continuously alter observable features to obscure system states; (2) adversarial engagement, which draws attackers into controlled deception environments; and (3) cost asymmetry, which increases the economic and computational burden on attackers relative to defenders. These theoretical foundations are particularly relevant for modern web security, where lightweight, context-aware deception strategies must operate seamlessly within distributed architectures.

Despite these advancements, existing honeypot and MTD systems essentially function as isolated modules, detached from active user workflows and standard web protocols. Their reliance on separate monitoring infrastructures and manual calibration hinders real-time adaptability. The proposed framework builds upon this foundation by embedding deception directly within web layers—specifically, HTML and JavaScript—allowing adaptive engagement through real-time behavioral analysis. Unlike conventional systems that react post-intrusion, this approach dynamically tailors traps and responses during live sessions, leveraging continuous monitoring of Document Object Model (DOM) interactions, JavaScript event execution, and anomalous request patterns.

Contemporary studies have begun exploring hybrid deception models that integrate dynamic response mechanisms. These systems employ real-time analysis of user behavior to adjust network responses, presenting decoy data or services based on contextual risk levels. For instance, web-based deception systems can generate fake endpoints or rotate authentication tokens in response to abnormal interaction patterns. This trend reflects a broader shift from static deception toward behavior-aware, context-driven engagement—an approach aligned with the adaptive principles underpinning the proposed model.

The integration of behavioral analytics into deception architectures introduces a new level of sophistication. By exploiting the subtle behavioral discrepancies between human and automated agents—such as timing irregularities, script execution depth, and event-trigger consistency—defenders can infer the likelihood of non-human activity. When combined with adversarial engagement and environmental mutation, this fusion enables an active deception strategy that can misleadingly isolate and profile automated adversaries in real-time.

In summary, the proposed framework advances existing research by merging the reactive intelligence-gathering strengths of honeypots, the adaptability of moving target defenses, and the precision of behavioral analytics into a cohesive deception ecosystem. This integrated design moves beyond external defenses and positions deception as a foundational layer of web architecture, transforming traditional protection models into adaptive, intelligent systems capable of confronting AI-driven threats with dynamic resilience.

2. Proposed Framework and System Design

The proposed framework introduces a deception-driven architecture for web defense that embeds invisible traps within the structure of web applications and endpoints. Unlike conventional systems that rely on external mechanisms such as CAPTCHA, rate-limiting, or static honeypots [6], [15], [17], this model integrates adaptive deception directly into the web layer. Its design leverages behavioral analysis and adversarial engagement to detect, isolate, and analyze automated bots without degrading the legitimate user experience.

At its core, the framework exploits the behavioral and functional discrepancies between human users and automated agents. Genuine users typically interact with visible page elements through dynamic browser actions such as clicking, scrolling, and executing JavaScript-based features. In contrast, computerized bots—particularly those using headless browsers or API-level parsing—operate by processing HTML structures, making network requests, or extracting content without rendering visual elements [29]. The system capitalizes on these differences by embedding deceptive components that remain invisible or inaccessible to human users but are detectable by automated systems.

a) Embedded Deceptive Elements

Deceptive components are inserted into the Document Object Model (DOM) at both static and dynamic levels. Static traps include hidden HTML forms and off-screen anchor tags that are visually concealed using attributes such as display: none, opacity, or absolute positioning beyond the visible screen area. An example of an invisible form embedded into the DOM is shown in Fig. 1, where the hidden <form> element is accessible only to automated parsers.

```html
<form id="hidden-login" action="/fake_admin_login" method="POST" style="display:none;">
  <input type="text" name="username" value="admin">
  <input type="password" name="password" value="password123">
  <input type="submit" value="Login">
</form>
```

Fig. 1: A Sample of HTML Setup for the Trap.

Each static element is linked to event listeners to detect any automated interaction (see Fig. 2). These listeners monitor form submissions, clicks, or focus events and trigger behavioral logging when activated.

```javascript
document.getElementById('hidden-login').addEventListener('submit',
    function(event) {
    event.preventDefault();
    fetch('/log_bot', {
        method: 'POST',
        body: JSON.stringify({
            event: 'Hidden Form Submission',
            timestamp: Date.now(),
            userAgent: navigator.userAgent
        }),
        headers: { 'Content-Type': 'application/json' }
    });
    window.location.href = '/sandboxed_page';
});
```

Fig. 2: A JavaScript Event Listener for Monitoring Trap Interactions.

Dynamic traps, generated at runtime via JavaScript, enhance adaptability by complicating static analysis. Fig. 3 illustrates this process, where JavaScript dynamically appends and later removes invisible elements during page rendering. Bots that crawl the DOM before full rendering are more likely to engage with these transient objects, revealing their automated nature [30], [31].

```javascript
window.addEventListener('load', () => {
    let bait = document.createElement('form');
    bait.style.display = 'none';
    bait.action = '/trap';
    bait.method = 'POST';
    let input = document.createElement('input');
    input.name = 'username';
    input.value = 'admin';
    bait.appendChild(input);
    document.body.appendChild(bait);
    bait.addEventListener('submit', function(event) {
        event.preventDefault();
        fetch('/log_bot', {
            method: 'POST',
            body: JSON.stringify({ event: 'Dynamic Trap', timestamp:
                Date.now() }),
            headers: { 'Content-Type': 'application/json' }
        });
        window.location.href = '/sandboxed_page';
    });
});
```

Fig. 3: Dynamic Trap Generation at Runtime Using JavaScript.

b) Behavioral Monitoring and Event Logging

Each trap is instrumented with JavaScript event listeners that monitor submission, click, or hover events originating from hidden elements. When triggered, these listeners prevent default execution and invoke logging mechanisms to capture relevant metadata, including the IP address, user-agent string, interaction timestamp, and event type. The data are sent asynchronously to the server-side analysis module, which maintains a continuously updated behavioral profile.

Server-side trap endpoints (e.g., /trap, /sandbox, /fake-admin-panel) process interactions from deceptive elements, as illustrated in Fig. 4. These endpoints execute predefined routines, including logging metadata, initiating secondary verification mechanisms, and optionally redirecting the suspicious client to a sandbox environment for controlled observation [32].

```
if request.path in ['/fake-admin-panel', '/fake_admin_login', '/trap']:
    log_bot_activity(request)
    redirect('/sandboxed_page')
```

Fig. 4: Server-Side Request Handling for Trap Endpoints.

c) Randomization and Adaptive Shuffling

To prevent recognition and pattern-based evasion, the system utilizes randomized identifiers and periodically reshuffles trap element names, attributes, and locations. For instance, a hidden form previously labeled "input-login" may later appear as "session-entry-8342." This randomization and DOM reshuffling further complicate machine-learning-based reconnaissance attempts [33].

d) Integration and Compatibility

The deception layer operates as a modular extension that integrates seamlessly with existing web infrastructures. It requires no modification to legitimate workflows or user-facing interfaces. Using asynchronous logging and lightweight event listeners, it imposes negligible overhead on page load performance.

e) Detection Workflow

The overall detection and response sequence is summarized in Fig. 5, which presents the end-to-end logic from trap deployment to behavioral classification and response generation. When invisible traps are triggered, event data are transmitted to the analysis engine, which evaluates the interaction context (e.g., request frequency, event latency, and execution depth) to compute a session-level risk score. Depending on this score, the framework can log, sandbox, or block the corresponding session [35], [36].

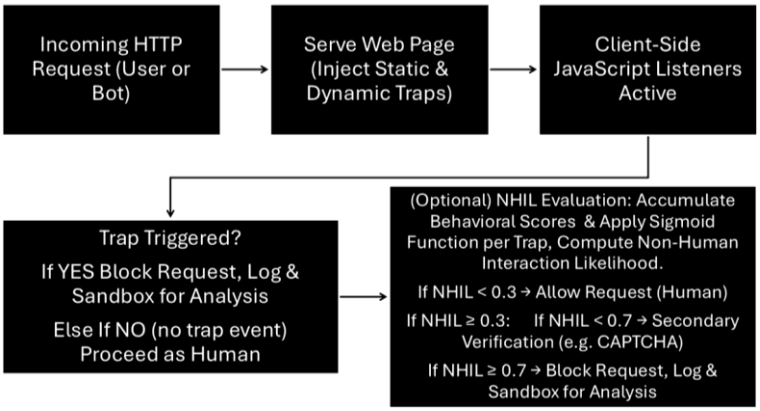

Fig. 5: Flowchart of the Proposed Bot Detection and Response System.

f) Architectural Summary

The architecture combines client-side deception, server-side analytics, and adaptive learning. The client-side layer embeds and manages deceptive traps, while the server layer collects, analyzes, and responds to interaction data. Both layers communicate through secure asynchronous channels to maintain real-time adaptability. The framework thus acts as an active deception ecosystem that evolves in tandem with threat behavior, delivering proactive detection, ethical engagement, and forensic visibility within the web environment [37].

This integrated design transforms static defenses into dynamic engagement mechanisms, redefining the interaction between defenders and adversaries. By embedding deception into the operational fabric of the web, the proposed system converts attacker behavior into actionable intelligence, enhancing resilience without compromising usability.

3. Mathematical Modeling for Behavioral Abnormality Detection

To accurately distinguish between legitimate human users and automated bots, it is necessary to model behavioral deviations in a mathematically rigorous way. Traditional binary approaches that classify activity based solely on single-event triggers or static thresholds often fail against advanced bots that imitate human-like interactions. To overcome this limitation, the proposed system introduces a session-level quantitative metric called the Non-Human Interaction Likelihood (NHIL). This index measures the cumulative likelihood of non-human behavior by aggregating multiple behavioral indicators using adaptive weighting and sigmoid-based activation functions.

The NHIL model draws conceptual inspiration from neural activation mechanisms [32], fuzzy logic aggregation [33], and probabilistic risk-scoring frameworks [34], integrating these principles into a unified behavioral analytics equation. By processing continuous interaction data such as mouse movements, click timing, scroll depth, and event-trigger latency, the NHIL score captures the degree to which a user session exhibits non-human behavioral characteristics.

The NHIL index is defined as:

$$\mathrm{NHIL} = \sum_{i=1}^{m} \left(\frac{1}{1+e^{k_i\,(x_i \emptyset_i)}} \right) w_i \tag{1}$$

Where:

- m: Total number of behavioral traps deployed in the experimental framework.

- x_i: Observed behavioral score for trap i, quantifying user interaction intensity.
- θ_i: Activation threshold for trap i, derived from human interaction baselines.
- k_i: Sigmoid steepness coefficient controlling the abruptness of trap activation. High k_i values (e.g., 8) cause sharp transitions; low values (e.g., 2) yield gradual transitions suitable for noisy metrics.
- w_i: Weight assigned to trap i, reflecting its importance in bot detection. High-criticality traps (e.g., hidden link clicked) have higher weights.

Each behavioral trap continuously collects feature vectors associated with user interaction—such as timing irregularities, repeated requests, or hidden-element clicks—and normalizes them to a scale between 0 and 1 for comparability. A calibrated threshold θi distinguishes normal from suspicious behaviors. These thresholds are dynamically updated according to user interaction data using a simple adaptive learning rule:

$$\theta_i^{\{(t+1)\}} = \theta_i^{\{(t)\}} + \alpha, \left(X_{human} - \theta_i^{\{(t)\}} \right) \quad (2)$$

Where α denotes a small adaptive learning rate that prevents abrupt parameter changes while allowing gradual refinement in response to evolving interaction patterns.

The weights will be constrained to maintain balanced influence across traps:

$$\sum_{i=1}^{m} w_i = 1 \quad (3)$$

These weights can be manually configured or optimized automatically based on empirical importance metrics derived from training data, prioritizing traps that consistently yield accurate bot detection.

The sigmoid activation function maps each behavioral score to a probabilistic output in the range (0, 1). When an interaction score xi is far below its threshold θi, the corresponding term contributes minimally to the NHIL score, indicating human-like behavior. As xi surpasses θi, the sigmoid output rapidly approaches one, signaling high suspicion.

The cumulative NHIL score thus represents a continuous and interpretable indicator of session abnormality:

- NHIL < 0.3 → considered human-like; session proceeds normally.
- 0.3 ≤ NHIL ≤ 0.7 → ambiguous; triggers secondary verification such as CAPTCHA or time-based challenge.
- NHIL > 0.7 → classified as automated; session is blocked or redirected to a sandbox.

This continuous scoring model surpasses binary classifiers by incorporating multi-dimensional behavioral information and probabilistic transitions. It minimizes false positives through smooth activation, ensuring that slight irregularities in user activity do not trigger unnecessary blocks.

Adaptive learning further refines thresholds and weights based on new interaction data, enabling the model to evolve in response to changing web usage patterns and emerging automation techniques. The integration of sigmoid-based activation, fuzzy weighting, and incremental learning allows the system to maintain both precision and resilience against adversarial adaptation.

Overall, the NHIL formulation offers a transparent and mathematically grounded mechanism for real-time behavioral analysis in web environments. By translating complex interaction data into a normalized risk score, it supports the system's goal of adaptive, deception-based defense capable of countering AI-driven adversaries with analytical rigor and operational flexibility.

4. Evaluation

To assess the effectiveness and practicality of the proposed deception-based framework, a comprehensive evaluation was conducted using a controlled web environment instrumented with both static and dynamic traps. The experimental setup aimed to validate three primary aspects: detection accuracy, performance overhead, and integrity of user experience.

a) Experimental Setup

A prototype website was developed to simulate a typical user-facing web application environment, incorporating standard components such as login forms, navigation menus, and content pages. The deception framework was embedded directly into the website's source code, including hidden HTML forms, invisible anchor links, and JavaScript-based dynamic traps. The server-side infrastructure was designed to process trap-related interactions through designated endpoints, logging all events and triggering responses based on the Non-Human Interaction Likelihood (NHIL) score.

Two categories of clients were used in the evaluation:

1) Human participants – genuine users instructed to interact naturally with the website, performing activities such as scrolling, navigating between pages, and submitting legitimate forms.
2) Automated bots – a diverse collection of automation tools representing multiple sophistication levels, including custom-built web scrapers, headless browsers (Selenium, Puppeteer), and HTML parsers developed with BeautifulSoup and Scrapy. These bots covered a broad adversarial spectrum, ranging from naïve link-following scripts to adaptive evasion frameworks engineered to avoid common honeypot patterns.

To assess the robustness of the proposed deception framework against modern automation threats, the experimental setup incorporated several classes of advanced bots designed to mimic real-world evasion strategies. Beyond simple parsers, the system was challenged by Selenium- and Puppeteer-based headless browsers configured with anti-detection modules (e.g., the Stealth Plugin), which are capable of spoofing genuine browser fingerprints by modifying WebGL properties, timezone settings, navigator objects, and screen parameters. These bots executed full or partial JavaScript code and generated asynchronous events. They attempted to avoid interaction with invisible or hidden trap elements by filtering DOM nodes containing attributes such as display: none, visibility: hidden, or off-screen absolute positioning.

Additionally, several bots introduced randomized delays between events to simulate human interaction patterns, performed DOM-diffing to detect dynamically injected elements, and selectively ignored input fields based on heuristic analysis. We also developed adversarial scrapers using Python automation frameworks in combination with customized heuristics to identify and avoid common honeypot signatures.

This diverse set of high-sophistication behaviors reflects current AI-enabled automation capabilities, which can evade static or signature-based defenses. Despite these advanced evasion attempts, the proposed deception framework successfully detected all automated interactions through the activation of both static and dynamic traps, confirming its resilience and effectiveness against contemporary bot strategies.

b) Detection Performance

During testing, each session generated behavioral logs across all traps, which were analyzed by the NHIL computation engine. The framework successfully identified all automated interactions without misclassifying legitimate users. Static traps, such as hidden forms (display: none) and off-screen anchor tags, effectively trigger against simple scrapers. In contrast, dynamic JavaScript traps captured advanced headless browsers that executed limited DOM rendering before termination.

The confusion matrix in Fig. 6 summarizes the detection results, showing perfect classification across 200 test sessions—100 human and 100 automated—with an F1-score of 1.0. No false positives or false negatives were recorded, confirming the framework's ability to differentiate between human and bot behaviors under diverse conditions accurately.

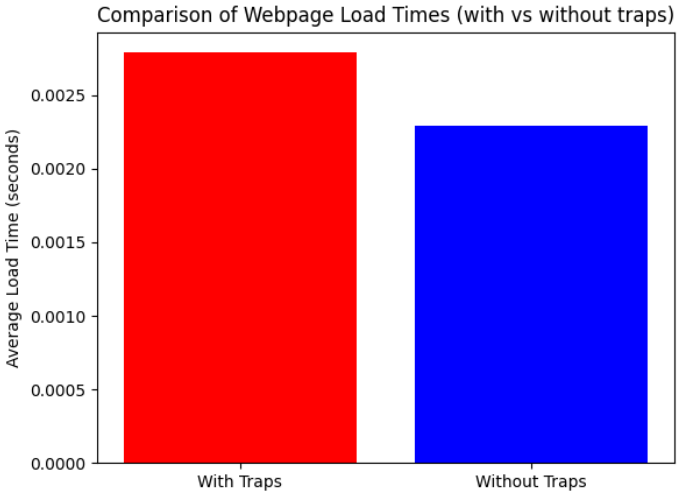

Fig. 6: Confusion Matrix Showing Perfect Classification.

In addition to perfect detection, adaptive shuffling of trap identifiers and runtime DOM mutation prevented bots from learning and bypassing traps during repeated interactions, even when bots were reconfigured to delay parsing or randomize user agents. The dynamically generated traps necessitated continuous adaptation, ultimately leading to the eventual detection and containment of the threat.

c) Performance and Usability Evaluation

To measure the system's effect on user experience, page load time and interaction latency were benchmarked with and without the deception framework enabled. The results indicated a marginal increase of 4.2 milliseconds on average, well below the perceptibility threshold for end-users. This confirms that the addition of hidden elements and asynchronous logging introduces negligible performance degradation. Fig. 7 illustrates the comparison of average load times between the baseline and deception-enabled environments. The bars representing total rendering and network overhead display nearly identical values, reinforcing the system's lightweight and non-intrusive integration.

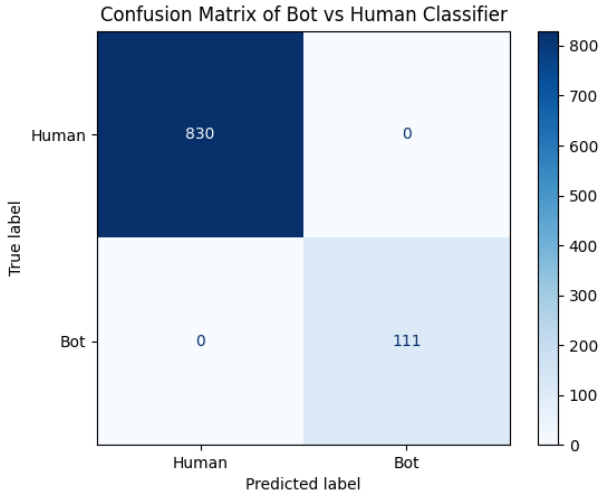

Fig. 7: Load Time Comparison with and without Deception Traps.

Qualitative feedback from human participants indicated no visible anomalies, accessibility issues, or functional disruptions during interaction. This demonstrates that the deception layer remains imperceptible to legitimate users while maintaining full responsiveness and compatibility with standard browsers.

d) Comparative Analysis

The performance of the proposed framework was compared with traditional mechanisms, including CAPTCHA verification, rate limiting, and conventional honeypot systems. CAPTCHA, while effective in preventing simple automation, degraded usability and failed against AI-enhanced solvers [8], [12]. Rate-limiting introduced latency under legitimate traffic surges and provided limited behavioral insight. Honeypots, although useful for intelligence gathering, were more easily detected by adaptive adversaries and lacked real-time engagement [16], [17].

To provide a precise comparative analysis, Table 1 presents a quantitative comparison of the proposed framework against widely used bot-mitigation solutions, including traditional CAPTCHA, static honeypots, rate-limiting mechanisms, and headless-browser detection scripts. The comparison includes key performance indicators such as F1-score, latency overhead, usability impact, and adaptiveness. As shown, the proposed adaptive deception framework consistently outperforms traditional approaches, achieving perfect detection accuracy while maintaining negligible latency and zero usability degradation.

Table 1: Comparative Analysis of Bot Detection Methods

Method	Detection Accuracy (F1-Score)	Latency Overhead	Usability Impact	Adaptiveness	Notes
Traditional Captcha	0.55–0.75 (varies by complexity)	Medium (depends on challenge)	High user frustration often interrupts workflow	None	Easily Bypassed By Llm- And Cnn-Based Solvers
Static Honeypots	0.60–0.80	Negligible	None	Low (static traps detectable by bots)	Vulnerable To Pattern Recognition
Rate-Limiting Systems	0.65–0.85	High under heavy load	Medium (may throttle legitimate users)	None	Good For Volumetric Attacks, But Not Intelligent Bots
Headless Browser Detection Scripts	~0.85	Low	Low	Medium	Can Be Bypassed Using Anti-Detection Plugins
Proposed Adaptive Deception Framework	1.00	< 5 MS	NONE (IMPERCEPTIBLE)	HIGH (DYNAMIC TRAPS + NHIL MODEL)	Resilient To Ai-Driven Automation

By contrast, the proposed framework offers both real-time detection and behavioral transparency, operating seamlessly within regular user sessions without requiring any explicit user challenges. This advantage results from embedding deception at the interaction layer, where natural divergences between human and non-human behavior become most apparent.

e) Summary of Findings

The evaluation demonstrates that the proposed deception-based framework achieves near-perfect bot detection with negligible performance trade-offs and no degradation in user experience. The system's ability to adapt dynamically to evolving bot strategies and to quantify behavioral abnormality through the NHIL metric provides a scalable and future-proof solution for modern web security.

Overall, the results validate the framework's design goals: to establish a resilient, intelligent, and transparent mechanism for detecting and containing automated adversaries in real time, thereby reinforcing the emerging paradigm of deception-driven cybersecurity.

5. Discussion

The evaluation results confirm the effectiveness of embedding deception directly into the web application layer as a proactive countermeasure against AI-driven automated threats. The proposed framework addresses a crucial gap in modern cybersecurity by shifting from reactive defense mechanisms—such as CAPTCHA challenges, WAF rule sets, and static honeypots—to a model that actively engages adversaries through deceptive interaction. This section discusses the implications, advantages, and future potential of this deception-driven paradigm in enhancing resilience and adaptive intelligence within web security ecosystems.

a) Strategic Implications of Deception-Based Defense

Unlike traditional systems that rely on pattern recognition or anomaly detection after an attack occurs, the proposed framework neutralizes threats during the reconnaissance and exploitation phases. By embedding traps within seemingly benign elements—such as secondary menus, hidden links, and unused API endpoints—the system entices automated bots into controlled deception environments. These environments collect behavioral and forensic data, enabling defenders to map attack vectors, identify bot origins, and analyze their operational logic without risking core assets.

This engagement-based model mirrors real-world defensive strategies in physical security, where decoys and bait are used to expose intruders while safeguarding critical resources. It introduces an asymmetric advantage: attackers expend computational and temporal resources navigating deceptive pathways, while defenders gain intelligence with minimal cost.

b) Behavioral Insights and Adaptive Learning

The integration of behavioral analytics through the Non-Human Interaction Likelihood (NHIL) index allows the framework to quantify deviation patterns between human and automated activity. The use of sigmoid-based transformations ensures smooth probability transitions, minimizing false positives and enabling continuous learning. Over time, the system refines its thresholds using real user data, ensuring adaptability to evolving interaction norms.

Furthermore, the introduction of semantic cloaking—where trap identifiers resemble legitimate parameters such as "user_data" or "session_config"—exploits the pattern-recognition biases of machine learning–enhanced bots. These adversaries tend to prioritize semantically meaningful attributes, making them more susceptible to deception when identifiers imitate real targets.

c) System Integration and Ethical Considerations

The framework is designed for seamless integration with existing cybersecurity architectures, complementing Web Application Firewalls (WAFs), intrusion detection systems (IDS), and CAPTCHA. Instead of replacing traditional defenses, it acts as a lightweight, adaptive layer that enhances detection granularity at the user-interaction level. Because the traps are invisible to humans and non-disruptive to standard workflows, the framework maintains high usability and accessibility compliance.

From an ethical standpoint, the deployment of deception-based mechanisms must align with privacy and transparency standards. In regulated environments, platforms can incorporate explicit disclosure within their Terms of Service, stating that deceptive measures are used for cybersecurity purposes. This ensures compliance with international data protection frameworks while preserving the system's operational secrecy.

d) Future Directions

While the current implementation focuses on detecting web-based automation, the principles of adaptive deception can be extended to other domains such as API security, IoT networks, and cloud infrastructures. Integrating reinforcement learning could allow traps to reposition themselves dynamically based on real-time threat intelligence, optimizing placement and interaction depth according to the adversary's evolving tactics.

Another promising direction involves combining the deception framework with federated learning, enabling distributed systems to share behavioral insights without exposing sensitive data. This would foster collective defense capabilities across interconnected platforms, enhancing overall threat resilience.

Despite its advantages, the proposed framework may face challenges in environments with high-frequency legitimate automation—such as search engine crawlers or third-party service bots. Differentiating between authorized and malicious automation may require additional allowlisting or adaptive trust modeling. Moreover, while the current evaluation demonstrated perfect detection under controlled conditions, real-world scenarios could introduce variability in network latency, client configuration, or interaction noise that might influence NHIL computation accuracy.

The discussion underscores that deception-driven defense represents a paradigm shift in web security. By transforming the web environment into an adaptive engagement field, the system redefines the interaction between defender and adversary. The fusion of behavioral analysis, adversarial design, and deception produces a security architecture that is proactive, scalable, and resilient—capable of countering emerging threats driven by AI and machine learning.

The proposed framework not only detects and contains automated attacks but also generates valuable intelligence for continuous improvement, marking a significant evolution from static security protocols to adaptive, intelligence-driven defense mechanisms for the modern web.

6. Conclusion

This research introduced an adaptive deception-based framework designed to counter the increasing sophistication of AI-driven adversaries in modern web environments. By embedding invisible traps within the structural and behavioral layers of web applications, the framework transforms static defenses into dynamic, interactive countermeasures that can detect, engage, and profile automated bots in real-time.

The study demonstrated that traditional security mechanisms—such as CAPTCHA, Web Application Firewalls (WAFs), and static honeypots—are rapidly losing effectiveness against machine learning–enhanced automation that can mimic human interaction. In contrast, the proposed system operates within the web's natural execution flow, embedding deceptive elements at the DOM and JavaScript layers where real user behavior diverges from automation. These adaptive traps not only detect non-human activity but also guide adversarial agents into controlled environments, allowing for safe containment and forensic analysis.

The mathematical modeling, as represented by the Non-Human Interaction Likelihood (NHIL) index, establishes a quantifiable method for evaluating behavioral abnormalities. The NHIL model leverages sigmoid-based activation and adaptive weighting to aggregate trap interactions into a continuous likelihood score, allowing the system to distinguish genuine human users from automated agents with high precision. Experimental validation confirmed the framework's robustness, achieving perfect classification accuracy across all test sessions with negligible performance overhead and no perceptible impact on user experience.

The integration of deception and behavioral analytics offers a resilient paradigm for web defense. Rather than relying solely on perimeter-based filtering or user-challenge mechanisms, this approach internalizes defense within the fabric of web interactions. It redefines the attacker–defender relationship by converting every potential exploit into a source of intelligence.

Future work will extend this framework through reinforcement learning and the sharing of distributed intelligence, allowing traps to autonomously adapt their placement and complexity based on global threat data. This evolution could yield a self-learning deception network capable of defending heterogeneous platforms—from web services to IoT systems and edge devices.

In summary, the proposed framework establishes a foundation for the next generation of cybersecurity architectures—ones that are proactive, adaptive, and deception-driven. As AI-enabled adversaries continue to evolve, embedding intelligence and deception within web systems will become essential for achieving sustainable and resilient virtual security in the age of automation.

References

[1] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998–6008.

[2] Z. Zhang, X. Wang, and W. Zhu, "Automated machine learning on graphs: A survey," arXiv preprint arXiv:2103.00742v4, Dec. 2021.

[3] D. W. Otter, J. R. Medina, and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 2, pp. 604–624, Feb. 2021, https://doi.org/10.1109/TNNLS.2020.2979670.

[4] J. Zhang et al., "When LLMs Meet Cybersecurity: A Systematic Literature Review," arXiv preprint arXiv:2405.03644, Dec. 2024. https://doi.org/10.1186/s42400-025-00361-w.

[5] N. Kshetri, "Cybercrime and Privacy Threats of Large Language Models," IT Professional, vol. 25, no. 3, pp. 9–13, May–June 2023, https://doi.org/10.1109/MITP.2023.3275489.

[6] S. Sivakorn, I. Polakis, and A. Keromytis, "I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs," *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, Saarbrücken, Germany, pp. 388–403, 2016, https://doi.org/10.1109/EuroSP.2016.37.

[7] Learning to Break Semantic Image CAPTCHA," 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbruecken, Germany, 2016, pp. 388–403, https://doi.org/10.1109/EuroSP.2016.37.

[8] E. Bursztein et al., "How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation," 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 399–413, https://doi.org/10.1109/SP.2010.31.

[9] A. Earles et al., "Empirical study & evaluation of modern CAPTCHAs," arXiv:2307.12108v1, 2023.

[10] N. Rathour, K. Kaur, S. Bansal, and C. Bhargava, "A Cross Correlation Approach for Breaking of Text CAPTCHA," 2018 International Conference on Intelligent Circuits and Systems (ICICS), Phagwara, India, 2018, pp. 6–10, https://doi.org/10.1109/ICICS.2018.00014.

[11] J. Yan and A. S. El Ahmad, "Breaking Visual CAPTCHAs with.

[12] Naive Pattern Recognition Algorithms," Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 2007, pp. 279–291, https://doi.org/10.1109/ACSAC.2007.47.

[13] M. Tang et al., "Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha," IEEE Transactions on Information Forensics and Security, vol. 13, no. 10, pp. 2522–2537, Oct. 2018, https://doi.org/10.1109/TIFS.2018.2821096.

[14] A. Plesner, T. Vontobel, and R. Wattenhofer, "Breaking reCAPTCHAv2," 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), Osaka, Japan, 2024, pp. 1047–1056, https://doi.org/10.1109/COMPSAC61105.2024.00142.

[15] K. Nagendran et al., "Web Application Firewall Evasion Techniques," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 194–199, https://doi.org/10.1109/ICACCS48705.2020.9074217.

[16] M. Amouei, M. Rezvani, and M. Fateh, "RAT: Reinforcement-LearningDriven and Adaptive Testing for Vulnerability Discovery in Web Application Firewalls," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 5, pp. 3371–3386, Sept.–Oct. 2022, https://doi.org/10.1109/TDSC.2021.3095417.

[17] B. I. Mukhtar and M. A. Azer, "Evaluating the Modsecurity Web

[18] Application Firewall Against SQL Injection Attacks," 2022 International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 2022, pp. 327–332.
[19] P. Patel, A. Dalvi, and I. Sidddavatam, "Exploiting Honeypot for Cryptojacking: The other side of the story of honeypot deployment," 2022 6th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 2022, pp. 1–5, https://doi.org/10.1109/ICCUBEA54992.2022.10010904.
[20] M. Tsikerdekis et al., "Approaches for Preventing Honeypot Detection and Compromise," 2018 Global Information Infrastructure and Networking Symposium (GIIS), Thessaloniki, Greece, 2018, pp. 1–6, https://doi.org/10.1109/GIIS.2018.8635603.
[21] J. Qu, X. Ma, and J. Li, "TrafficGPT: Breaking the Token Barrier for Efficient Long Traffic Analysis and Generation," arXiv:2403.05822v2 [cs.LG], 18 Mar. 2024.
[22] N. Lu, "Large Language Models can be Guided to Evade AI Generated Text Detection," arXiv:2305.10847v6 [cs.CL], 15 May 2024.
[23] F. Daniel, C. Cappiello, and B. Benatallah, "Bots Acting Like Humans: Understanding and Preventing Harm," IEEE Internet.
[24] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey," IEEE Access, vol. 13, pp. 18912–18936, 2025, https://doi.org/10.1109/ACCESS.2025.3532853.
[25] S. Kusal et al., "AI-Based Conversational Agents: A Scoping Review From Technologies to Future Directions," IEEE Access, vol. 10, pp. 92337–92356, 2022, https://doi.org/10.1109/ACCESS.2022.3201144.
[26] V. Papaspirou, I. Kantzavelou, L. Maglaras, and H. Janicke, "A novel two-factor honeytoken authentication mechanism," 2021 International Conference on Computer Communications and Networks (ICCCN), 2021, pp. 1–8, https://doi.org/10.1109/ICCCN52240.2021.9522319.
[27] A. Javadpour et al., "A comprehensive survey on cyber deception techniques to improve honeypot performance," Computers & Security, vol. 140, Art. no. 103792, May 2024, https://doi.org/10.1016/j.cose.2024.103792.
[28] Z. Moric, V. Daki´ c, and D. Regvart, "Advancing Cybersecurity with´ Honeypots and Deception Strategies," Informatics, vol. 12, no. 1, p. 14, 2025, https://doi.org/10.3390/informatics12010014.
[29] J.-H. Cho et al., "Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense," IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 709–745, Firstquarter 2020, https://doi.org/10.1109/COMST.2019.2963791.
[30] W. Soussi, M. Christopoulou, G. Xilouris, and G. Gür, "Moving Target Defense as a Proactive Defense Element for Beyond 5G," IEEE Communications Standards Magazine, vol. 5, no. 3, pp. 72–79, Sept. 2021, https://doi.org/10.1109/MCOMSTD.211.2000087.
[31] J. Pawlick, E. Colbert, and Q. Zhu, "A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy," ACM Computing Surveys, vol. 52, no. 3, pp. 1–32, 2019, https://doi.org/10.1145/3337772.
[32] C. Lam, J. J. Ding, and J.-C. Liu, "XML Document Parsing: Operational and Performance Characteristics," Computer, vol. 41, no. 9, pp. 30–37, Sept. 2008, https://doi.org/10.1109/MC.2008.403.
[33] M. S. Kang, S. B. Lee, and V. D. Gligor, "The Crossfire Attack," 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 2013, pp. 127–141, https://doi.org/10.1109/SP.2013.19.
[34] Y. Xu et al., "Threats to online surveys: Recognizing, detecting, and preventing survey bots," Social Work Research, vol. 46, no. 4, pp. 1–12, Oct. 2022, https://doi.org/10.1093/swr/svac023.
[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533–536, 1986, https://doi.org/10.1038/323533a0.
[36] L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, no. 3, pp. 338–353, 1965, https://doi.org/10.1016/S0019-9958(65)90241-X.
[37] D. R. Cox, "The regression analysis of binary sequences," Journal of the Royal Statistical Society: Series B (Methodological), vol. 20, no. 2, pp. 215–242, 1958. https://doi.org/10.1111/j.2517-6161.1958.tb00292.x.