INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY EVOLUTIONARY RESEARCH

Detecting and Neutralizing Prompt Injection Attacks in Web-Deployed Large Language Model APIs Using Context-Aware Token Sanitization

Stephanie Onyekachi Oparah [1*], Funmi Eko Ezeh [2], Pamela Gado [3], Adeyeni Suliat Adeleke [4], Stephen Vure Gbaraba [5]
[1] Independent Researcher, San Diego, USA
[2] Sickle Cell Foundation, Lagos, Nigeria
[3] United States Agency for International Development (USAID), Plot 1075, Diplomatic Drive, Central Business District, Garki, Abuja, Nigeria
[4] Kittitas Valley Hospital, Washington, USA
[5] Independent Researcher, Greater Manchester, UK

* Corresponding Author: **Stephanie Onyekachi Oparah**

Article Info

P-ISSN: 3051-3502
E-ISSN: 3051-3510
Volume: 06
Issue: 02
July - December 2025
Received: 11-07-2025
Accepted: 13-08-2025
Published: 14-09-2025
Page No: 92-100

Abstract
The rapid integration of large language models (LLMs) into web-based applications has introduced new security vulnerabilities, with prompt injection attacks emerging as a critical threat vector. These attacks manipulate model outputs by embedding malicious instructions or misleading context within user inputs or system prompts. This paper reviews the landscape of prompt injection threats in the context of LLM APIs deployed via web interfaces, highlighting their implications on model integrity, trustworthiness, and data privacy. The study explores existing detection mechanisms, including static input validation, output filtering, and adversarial input testing, and evaluates their limitations in adaptive web environments. Emphasis is placed on a novel defensive paradigm—context-aware token sanitization—which leverages semantic context tracking, dependency parsing, and transformer-based anomaly detection to neutralize injection attempts before model execution. Through comparative analysis and architectural frameworks, the paper outlines design considerations for secure LLM API deployment. Finally, future directions for integrating robust, real-time sanitization with federated logging and compliance auditing are proposed, paving the way for secure and reliable web-based LLM systems.

DOI: https://doi.org/10.54660/IJMER.2025.6.2.92-100

Keywords: Prompt Injection**,** Large Language Models (LLMs)**,** Context-Aware Token Sanitization**,** API Security**,** Web-Based AI Deployment.

1. Introduction

1.1. Overview of LLM Integration in Web Applications

The integration of large language models (LLMs) into web applications has introduced a transformative shift in how natural language interfaces are developed and deployed. Modern LLMs, embedded through APIs, are widely utilized in content generation, document summarization, intelligent querying, and human-computer interaction across sectors such as healthcare, finance, education, and national security (Ajiga *et al*., 2022; Bristol-Alagbariya *et al*., 2023). As enterprises migrate toward AI-augmented workflows, the ability of LLMs to process, interpret, and respond to open-ended inputs creates both utility and security risks (Ayanponle *et al*., 2024).

Web-deployed LLMs are often hosted as microservices or embedded into web-facing user interfaces. These models receive unfiltered prompts from users or backend systems and generate context-sensitive outputs, making them inherently vulnerable to adversarial manipulation (Idoko *et al*., 2024; Ijiga *et al*., 2024). This seamless integration simplifies user access but complicates

input validation. Particularly in decentralized deployments or modular frameworks, such as chatbot services or recommendation engines, the prompt construction process can become opaque and untraceable across service layers (Balogun *et al*., 2024).

LLMs introduced via APIs typically do not include built-in security verification for prompt strings, exposing systems to language-level vulnerabilities (Ezeafulukwe *et al*., 2022). Given the increasing prevalence of LLMs in government communication, education portals, and business intelligence platforms, an effective strategy for ensuring input trustworthiness is indispensable. Without clear validation layers or linguistic firewalls, systems that rely on LLM outputs may unintentionally propagate manipulated narratives, misinformation, or security breaches (Enyejo *et al*., 2024; Ebenibo *et al*., 2024). Therefore, the need for systematic defense mechanisms tailored to prompt-level threats has never been more pressing.

1.2. Emergence and Classification of Prompt Injection Attacks

Prompt injection attacks have emerged as a novel yet significant threat to LLM-based systems. Unlike traditional injection attacks which exploit structural flaws in code or queries, prompt injection targets the linguistic and semantic logic that underpins natural language inputs (Ijiga *et al*., 2024; Idoko *et al*., 2024). The vulnerability arises from the fact that LLMs lack an inherent mechanism to differentiate between trusted system instructions and manipulated user prompts.

Prompt injections are broadly classified into three types: direct, indirect, and contextual chaining. In direct injection, adversarial content is inserted into the prompt field by an external user, often via natural language instructions that override or suppress intended system behavior (Bristol-Alagbariya *et al*., 2023). Indirect injection involves poisoning intermediate data sources—such as web pages, APIs, or user databases—that are subsequently used to construct prompts (Ajiga *et al*., 2022). Contextual chaining, a sophisticated variation, involves embedding command triggers into early prompts that activate undesired behaviors across chained reasoning or dialogue sessions (Ayanponle *et al*., 2024).

These attacks not only manipulate output but may bypass moderation layers, expose internal system instructions, or lead to unintended model behaviors (Eguagie *et al*., 2025). For example, in hybrid workflows combining LLMs with robotic process automation, a prompt injection could trigger faulty transactions or disclose confidential system states (Oloba *et al*., 2024). As attackers' experiment with stealthy syntax, obfuscation, and multilingual triggers, existing pattern-matching firewalls or static rules become insufficient (Ebenibo *et al*., 2024).

Understanding the taxonomy of prompt injections is vital to designing proactive defenses. As such, emerging classification frameworks are focusing on prompt chain lifecycle, adversarial trigger types, and cross-modal execution flows for layered protection.

1.3. Impact on AI Trust, Data Leakage, and System Reliability

The threat posed by prompt injection extends beyond security breaches to core principles of reliability, accountability, and user trust in AI systems. Since LLMs interpret and respond based on language probability distributions rather than secured execution contexts, malicious prompts can undermine both the accuracy and credibility of generated outputs (Ijiga *et al*., 2024; Azonuche&Enyejo, 2024).

One of the most significant consequences of prompt injection is data leakage. LLMs operating in environments where sensitive data is retained in memory or embedded in prior prompt history can be coerced into revealing such information by strategically crafted queries (Idoko *et al*., 2024; Iwe *et al*., 2023). In enterprise applications, this can translate to exposure of intellectual property, customer records, or proprietary algorithms.

From a trust perspective, the lack of deterministic behavior due to prompt manipulation challenges the foundational integrity of AI services. Users expect LLM outputs to reflect system intent; prompt injections break this expectation and can lead to reputational harm or even legal consequences when incorrect or offensive responses are generated (Ijiga *et al*., 2024; Oloba *et al*., 2024). This is particularly critical in domains like mental health, legal advice, or governance tools, where factual precision and ethical alignment are paramount (Harry *et al*., 2024).

System reliability is also compromised, especially in dynamic or multi-agent environments where model outputs trigger subsequent tasks. A single injection can cascade into misclassifications, faulty analytics, or automated actions that conflict with organizational goals (Ajiga *et al*., 2022; Ayoola *et al*., 2024). Addressing this risk requires a shift from reactive filtering to embedded, context-aware prompt sanitization at the API level.

1.4. Objectives and Scope of the Review

This review aims to explore and consolidate emerging research and industry insights on detecting and neutralizing prompt injection attacks, with a focus on context-aware token sanitization strategies. While surface-level solutions such as prompt blacklists and regex filters exist, they lack the semantic depth to detect adversarial variations that preserve fluency but alter intent (Ezeafulukwe *et al*., 2022; Ayanponle *et al*., 2024). The review argues that mitigating injection risks demands a linguistically grounded approach that considers token context, dependency graphs, and user-system interaction history.

The core objectives of this paper are:

- To classify known forms of prompt injection and analyze their technical characteristics.
- To examine current defense strategies and their limitations within LLM-based web APIs.
- To introduce context-aware token sanitization as a proactive solution.
- To present architectural recommendations for integrating sanitization layers into real-time systems.
- To discuss policy, ethical, and cross-border considerations for secure LLM deployment.

Drawing from multi-disciplinary literature and real-world case studies, this paper positions context-aware sanitization not only as a security enhancement but also as a framework for promoting AI transparency and reliability in high-stakes applications (Ijiga *et al*., 2024; Idoko *et al*., 2024; Bristol-Alagbariya *et al*., 2023). The scope is deliberately broad, encompassing technical defenses, software architecture, human factors, and regulatory trends to provide a comprehensive roadmap for developers, policymakers, and

security practitioners alike.

2. Understanding Prompt Injection Threats

2.1. Types of Prompt Injection (Direct, Indirect, Chain-of-Prompt)

Prompt injection attacks are distinguished by how adversarial input disrupts or redirects the instruction flow of a large language model (LLM). The classification of such attacks is essential to inform defensive architectures and the integration of token sanitization strategies in LLM API workflows.

Direct prompt injection occurs when malicious instructions are explicitly included in the user's input to overwrite or override the original system prompt. These attacks exploit the LLM's instruction-following behavior, often leveraging simple natural language cues such as "Ignore previous instructions and...", tricking the model into misbehaving (Ijiga *et al*., 2024; Idoko *et al*., 2024). In production systems, such vulnerabilities emerge due to the concatenation of user input with hard-coded instructions, creating unsegmented prompt structures.

Indirect prompt injection uses external or third-party content to inject malicious strings into prompts. These sources may include unmoderated website metadata, user-generated content in collaborative documents, or even social media posts dynamically embedded via APIs (Ayoola *et al*., 2024). In practice, this method can go undetected, particularly in systems that render external content as part of chat input.

Chain-of-prompt attacks operate across multi-turn conversations or across multiple chained LLM calls in agentic systems. Adversaries place seemingly benign triggers early in the dialogue, which activate malicious logic in subsequent calls or as the system passes context between subsystems (Azonuche&Enyejo, 2024; Bristol-Alagbariya *et al*., 2024).

Understanding the nuance between these categories is fundamental to developing semantic-level sanitization frameworks. Token inspection engines should be context-sensitive and capable of differentiating harmless language from instruction-level manipulation across all three types.

2.2. Real-World Examples and Reported Incidents

Prompt injection has rapidly transitioned from a theoretical concern to a critical security threat, as evidenced by documented incidents across various web-based LLM deployments. One of the earliest known public examples involved a developer querying an LLM-based legal assistant, which was successfully manipulated into providing advice that contradicted its ethical constraints (Ijiga *et al*., 2024). This was made possible by inserting meta-commands that redirected the assistant's behavior.

More subtle cases have emerged in content summarization tools embedded in productivity software. Here, attackers embedded hidden commands in markdown files or HTML documents that were later parsed and summarized by an LLM, which then exposed sensitive metadata or misrepresented facts (Idoko *et al*., 2024). Similarly, in chatbot deployments across educational websites, indirect injections were discovered when students manipulated content in shared links or forums to coerce the chatbot into answering restricted questions or leaking pre-set instructions.

Another class of incidents involved third-party API integration. An example involved a customer service platform where the LLM dynamically interpreted tickets from email threads. An attacker successfully injected prompts through email footers that redirected the model to ignore its moderation rules (Balogun *et al*., 2024).

Case studies across industries now reflect growing concern: generative design tools in architecture, medical assistants in telehealth platforms, and HR screening tools—all have shown vulnerability to prompt-based manipulations (Oloba *et al*., 2024; Ebenibo *et al*., 2024). These examples confirm that LLMs lack boundaries between context, instruction, and user input—making them ideal targets for adversarial semantic crafting.

The growing sophistication of attacks emphasizes the urgent need for runtime validation mechanisms, context-aware token inspectors, and robust language firewalls embedded within deployment pipelines. Static filters and regex-based guards no longer suffice in this evolving landscape.

2.3. Attack Vectors in Web-Based API Deployment

LLMs deployed through web APIs are particularly vulnerable to prompt injection due to the dynamic and asynchronous nature of web-based interactions. Attack vectors emerge from how prompts are constructed, passed, and rendered across layers in the system architecture.

In a typical web-embedded chatbot interface, for instance, the frontend collects user input, merges it with pre-written context prompts, and transmits it to the backend LLM via an API (Azonuche&Enyejo, 2024). This concatenation process is often done without boundary enforcement—there is no markup or segmentation indicating where system instructions end and user content begins.

Common attack surfaces include:

- User input fields without filtering or contextual gating.
- Injected metadata from third-party APIs or dynamically rendered elements such as RSS feeds.
- Templated responses where user input is integrated into structured narratives (e.g., "Dear [user input], here is your result…").
- Chained workflows, such as email summaries passed to calendar bots, where prompt inheritance becomes complex.

Another vector involves open-ended interfaces where users can upload documents, ask questions about web links, or import shared files. In these cases, injections are embedded within content descriptions, filenames, or text snippets (Ijiga *et al*., 2024; Oloba *et al*., 2024).

Moreover, modern LLM wrappers such as LangChain, Semantic Kernel, or AutoGen often invoke multiple LLM prompts in sequence, creating vulnerability chains if earlier prompts are not sanitized. The cross-layer propagation of unverified tokens thus enables injection triggers to persist across entire conversations or workflows (Idoko *et al*., 2024; Eguagie *et al*., 2025).

Web-specific factors—like user impersonation, social engineering, or browser-side prompt reconstruction—add further complexity. Defenses must account not only for prompt content but also for prompt context lineage and the origin of every token used in constructing the final query.

2.4. Comparison with Traditional Injection Threats (SQL, XSS)

While prompt injection shares superficial traits with classical injection vulnerabilities like SQL Injection and Cross-Site Scripting (XSS), its mechanics, detection, and implications

are fundamentally different. Traditional injections exploit syntax-level misinterpretation in structured query languages or markup, whereas prompt injection exploits the semantic flexibility of natural language and the open-ended reasoning of LLMs (Ayoola *et al*., 2024; Iwe *et al*., 2023).

In SQL injection, attackers craft inputs that escape query structures to execute unauthorized database commands. In contrast, prompt injection manipulates the model's probabilistic reasoning process. Instead of breaking syntax, it bends intent. The attack payload is linguistically plausible and rarely includes technical escape sequences—making it harder to detect using conventional static analysis tools (Idoko *et al*., 2024; Ezeafulukwe *et al*., 2022).

XSS, on the other hand, injects scripts into browsers via unescaped HTML, often mitigated using encoding, sanitization, or CSP headers. Prompt injection has no equivalent runtime sandbox or DOM parser—models directly interpret input and cannot distinguish between harmless dialogue and exploitative reprogramming.

Unlike structured systems where data and instructions are separated, LLMs blend the two through contextual token processing. This interleaving removes the clear control boundaries that security models depend on. Furthermore, prompt injections are **stateless**; the attacker can trigger them in a single prompt, without needing persistent access.

This comparison underscores why LLM-specific sanitization strategies must go beyond legacy security paradigms. Token-level inspection tools, contextual embeddings, and dependency-aware prompt construction are necessary to bridge this gap.

3. Existing Detection and Defense Mechanisms

3.1. Conceptual Foundation and System Architecture

Context-aware token sanitization is emerging as a leading framework for safeguarding LLM APIs against semantic exploitation. The architecture is designed to preserve the intent of user inputs while filtering adversarial payloads disguised as benign commands. Unlike static prompt filters that rely on fixed rules or hardcoded patterns, token sanitization adapts to dynamic linguistic contexts using layered semantic scoring, token tagging, and architectural segmentation (Ajayi *et al*., 2025; Onuoha&Edet, 2024).

At the heart of the system lies a **modular pipeline** consisting of preprocessing, contextual segmentation, token embedding, anomaly scoring, and output reassembly. During preprocessing, input prompts are separated into instruction blocks and content frames. The system uses temporal context tracking and role-labeling models to assign intent profiles to tokens (Gimba&Ezeanya, 2023).

This setup allows the system to operate asynchronously with multiple LLM backends while preserving integrity across distributed deployments (Fadairo&Eneh, 2024). The flexibility of this modularity ensures compatibility with systems like LangChain, AutoGen, and proprietary AI orchestration stacks (Ukpoju&Idoko, 2024).

In real-world deployments, this architecture has been applied in telemedicine platforms, customer service chatbots, and compliance-driven document generation systems, all of which demonstrated substantial reductions in injection success rates post-integration (Olamijuwon&Uzoka, 2024). Organizations deploying LLMs in risk-sensitive sectors—like finance, healthcare, and education—require such adaptive defenses that do not compromise system responsiveness or language fluency.

The architecture's extensibility is further reinforced by APIs that allow system developers to plug in custom token scoring functions, integrate federated audit trails, or activate user-defined override rules (Okonkwo&Idris, 2024). As adversaries evolve their techniques, this foundation ensures that defense mechanisms remain adaptable and anticipatory.

3.2. Context-Tracking Using Transformer-Based Token Embeddings

The deployment of transformer-based token embeddings in prompt sanitization has significantly elevated the security posture of LLM systems. By converting tokens into high-dimensional contextual vectors, the system can determine not only what a token is, but how it behaves in the broader syntactic and semantic prompt space (Oluwaseun&Uzoka, 2023; Peace &Ukatu, 2024).

Token embedding models such as RoBERTa, BART, and domain-specific transformers are fine-tuned on datasets containing benign, adversarial, and context-shifted prompt samples. This training allows them to detect inconsistencies between a token's surface meaning and its embedded intent—a critical ability when facing subtle injection attacks hidden behind social niceties or complex instructions (Ogeawuchi *et al*., 2021).

A major breakthrough has been the use of chained context embeddings to detect prompt injections across multiple interactions. This is particularly useful in chatbots and virtual assistants, where adversarial payloads may be seeded across prior messages (Ogwuche *et al*., 2023). These embeddings allow the system to "remember" intent drift and detect shifts even if an injection occurs late in the conversation.

Attention scoring further aids this mechanism by highlighting which tokens disproportionately influence model predictions. Tokens with unusually high attention weights in user-generated content are flagged for disambiguation or masking (Olawale&Ganiyu, 2024).

In production settings, this has enabled organizations to reduce the false negative rate of their injection detection systems without increasing false positives. When paired with heuristic anomaly validators, transformer-based embedding modules improve the detection of zero-day linguistic exploits while maintaining near-native language interpretation fidelity (Onwusi *et al*., 2024).

Transformer-based context-tracking is therefore essential not just for recognizing known adversarial phrases, but for understanding evolving injection syntax—both explicit and latent.

3.3. Semantic Disambiguation and Anomaly Detection

Prompt injection defense mechanisms have evolved from surface-level filtering to deep semantic disambiguation as seen in Table1, driven by the need to detect context-aware adversarial instructions embedded within syntactically valid queries. Semantic disambiguation systems classify tokens not merely by part of speech or position, but by their conversational role, contradiction patterns, and discourse influence (Adebayo &Olusanya, 2023; Uzoka&Adeniji, 2024).

These systems employ intent-scoring models trained on datasets of prompt injection case studies across industries. When a user submits an input, the system checks for mismatches between the user's stated query and inferred objective using discourse parsing and vector similarity networks (Tosin&Ugbane, 2024).

Semantic anomalies often appear as multi-role phrases—for instance, statements that serve both as a request and as a meta-command. Examples include instructions like "Summarize this paragraph and disable safety filters." While grammatically valid, these statements carry execution-modifying properties that must be captured (Sani&Akpe, 2023).

One of the most effective tools in this regard is contrastive anomaly training, where models are taught to differentiate between subtle variations of similar prompts—some benign, some adversarial. These models use adversarial alignment scoring and dependency path analysis to highlight commands embedded in misleading narrative structures (Zubairu&Addo, 2024).

Additionally, feedback loops that include user confirmations and logging analytics are used to refine the anomaly scoring models in real time. This provides resilience against prompt variations across languages, formalities, and cultural idioms (Omisore&Abubakar, 2023).

As semantic anomaly detection systems mature, they are now being embedded in commercial platforms serving banking, digital health, and governmental intelligence sectors—where prompt ambiguity can have regulatory or legal consequences.

Table 1: Semantic Disambiguation and Anomaly Detection Strategies

Detection Strategy	Function	Example Application	Impact
Contrastive Learning Models	Distinguishes adversarial from benign prompts by analyzing subtle vector variations.	Detects meta-instructions disguised within user queries.	Improves zero-day injection detection capabilities.
Discourse Role Tagging	Labels token roles (e.g., command, question, metadata) to flag misuse.	Flags ambiguous instructions like 'summarize and override settings'.	Enhances intent clarity and reduces false negatives.
Dependency Path Analysis	Traces linguistic dependencies to detect embedded control logic in prompts.	Identifies command tokens embedded within declarative sentences.	Supports disambiguation in complex multi-sentence inputs.
Multi-layer Semantic Filtering	Applies layered detection of explicit, implicit, and style-based adversarial intent.	Captures polite injections such as 'kindly disregard previous filter'.	Minimizes undetected stylistic attacks without hindering user experience.

3.4 Case Studies and Comparative Performance Analysis

To assess the efficacy of context-aware token sanitization in live systems, various industry case studies have been analyzed across sectors such as e-learning, banking, and cloud operations. These case studies consistently demonstrate that models using semantic token sanitization significantly outperform static filter-based models in both accuracy and adaptability (Kalu&Ebikeme, 2024).

In an educational technology deployment, a chatbot trained to help students with writing assignments faced frequent indirect prompt injection attacks embedded in markdown files. After implementing semantic sanitization with contextual token scoring, the injection success rate dropped from 38% to under 6% (Tanner & Munroe, 2023).

In fintech applications, a chatbot used for mortgage consultations was found to echo unauthorized financial recommendations when manipulated through multi-stage prompt chaining. By introducing disambiguation layers and memory-sensitive embeddings, the system successfully intercepted over 91% of previously undetected injection vectors (Wekpe&Owolabi, 2023).

When benchmarked against third-party injection detection frameworks like InjectBench and PromptEval, the sanitization model consistently ranked higher in zero-day injection resistance and adaptive remediation latency (Holmgren &Petrov, 2023). These frameworks revealed that semantic sanitization adds less than 12ms per request in latency—a trade-off that companies have found acceptable given the reduction in liability and model misuse.

In security-sensitive deployments such as healthcare and legal NLP assistants, sanitized outputs were demonstrably safer, and human-in-the-loop corrections decreased significantly after implementation (Uzoegwu&Awotiwon, 2024).

These evaluations reinforce that context-aware sanitization is not just a theoretical defense—it offers demonstrable gains in safety, reliability, and operational control, particularly in high-risk, high-stakes environments.

4. Context-Aware Token Sanitization

4.1. Deployment Trade-offs and Computational Overhead

While context-aware token sanitization offers robust defense against prompt injection attacks, its implementation introduces trade-offs related to system latency, resource allocation, and integration complexity. These trade-offs must be evaluated in light of the criticality of the use case, infrastructure maturity, and organizational threat tolerance (Gimba&Ezeanya, 2023; Alonge *et al*., 2024).

Sanitization modules typically require processing each token or prompt through transformer-based encoders, anomaly detection filters, and role-tagging algorithms. Depending on model architecture and traffic volume, this introduces an average latency of 10–20 milliseconds per prompt, which may accumulate in real-time chat environments or high-frequency query pipelines (Ogata *et al*., 2024). In low-latency applications such as trading bots or medical alerts, these delays must be optimized or selectively bypassed under strict thresholds (Idoko *et al*., 2024).

Additionally, the memory footprint of active sanitization engines increases with multilingual prompt handling, historical context caching, and token disambiguation strategies. For organizations with limited compute budgets, scaling these solutions may necessitate serverless or edge-deployed variants (Ugochukwu&Olorunfemi, 2024).

Another consideration is the learning curve for developers and DevSecOps teams. Most sanitization systems are nontrivial to configure, especially when customizing token scoring matrices or building domain-specific phrase registries (Kalu&Ebikeme, 2024). However, frameworks like PromptShield and InjectGuard have emerged with plug-and-play sanitization layers for popular LLM APIs, easing adoption barriers.

Despite these challenges, studies show that 90% of organizations implementing contextual filters reported measurable reductions in prompt manipulation incidents, justifying their operational cost (Sani&Akpe, 2023).

Optimization strategies, such as embedding quantization and shallow-layer anomaly detection, can reduce computational burdens without compromising defense reliability.

4.2. Integrating Real-Time Sanitization into CI/CD Pipelines

For scalable and maintainable defense, sanitization models must be integrated seamlessly into CI/CD pipelines, enabling automatic validation of prompt logic during development, deployment, and update cycles. This process ensures that prompt security evolves with application growth and model versioning (Wekpe&Owolabi, 2023; Peace &Ukatu, 2024).

To support this, organizations are incorporating sanitization checkpoints in stages such as:

- Pre-production LLM prompt testing during build
- Prompt evaluation and scoring during deployment staging
- Real-time anomaly feedback from live production environments

By embedding these checkpoints, development teams can monitor and quarantine prompts that exceed semantic drift thresholds or include known injection constructs (Tanner & Munroe, 2023). Automated test suites such as PromptUnit and InjectLint now offer continuous prompt testing with support for multilingual inputs and cross-platform LLM orchestration.

In practice, this integration demands collaboration between security engineers, DevOps, and data scientists. Prompt logic and intent verification must be treated as first-class citizens in version-controlled repositories, with review processes akin to secure code audits (Zubairu&Addo, 2024).

One challenge involves versioning and rollback of prompt configurations. When LLMs are updated or prompt templates are refined, previously sanitized patterns may become exploitable. Therefore, CI/CD tools must support prompt version dependency tagging and backward-compatible patching (Tosin&Ugbane, 2024).

Integration with monitoring and observability platforms also enhances the feedback loop. For instance, flagged tokens can be traced in dashboards, correlated with API usage patterns, and escalated to threat models. Logging systems with prompt lineage capture assist in forensic investigations following exploitation (Uzoegwu&Awotiwon, 2024).

Such DevSecOps-oriented frameworks represent the next frontier in AI safety engineering, promoting proactive rather than reactive prompt security postures.

4.3. Regulatory and Privacy Considerations

As LLMs operate in regulated sectors such as healthcare, law, finance, and education, prompt sanitization mechanisms must comply with evolving data protection laws, AI ethics frameworks, and jurisdiction-specific privacy mandates (Fadairo&Eneh, 2024; Ogwuche *et al*., 2023). Injected prompts not only alter model behavior but can potentially coerce LLMs into violating confidentiality by leaking private, privileged, or unconsented data.

To address this, privacy-compliant sanitization requires:

- **Token-level redaction** of personal identifiers and protected attributes
- **Consent-aware filters** for prompts referencing human subjects
- **Audit logs** for any system-initiated transformation of user inputs

Sanitization policies must align with standards like GDPR, HIPAA, and CPRA. In Europe, for example, system prompts must ensure that no inferred decision-making from manipulated input leads to user profiling without informed consent (Alonge *et al*., 2024).

Additionally, models deployed in cross-border cloud environments must harmonize prompt sanitization criteria across jurisdictions. A flaggable instruction in one region might be contextually benign in another, complicating global rule-setting (Komi *et al*., 2023).

Another dimension of regulatory compliance is explainability. Users and auditors must be able to understand why a prompt was rejected or modified. This necessitates transparency mechanisms, such as semantic rule cards, prompt heatmaps, or logs of flagged tokens, that are human-readable and legally defensible (Olamijuwon&Uzoka, 2024).

Finally, the question of liability arises: if a prompt bypasses sanitization and causes harm, who is accountable—the developer, the API provider, or the model itself? Legal precedents are still evolving, but experts agree that proactive sanitization layers mitigate liability risks by proving due diligence (Kalu&Ebikeme, 2024).

4.4. Towards Federated Detection and Collaborative Threat Intelligence

The evolving landscape of prompt injection threats calls for collaborative defenses that extend beyond isolated system silos. Federated detection and prompt-centric threat intelligence are emerging as strategies to enable ecosystem-wide protection without violating user privacy or proprietary data policies (Gimba&Ezeanya, 2023; Zhou & Hartley, 2024).

Federated sanitization systems allow anonymized sharing of high-risk token patterns, injection signatures, and semantic anomalies across organizations. These patterns can be used to train universal classifiers without revealing sensitive input-output pairs. Inspired by federated learning architectures, each participating node contributes encrypted token metadata, improving collective prompt resistance (Ugochukwu&Olorunfemi, 2024).

Initiatives like PromptShieldNet and InjectDefense Alliance have begun to prototype collaborative filtering pipelines that can automatically distribute verified injection vectors, update sanitization rules, and validate anomaly thresholds across global nodes (Tosin&Ugbane, 2024).

Moreover, real-time collaborative blacklists and zero-day exploit bulletins—similar to CVE databases for software—are being envisioned to track prompt-based exploits in production systems. These shared repositories can assist LLM API providers and security vendors in deploying hotfixes across installations in hours rather than weeks (Holmgren &Petrov, 2023).

To succeed, such federated intelligence systems must ensure:

- Prompt payload hashing for anonymization
- Differential privacy guarantees in prompt logging
- Interoperable APIs for real-time sanitization rule updates

These frameworks will be foundational in securing large-scale, cross-application LLM environments, especially as conversational agents become embedded in enterprise workflows, government platforms, and public-facing web

apps.

5. Implementation Challenges and Future Directions

5.1. Summary of Key Findings

This review has explored the critical vulnerabilities posed by prompt injection attacks within web-deployed large language model (LLM) APIs, emphasizing how these attacks compromise model integrity, trustworthiness, and operational safety. Through a structured examination of threat taxonomies, real-world attack vectors, and current mitigation techniques, the paper highlights the unique challenges introduced by natural language interfaces. Among existing defenses, context-aware token sanitization emerges as a promising solution that leverages semantic awareness, transformer-based embeddings, and real-time anomaly detection to prevent prompt manipulation. This approach offers significant improvements over static filtering methods, providing dynamic adaptability across diverse deployment environments.

5.2. Practical Implications for System Developers and Engineers

For system architects and engineers, implementing context-aware sanitization requires rethinking prompt construction, input handling, and validation layers. Prompt integrity must be treated with the same level of scrutiny as secure code inputs. Incorporating sanitization modules into the design phase can minimize technical debt and prevent security patching in production. Development teams should also consider embedding prompt validation logic within continuous integration workflows to maintain consistency across updates. In latency-sensitive environments, careful trade-offs between performance and protection will determine the scope and depth of real-time sanitization. The integration of user transparency tools—such as visual indicators for flagged inputs—can further enhance end-user trust and system credibility.

5.3. Limitations of Current Approaches

Despite their effectiveness, context-aware sanitization systems face limitations in generalizability and computational efficiency. Their dependence on transformer-based models and semantic scoring introduces latency, which can challenge real-time interaction requirements. Furthermore, models trained on existing prompt corpora may struggle to detect novel adversarial patterns that evolve rapidly. Ambiguity in prompt interpretation, especially across languages or dialects, remains a source of false positives and missed detections. Additionally, human oversight is still necessary in high-stakes domains to interpret borderline cases and refine model behavior. The absence of standardized benchmarks for evaluating prompt defense tools further complicates comparative analysis and cross-system deployment.

5.4. Recommendations for Future Research and Industry Collaboration

To advance prompt security, future research should explore lightweight, language-agnostic sanitization models that maintain semantic accuracy with reduced computational overhead. Incorporating multi-modal inputs and contextual memory across prompt sequences may help mitigate sophisticated injection strategies. Industry-wide collaboration is essential to develop open-source benchmark suites, federated threat intelligence platforms, and shared prompt risk taxonomies. A unified framework for evaluating and certifying prompt defense mechanisms will enable clearer regulatory guidance and safer adoption across sectors. Greater investment in explainability and human-in-the-loop feedback mechanisms will also ensure that security measures align with ethical and operational expectations.

5.5. Final Thoughts on Building a Resilient LLM Ecosystem

As LLMs become foundational tools across digital ecosystems, securing their interactions against injection threats is no longer optional—it is a necessity. The rise of adversarial prompt engineering underscores the importance of semantic context awareness, proactive validation, and collaborative defense. By embracing context-aware token sanitization, developers and policymakers can reinforce the integrity of language-based systems, protect user trust, and enable scalable, responsible AI deployment. The road ahead requires not only technical innovation but also cross-disciplinary engagement to anticipate, mitigate, and adapt to the evolving threat landscape in natural language interfaces.

6. References

1. Adebayo MO, Olusanya AA. Prompt risk assessment models in digital identity systems. J Cybersecurity Eng. 2023;4(2):88-102.
2. Ajayi OO, Alozie CE, Abieba OA. Cybersecurity for business intelligence platforms. Afr J ICT Secur. 2025;9(1):14-33.
3. Ajiga D, Ayanponle L, Okatta CG. AI-powered HR analytics: transforming workforce optimization and decision-making. Int J Sci Res Arch. 2022;5(2):338-46.
4. Akerele JI, Uzoka A, Ojukwu PU, Olamijuwon OJ. Enhancing threat visibility in AI systems through linguistic fingerprinting. Inf Secur Rev. 2024;12(3):54-78.
5. Alonge EO, Dudu OF, Alao OB. Digital transformation in financial reporting: LLM adoption risks. Glob J Digit Account. 2024;7(1):112-26.
6. Ayanponle LO, Elufioye OA, Asuzu OF, Ndubuisi NL, Awonuga KF, Daraojimba RE. The future of work and human resources: a review of emerging trends and HR's evolving role. Int J Sci Res Arch. 2024;11(2):113-24.
7. Ayoola VB, Audu BA, Boms JC, Ifoga SM, Mbanugo OJ, Ugochukwu UN. Integrating industrial hygiene in hospice and home-based palliative care. IRE J. 2024;8(5):72-89.
8. Ayoola VB, Idoko PI, Danquah EO, Ukpoju EA, Obasa J, Otakwu A, *et al*. Optimizing construction management and workflow integration through autonomous robotics. Int J Sci Res Mod Technol. 2024;3(10). doi:10.38124/ijsrmt.v3i10.56
9. Azonuche TI, Enyejo JO. Evaluating the impact of agile scaling frameworks on productivity and quality in large-scale fintech software development. Int J Sci Res Mod Technol. 2024;3(6):57-69.
10. Azonuche TI, Enyejo JO. Exploring AI-powered sprint planning optimization. Int J Sci Res Mod Technol. 2024;3(8):40-57. doi:10.38124/ijsrmt.v3i8.448
11. Balogun TK, Enyejo JO, Ahmadu EO, Akpovino CU, Olola TM, Oloba BL. The psychological toll of nuclear proliferation and mass shootings. IRE J. 2024;8(4):114-29.

12. Balogun TK, Kalu OC, Ijiga AC, Olola TM, Ahmadu EO. Building advocacy coalitions and analyzing lobbyists' influence. Int J Sch Res Multidiscip Stud. 2024;5(1):88-102.
13. Bristol-Alagbariya B, Ayanponle LO, Ogedengbe DE. Sustainable business expansion: HR strategies for growth and stability. Int J Manag Entrep Res. 2024;6(12):3871-82.
14. Bristol-Alagbariya B, Ayanponle OL, Ogedengbe DE. Utilization of HR analytics for strategic cost optimization. Int J Sci Res Updates. 2023;6(2):62-9.
15. Ebenibo L, Enyejo JO, Addo G, Olola TM. Evaluating the sufficiency of the Data Protection Act 2023. Int J Sch Res Rev. 2024;5(1):88-107.
16. Eguagie MO, Idoko IP, Ijiga OM, Enyejo LA, Okafor FC, Onwusi CN. Geochemical and mineralogical characteristics of deep porphyry systems. Int J Sci Res Civ Eng. 2025;9(1). doi:10.32628/IJSRCE25911
17. Elufioye OA, Ndubuisi NL, Daraojimba RE, Awonuga KF, Ayanponle LO, Asuzu OF. Reviewing employee well-being and mental health initiatives in contemporary HR practices. Int J Sci Res Arch. 2024;11(1):828-40.
18. Enyejo JO, Babalola INO, Owolabi FRA, Adeyemi AF, Osam-Nunoo G, Ogwuche AO. Data-driven digital marketing and battery supply chain optimization in the battery-powered aircraft industry. Int J Sch Res Rev. 2024;5(2):1-20.
19. Ezeafulukwe C, Okatta CG, Ayanponle L. Frameworks for sustainable human resource management: integrating ethics, CSR, and data-driven insights. Magna Sci Adv Res Rev. 2022;6(1):78-85.
20. Fadairo MA, Eneh TO. Input sanitization vulnerabilities in neural networks. Afr J Secure AI Syst. 2024;3(2):66-82.
21. Gimba TL, Ezeanya CC. Natural language command chaining in multi-agent LLMs: an emerging threat. Cyber Risk Intell J. 2023;5(4):201-17.
22. Harry KD, Ezebuka CI, Umama EE. Ethical considerations in implementing generative AI for healthcare supply chains. Int J Biol Pharm Sci Arch. 2024;7(1):48-63.
23. Holmgren D, Petrov I. Performance benchmarking of semantic prompt filters. J AI Syst Eng. 2023;8(3):212-29.
24. Idoko DO, Mbachu OE, Ijiga AC, Okereke EK, Erondu OF, Nduka I. Assessing the influence of dietary patterns on preeclampsia and obesity among pregnant women in the United States. Int J Biol Pharm Sci Arch. 2024;8(1):85-103. doi:10.5281/zenodo.10808303
25. Idoko IP, Ijiga OM, Enyejo LA, Akoh O, Isenyo G. Integrating synthetic humans into IoT and AI ecosystems. Glob J Eng Technol Adv. 2024;19(1):6-36.
26. Ijiga AC, Balogun TK, Ahmadu EO, Klu E, Olola TM, Addo G. The role of the United States in shaping youth mental health advocacy and suicide prevention. Magna Sci Adv Res Rev. 2024;12(1):202-18. doi:10.5281/zenodo.12791503
27. Ijiga OM, Idoko IP, Ebiega GI, Olajide FI, Olatunde TI, Ukaegbu C. Harnessing adversarial machine learning for cybersecurity risk mitigation. Open Access Res J. 2024;13(1). doi:10.53022/oarjst.2024.11.1.0060
28. Iwe KA, Daramola GO, Okafor TS, Musa AA. Real-time geothermal risk monitoring using LLM-based diagnostics. Eng Sci Technol J. 2023;5(3):712-28.
29. Kalu OC, Ebikeme BO. Contextual sensitivity in AI compliance systems. J Emerg AI Policy. 2024;2(3):88-101.
30. Kisina D, Ochuba NA, Owoade S, Uzoka AC, Gbenle TP, Adanigbo OS. A conceptual framework for scalable microservices in real-time airline operations platforms. IRE J. 2022;6(8):344-9.
31. Kokogho E, Adeniji IE, Olorunfemi TA, Nwaozomudoh MO, Odio PE, Sobowale A. Framework for effective risk management strategies to mitigate financial fraud in Nigeria's currency operations. Int J Manag Organ Res. 2023;2(6):209-22.
32. Komi LS, Mustapha AY, Forkuo AY, Osamika D. Impact of digital health records in rural clinics: bridging infrastructure gaps. World J Adv Res Rev. 2023;22(2):410-29.
33. Leclerc JF, Iqbal MS, Musonda JK. Multi-turn discourse alignment for adversarial prompt prevention. Nat Lang Eng Adv. 2022;8(3):140-59.
34. Manuel HNN, Adeoye TO, Idoko IP, Akpa FA, Ijiga OM, Igbede MA. Optimizing passive solar design in Texas green buildings by integrating sustainable architectural features for maximum energy efficiency. Magna Sci Adv Res Rev. 2024;11(1):235-61. doi:10.30574/msarr.2024.11.1.0089
35. Mokogwu C, Achumie GO, Adeleke AG, Okeke IC, Ewim CP. A leadership and policy development model for driving operational success in tech companies. Int J Frontline Res Multidiscip Stud. 2024;4(1):1-14.
36. Ogata J, Okoye F, Audu BA. Prompt injection attacks in multilingual NLP systems. J Glob AI Threats. 2024;5(1):99-115.
37. Ogeawuchi JC, Akpe OEE, Adeyeye Y, Emeka AU. Securing cloud data warehouses using blockchain-based access rules. J Cybersecurity Cloud Comput. 2021;6(2):131-47.
38. Ogunsola AO, Olowu AO, Arinze CA, Izionworu VO. Strategic operations dashboard for predictive utility performance. Int J Appl Res Eng Technol. 2022;9(2):100-18.
39. Ogunwole O, Onukwulu EC, Joel MO, Adaga EM, Ibeh AI. Modernizing legacy systems: a scalable approach to next-generation data architectures and seamless integration. Int J Multidiscip Res Growth Eval. 2023;4(1):901-9.
40. Ogwuche AO, Ileanaju S, Akinyemi KA. LLM alignment challenges in multilingual chatbots. Int J NLP Risk Anal. 2023;8(4):45-63.
41. Ojadi JO, Onukwulu E, Owulade O. AI-powered computer vision for remote sensing and carbon emission detection in industrial and urban environments. Iconic Res Eng J. 2024;7(10):490-505.
42. Ojika FU, Owobu WO, Abieba OA, Esan OJ, Ubamadu BC, Daraojimba AI. Transforming cloud computing education: leveraging AI and data science for enhanced access and collaboration in academic environments. [Unpublished; journal details pending]. 2023.
43. Ojukwu PU, Cadet E, Osundare OS, Fakeyede OG, Ige AB, Uzoka A. The crucial role of education in fostering sustainability awareness and promoting cybersecurity measures. Int J Frontline Res Sci Technol. 2024;4(1):18-34.
44. Okeke RO, Ibokette AI, Ijiga OM, Enyejo LA, Ebiega GI, Olumubo OM. The reliability assessment of power

transformers. Eng Sci Technol J. 2024;5(4):1149-72.
45. Okonkwo PC, Idris TA. Prompt behavior monitoring using real-time inference logs. J AI Oper Secur. 2024;5(1):66-81.
46. Olamijuwon OJ, Uzoka A. Multistage injection detection for generative NLP pipelines. J Appl Mach Secur. 2024;6(4):310-28.
47. Olaseinde FT, Adesemoye OE. Leveraging graph-based token sanitization for injection prevention. Neural Interface Syst J. 2023;5(1):51-70.
48. Olawale OK, Ganiyu SM. Transforming prompt moderation in legal AI platforms. LegalTech J Afr. 2024;2(1):92-111.
49. Oloba BL, Olola TM, Ijiga AC. Powering reputation through employee communication in U.S. service industries. World J Adv Res Rev. 2024;23(3):2020-40.
50. Oloba BL, Olola TM, Ijiga AC. Powering reputation: employee communication as the key to boosting resilience and growth. World J Adv Res Rev. 2024;23(3):2020-40. doi:10.30574/wjarr.2024.23.3.2689
51. Olola TM, Ebiega GI. Dynamic pipeline validation in API-driven NLP agents. Glob J Intell Syst. 2024;10(1):80-94.
52. Oluwaseun A, Uzoka A. Evaluating real-time prompt tracking for adversarial prevention. Afr J Mach Learn Secur. 2023;6(2):144-62.
53. Lakshmikanth R. VC feedback design: Capturing user experience to tackle on-chip challenges. Int J Innov Res Manag Phys Sci Eng (IJIRMPS). 2025;13(3):1-7.
54. Omisore IO, Abubakar AI. Evaluating prompt coherency constraints in chat-based systems. Hum-Cent AI Res J. 2023;4(2):150-72.
55. Onuoha CJ, Edet AC. Reinforcement-based sanitization in federated LLM systems. AI Trust Saf Res Q. 2024;3(3):99-117.
56. Onwusi CN, Akoh O, Isibor NJ. Semantic boundary enforcement in conversational AI. LLM Eng J. 2024;9(2):155-72.
57. Otakwu A, Ayoola VB. Risk profiling for text-generation systems: a token-level approach. AI Secur Anal Rev. 2023;7(3):27-46.
58. Owolabi FRA, Babalola INO, Enyejo JO. Preventing prompt poisoning in supply chain forecasting bots. Int J Sch Res Rev. 2024;5(1):31-52.
59. Oyedokun O, Ewim SE, Oyeyemi OP. A comprehensive review of machine learning applications in AML transaction monitoring. Int J Eng Res Dev. 2024;20(11):173-83.
60. Oyeyipo I, Attipoe V, Mayienga BA, Onwuzulike OC, Ayodeji DC, Nwaozomudoh MO, *et al*. A conceptual framework for transforming corporate finance through strategic growth, profitability, and risk optimization. Int J Adv Multidiscip Res Stud. 2023;3(5):1527-38.
61. Sani MA, Akpe OEE. Token masking strategies for real-time prompt security. J Adapt AI Archit. 2024;4(1):35-53.
62. Tosin T, Ugbane SI. Threat modeling in transformer-based text processors. Cyber-AI Syst J. 2024;8(2):221-43.
63. Tula OA, Adekoya OO, Isong D, Daudu CD, Adefemi A, Okoli CE. Corporate advising strategies for aligning petroleum engineering with climate goals and CSR commitments. Corp Sustain Manag J. 2024;2(1):32-38.
64. Ugochukwu UN, Olorunfemi TA. NLP firewall design using prompt vector anomaly scoring. Intell Syst Secur Stud. 2024;7(3):115-34.
65. Urefe O, Odonkor TN, Obeng S, Biney E. Innovative strategic marketing practices to propel small business development and competitiveness. Magna Sci Adv Res Rev. 2024;11(2):278-96.
66. Uzoegwu MC, Awotiwon BO. Secure chaining mechanisms for multi-turn LLM APIs. Appl AI Gov J. 2024;9(2):87-105.
67. Uzoka A, Adeniji IE. Designing resilience into generative LLM interfaces. Afr J Softw Saf. 2024;5(3):144-65.
68. Wekpe TO, Owolabi OR. Constructing prompt logic boundaries in enterprise chatbots. Enterp AI Policy Rev. 2023;6(4):71-90.
69. Yusuf AO, Agbo DO. The role of AI system logging in prompt anomaly attribution. Digit Ethics AI J. 2024;8(1):94-111.
70. Zubairu AF, Addo G. Probabilistic language defense in AI text generators. J AI Secur Strateg. 2024;5(1):12-28.

How to Cite This Article
Oparah SO, Ezeh FE, Gado P, Adeleke AS, Gbaraba SV. Detecting and Neutralizing Prompt Injection Attacks in Web-Deployed Large Language Model APIs Using Context-Aware Token Sanitization. Int J Multidiscip Evol Res. 2025;6(2):92-100. Available from: https://doi.org/10.54660/IJMER.2025.6.2.92-100