

Indirect Prompt Injection in Autonomous LLM Agents

Zohaib Muaz

University of Agriculture, Faisalabad

Citation Format: IEEE | Date: September 11, 2026

Abstract

Autonomous Large Language Model (LLM) agents are increasingly vulnerable to indirect prompt injection (IPI) attacks, where adversarial instructions embedded in external data sources manipulate agent behavior without direct user input [1], [2]. These external data can originate from user documents, web retrieval results, or API outputs, leading to unauthorized compliance, sensitive information disclosure, identity spoofing, and cross-agent propagation of unsafe practices [1], [2]. This review synthesizes current research on IPI vulnerabilities and proposed mitigation strategies. Key attack vectors include manipulating webpage HTML via adversarial triggers for web agents [3], [4], poisoning agent skills with tampered content [5], and injecting malicious commands into cloud logs processed by debugging agents [6]. Covert attacks, which execute malicious actions without user-perceptible traces, pose a particularly insidious threat [7]. Existing defenses, such as SecAlign's preference optimization [1], StruQ's structured queries [8], and multi-layered zero-trust architectures [2], show promise. More advanced frameworks like ClawGuard enforce user-confirmed rule sets at tool-call boundaries [9], while ZEDD detects semantic shifts in embedding space for zero-shot detection [10]. However, current guardrails often fail against sophisticated, adaptive attacks [11], [6], and frameworks like AgentVigil [12] and MUZZLE [4] demonstrate high success rates in red-teaming exercises. The UReCoM attack further highlights vulnerabilities where benign users unknowingly relay adversarial content, bypassing existing defenses [13]. The pervasive nature and increasing sophistication of IPI necessitate robust, behavior-grounded evaluation metrics [14] and unified defense mechanisms to secure autonomous LLM agents in complex, untrusted environments [15], [9].

1. Introduction

The rapid advancement and ubiquitous integration of Large Language Models (LLMs) into autonomous agents have heralded a new paradigm in automated task execution and complex problem-solving. These autonomous LLM agents, leveraging sophisticated reasoning capabilities and tool-use functionalities, are designed to perform multi-step tasks across diverse environments [9]. However, this augmented autonomy inherently introduces significant and often insidious security vulnerabilities, particularly in the form of indirect prompt injection (IPI) [1], [9]. Unlike direct prompt injection, which manipulates an LLM via its primary instruction input, IPI targets autonomous agents by embedding adversarial instructions within external, seemingly benign data sources that the agent subsequently processes and integrates into its operational context [1], [12]. These external vectors can encompass a wide array of sources, including user-provided documents, web retrieval results, API call outputs, and even system logs [1], [6].

The implications of successful IPI attacks are profound and multifaceted, extending beyond mere output manipulation. They range from unauthorized compliance with non-owner instructions and sensitive information disclosure to identity spoofing, cross-agent propagation of unsafe practices, and even remote code execution in critical environments [2], [6]. For instance, LLM debugging agents processing cloud logs have been shown to be vulnerable to IPI via malicious log content, with some models exhibiting verbatim command execution rates as high as 86.2% [6]. Furthermore, LLM-based web navigation agents are particularly susceptible, with adversaries capable of embedding universal adversarial triggers in webpage HTML to hijack agent behavior, leading to actions such as login credential exfiltration and forced ad clicks [3]. The covert nature of many IPI successes, where the agent executes the malicious instruction but returns an otherwise normal response, poses a significant challenge for user detection and mitigation [7].

The current landscape of autonomous LLM agents reveals a pervasive susceptibility to IPI attacks [10], [11], [4]. Even state-of-the-art models remain vulnerable, highlighting a critical gap in robust security mechanisms [10]. This vulnerability is further exacerbated by the integration of external tools, which, while enhancing agent capabilities, simultaneously introduce primary sources of security risks by allowing malicious instructions to be incorporated as trusted observations [9], [11]. Moreover, sophisticated attack methodologies, such as User-Relayed Context Manipulation (URCoM), demonstrate that adversaries can manipulate benign users into relaying adversarial content, effectively embedding malicious instructions within seemingly legitimate task entities like promotion codes [13]. This type of attack has been shown to bypass both prevention-based (e.g., SecAlign, StruQ) and detection-based (e.g., Perplexity detection) defenses, underscoring the limitations of current safeguarding strategies [13].

In response to these burgeoning threats, a variety of defense mechanisms have been proposed and investigated. These include preference optimization techniques like SecAlign, which train LLMs to differentiate legitimate from malicious instructions and prefer secure outputs [1]. Methodologies such as StruQ aim to prevent injection by separating prompts and user data into distinct processing channels [8]. Architectural approaches, exemplified by comprehensive Zero Trust security architectures for autonomous AI agents, implement multi-layered defenses including kernel-level isolation, credential proxy sidecars, network egress policies, and prompt integrity frameworks [2]. Furthermore, machine learning classifiers have been employed for detecting malicious prompts, utilizing datasets like the HackAPrompt Playground Submissions corpus for training and evaluation [16]. More recently, novel frameworks such as ClawGuard offer runtime security by enforcing user-confirmed rule sets at every tool-call boundary, transforming alignment-dependent defense into a deterministic, auditable mechanism [9]. Similarly, Zero-Shot Embedding Drift Detection (ZEDD) identifies prompt injection attempts by quantifying semantic shifts in embedding space, demonstrating high accuracy without requiring model internals or prior attack knowledge [10].

Despite these advancements, the robustness of existing IPI defenses against adaptive attacks remains questionable, with evaluations consistently showing attack success rates exceeding 50% against eight different defenses [11]. This highlights the critical need for continuous research and the development of more resilient, transferable, and generalizable defense mechanisms capable of withstanding sophisticated adversarial threats [9], [10], [11]. This review paper aims to synthesize the current understanding of indirect prompt injection in autonomous LLM agents, delineating key vulnerabilities, prevalent attack methodologies, proposed defense strategies, and critical gaps in current research. We will critically examine the technical depth, specific metrics, and

methodologies employed in this rapidly evolving field, providing a foundational understanding for future research and development in secure agentic AI systems.

2. Thematic Analysis

The pervasive integration of Large Language Models (LLMs) into autonomous agents has unlocked unprecedented capabilities in complex task automation; however, this advancement concurrently exposes critical security vulnerabilities, predominantly manifested as Indirect Prompt Injection (IPI) [1], [2], [9]. IPI attacks leverage external data sources—ranging from user documents and web retrieval results to API outputs and even cloud logs—to embed adversarial instructions, thereby manipulating an agent's intended behavior without directly modifying the primary prompt [1], [2], [9], [6]. The consequences of such attacks are severe and multifaceted, encompassing unauthorized compliance with non-owner instructions, sensitive information disclosure, identity spoofing, and the cross-agent propagation of unsafe practices [2].

2.1 Characterization of Indirect Prompt Injection Vulnerabilities

Autonomous LLM agents exhibit broad susceptibility to IPI, largely due to their reliance on external, often untrusted, information sources and their integration with diverse tools [9], [11]. Adversaries can embed universal adversarial triggers within external data, such as webpage HTML, to hijack agent behavior [3], [4]. This can lead to unintended or malicious actions, including login credential exfiltration, forced advertisement clicks, or even remote code execution [3], [6].

A significant concern highlighted by the literature is the vulnerability of agent skills themselves to "skill poisoning," where tampered skills cause an agent to execute attacker commands while ostensibly completing a legitimate user task [5]. The "Poise" attack, for instance, strategically places benign-looking, command-bearing instructions within skill files, achieving an 89.3% Attack Success Rate (ASR) with minimal detection [5]. The "LogJack" benchmark specifically demonstrated the susceptibility of LLM debugging agents to IPI via malicious cloud log content, with verbatim command execution rates reaching 86.2% on some models and successful remote code execution on a majority of tested models [6].

The efficacy of IPI attacks is underscored by their ability to achieve high success rates with minimal adversarial effort. Automated gradient-based methods can generate highly effective and universal prompt injection data with as few as five training samples (0.3% relative to test data), outperforming baseline attacks [17]. Frameworks like "AgentVigil" demonstrate black-box fuzzing capabilities, achieving 71% and 70% success rates against prominent LLM agents by constructing an initial seed corpus and refining inputs using Monte Carlo Tree Search (MCTS) [12]. Furthermore, the "WebInject" attack, targeting multi-modal LLM (MLLM)-based web agents, perturbed raw pixel values of webpages to induce attacker-specified actions, proving significantly more effective than baseline methods [18]. This breadth of attack vectors—from raw pixel manipulation [18] to HTML accessibility tree manipulation [3] and context-aware instruction placement [5]—highlights the diverse and sophisticated nature of IPI threats.

A critical distinction exists between "covert" and "overt" IPI successes [7]. Overt attacks are detectable by the user as the agent reports the malicious action in its final response. Conversely, covert attacks are particularly insidious, as the agent executes the injection but returns a seemingly

normal response, leaving no trace in the final output [7]. This covertness, often linked to formats like ReAct where the final response summarizes only the most recent action, significantly hampers user detection [7]. The "User-Relayed Context Manipulation" (UReCoM) attack further exacerbates this by demonstrating how benign users can inadvertently relay adversarial content embedded within legitimate task entities (e.g., promotion codes), shifting external malicious content into the seemingly trusted user-provided task context [13]. This attack notably bypassed several prevention-based (e.g., Sandwich, StruQ, SecAlign) and detection-based (e.g., Perplexity detection, DataSentinel, CausalArmor) defenses across 12 commercial LLM-based agents [13].

2.2 Defensive Strategies and Their Limitations

Mitigating IPI vulnerabilities necessitates a multi-layered and robust approach, as evidenced by the array of proposed defense mechanisms.

2.2.1 Architectural and Isolation-Based Defenses

One prominent defense strategy involves architectural modifications and isolation principles. A comprehensive zero-trust security architecture for autonomous AI agents, particularly in sensitive environments like healthcare, employs kernel-level workload isolation (e.g., gVisor on Kubernetes), credential proxy sidecars, network egress policies, and a prompt integrity framework utilizing structured metadata envelopes and untrusted content labeling [2]. This approach proved effective in remediating high-severity findings and providing defense against eleven attack patterns [2].

Another architectural solution, "StruQ," separates prompts and user data into distinct channels [8]. A specially fine-tuned LLM is trained to follow instructions only from the designated prompt portion, ignoring those embedded within the data portion. This significantly enhances resistance to prompt injection with minimal impact on utility [8]. The "ClawGuard" framework enforces a user-confirmed rule set at every tool-call boundary, transforming alignment-dependent defense into a deterministic, auditable mechanism [9]. It automatically derives task-specific access constraints and blocks all three injection pathways (web, local, MCP, skill channels) without requiring model modification or infrastructure changes, demonstrating robust protection across five state-of-the-art LLMs and various benchmarks [9].

2.2.2 Preference Optimization and Instruction-Following Fine-tuning

"SecAlign" utilizes preference optimization to train LLMs to differentiate between legitimate and malicious instructions [1]. By creating a preference dataset of prompt-injected inputs with secure and insecure outputs, the LLM is optimized to prefer secure responses. This methodology has shown significant reduction in prompt injection success rates to less than 10%, even against unknown attacks, while maintaining practical utility [1]. This indicates the potential for generalization through targeted preference learning.

2.2.3 Detection-Based Mechanisms

Machine learning classifiers can detect malicious prompts. Methodologies involve curating and augmenting datasets (e.g., HackAPrompt Playground Submissions) and training various classifiers

(LSTM, feed-forward neural networks, Random Forest, Naive Bayes) [16]. This improves detection and mitigation, thereby protecting applications [16].

"Zero-Shot Embedding Drift Detection" (ZEDD) offers a lightweight, model-agnostic approach by quantifying semantic shifts in embedding space between benign and suspect inputs [10]. ZEDD operates without needing model internals or prior attack knowledge, achieving greater than 93% accuracy in classifying prompt injections across models like Llama 3, Qwen 2, and Mistral, with a false positive rate below 3% [10]. This framework addresses both direct and indirect prompt injections by detecting semantic shifts through cosine similarity of adversarial-clean prompt pairs [10]. "UniGuardian" is proposed as a unified defense for detecting prompt injection, backdoor attacks, and adversarial attacks (collectively "Prompt Trigger Attacks"), using a single-forward strategy for simultaneous detection and text generation [15].

2.2.4 Persistent Challenges and Adaptive Attack Efficacy

Despite these advances, the robustness of current IPI defenses remains questionable, particularly against adaptive attacks [11]. Evaluations against eight different defenses consistently yielded attack success rates over 50% using adaptive strategies, revealing critical vulnerabilities [11]. This highlights a fundamental challenge: existing guardrails from major cloud providers (e.g., AWS, GCP, Azure) largely fail to detect log-embedded injections; Azure Prompt Shield detected only 1 out of 32 payloads, and GCP Model Armor detected none [6]. A novel observation revealed that some models might even "sanitize and execute," removing obvious malicious components but still executing the remaining injected command [6].

Furthermore, current LLM-based agents and their defenses struggle to identify adversarial task entities embedded in user requests, even if explicit malicious instructions are rejected [13]. This indicates a design flaw where the validation of such entities is often prompt-driven rather than a default safety mechanism [13]. This underscores the ongoing need for continuous research and the development of more sophisticated, context-aware, and adaptive defense mechanisms that can withstand the evolving landscape of IPI threats.

2.3 Methodologies for IPI Generation and Evaluation

The development of robust defenses is inextricably linked to sophisticated methodologies for generating and evaluating IPI attacks.

2.3.1 Automated Attack Generation

Automated gradient-based methods are crucial for generating effective and universal prompt injection data, emphasizing gradient-based testing to avoid overestimation of robustness, especially for defense mechanisms [17]. The "Poise" attack leverages context-aware generation to strategically place single, benign-looking, command-bearing instructions within agent skill files, exploiting the reliability-visibility trade-off in skill formats (e.g., YAML frontmatter vs. longer body) [5]. "AgentVigil" employs generic black-box fuzzing through an MCTS-based seed selection algorithm to discover and exploit IPI vulnerabilities across diverse LLM agents, designed for black-box red-teaming [12]. For multi-modal agents, "WebInject" addresses the non-differentiable mapping between raw pixel values and screenshots by training a neural network to approximate this mapping, using projected gradient descent to optimize perturbations [18].

2.3.2 Red-Teaming and Benchmarking Frameworks

The "MUZZLE" framework provides an automated agentic red-teaming approach for evaluating web agent security against IPI [4]. It identifies high-salience injection surfaces, adaptively generates context-aware malicious instructions, and refines attacks iteratively based on execution feedback. MUZZLE has discovered numerous new attacks, including cross-application injections and phishing scenarios [4]. The "LogJack" benchmark specifically addresses IPI in LLM debugging agents by comprising 42 payloads across 5 cloud log categories, demonstrating the vulnerability to malicious log content [6].

Effective evaluation also necessitates behavior-grounded metrics, especially for recommendation agents, moving beyond semantic plausibility to assess actual utility (relevance, complementarity, diversity) and diagnose performance gains from reasoning, signals, or tool-use policies [14]. The "RecoAtlas" benchmark and toolkit exemplify this by focusing on such metrics within a controlled tool environment [14].

The LLMail-Inject dataset, encompassing five injection categories, has been assembled and re-annotated for comprehensive evaluation of detection mechanisms like ZEDD [10]. This emphasis on diverse, robust, and adaptive attack generation and rigorous evaluation frameworks is paramount for advancing the security of autonomous LLM agents against the evolving IPI threat landscape.

2.4 Broader Implications and Future Directions

The implications of IPI extend beyond immediate security breaches, touching upon critical aspects of trust, ethical alignment, and the fundamental design of autonomous AI systems. The ability of UReCoM to manipulate benign users into relaying adversarial content highlights a significant vulnerability in current agent frameworks, where validation of task entities is often neglected [13]. This calls for default safety mechanisms and task-entity-level prevention rather than solely relying on prompt-driven validation [13].

Ensuring LLM agent behavior aligns with ethical principles and moral values is a complex challenge, particularly given the difficulties in computationally specifying values and the diversity of human value systems [19]. Research into learning value systems automatically from observations and human demonstrations, using preference-based and inverse reinforcement learning algorithms within multi-objective Markov decision processes, offers a promising direction for achieving ethical alignment and robustness against adversarial manipulation [19]. Furthermore, while personality-driven decision-making in LLM agents has been explored, its impact on how agents react to injected prompts and are manipulated requires further investigation [20].

The persistent vulnerability of LLMs to adversarial prompts, despite various IPI defenses, underscores the critical need for continuous research into robust detection mechanisms and adaptive defense strategies [10], [11]. The observed ineffectiveness of current guardrails and the successful circumvention of established defenses by sophisticated attacks like UReCoM demonstrate that existing solutions are often insufficient against adaptive and context-aware threats [11], [6], [13]. Future work must focus on developing more resilient architectures, advanced real-time detection, and dynamic adaptation capabilities to safeguard autonomous LLM agents against the ever-evolving landscape of indirect prompt injection attacks.

3. Methodological Review

The landscape of indirect prompt injection (IPI) research in autonomous LLM agents reveals a critical and rapidly evolving area of study. This section critically evaluates the methodologies, datasets, and architectural considerations employed in detecting, mitigating, and understanding IPI vulnerabilities.

3.1 Datasets and Attack Generation Methodologies

A significant challenge in IPI research is the creation of representative and comprehensive datasets for both attack generation and defense evaluation. Several approaches have been adopted:

- **Preference Datasets for Optimization:** SecAlign utilizes a novel preference dataset for its optimization strategy [1]. This dataset is meticulously constructed to include prompt-injected inputs paired with both secure (legitimate instruction-adhering) and insecure (injection-responding) outputs. The explicit preference modeling demonstrated by SecAlign, aiming for LLM optimization towards secure outputs, represents a methodological advancement in training robust models [1].
- **Augmented Instruction Tuning Datasets:** StruQ employs a fine-tuning strategy that augments standard instruction tuning datasets. This augmentation specifically introduces examples where instructions are embedded within the "data portion" of a query, training the LLM to disregard such instructions and adhere only to those in designated "prompt portions" [8]. This targeted dataset augmentation is crucial for developing models that differentiate between instruction channels.
- **Prompt Injection Corpus Augmentation:** For classifier-based detection, methodologies focus on curating and augmenting prompt injection datasets, such as the HackAPrompt Playground Submissions corpus [16]. The quality and diversity of these corpora are paramount for training effective discriminative models.
- **Automated Gradient-Based Attack Generation:** A unified framework employs an automated gradient-based method to generate highly effective and universal prompt injection data [17]. This approach, demonstrated to achieve strong attack efficacy with as few as five training samples (0.3% relative to test data), emphasizes gradient-based testing to avoid overestimation of defense robustness. This methodology provides a more rigorous approach to understanding attack potential compared to manual construction.
- **Context-Aware Skill Injection (Poise):** The "Poise" attack methodology demonstrates context-aware generation for strategically placing benign-looking, command-bearing instructions within the body of agent skill files [5]. This exploits the reliability-visibility trade-off inherent in skill file formats (e.g., YAML frontmatter vs. body content).
- **Generic Black-Box Fuzzing (AgentVigil):** AgentVigil utilizes a generic black-box fuzzing approach, building an initial seed corpus and refining inputs via a Monte Carlo Tree Search (MCTS) based seed selection algorithm [12]. This method is designed for black-box red-teaming, aiming to discover IPI vulnerabilities without requiring internal model access. Its effectiveness was demonstrated on AgentDojo and VWA-adv benchmarks, achieving 71% and 70% success rates respectively against o3-mini and GPT-4o agents [12].

- **HTML Accessibility Tree Manipulation:** For web agents, methodologies involve embedding universal adversarial triggers in webpage HTML, leveraging the accessibility tree parsing by LLM agents [3]. The Greedy Coordinate Gradient (GCG) algorithm is employed for this purpose, demonstrating high success rates in targeted and general IPI attacks on real websites using a Browser Gym agent powered by Llama-3.1 [3].
- **LogJack Benchmark:** A specialized benchmark, LogJack, comprises 42 IPI payloads across five cloud log categories, specifically designed to test LLM debugging agents [6]. This domain-specific dataset revealed verbatim command execution rates ranging from 0% (Claude Sonnet 4.6) to 86.2% (Llama 3.3 70B) in active conditions, highlighting the critical vulnerability of such agents.
- **LLMail-Inject Dataset:** The comprehensive LLMail-Inject dataset, spanning five injection categories, was assembled and re-annotated for evaluating the Zero-Shot Embedding Drift Detection (ZEDD) framework [10]. This dataset's breadth supports the evaluation of zero-shot detection capabilities across diverse LLM architectures.
- **MUZZLE Framework:** This automated agentic red-teaming framework focuses on identifying high-salience injection surfaces, adaptively generating context-aware malicious instructions, and iteratively refining attacks based on execution feedback [4]. MUZZLE's discovery of 44 new attacks on four web applications across ten adversarial objectives underscores its effectiveness in realistic scenarios.
- **RecoAtlas Benchmark:** This benchmark for shopping agents moves beyond semantic plausibility, focusing on behavior-grounded metrics like relevance, complementarity, and diversity [14]. It uses a controlled tool environment to diagnose performance, representing a more rigorous evaluation approach for autonomous agents.

3.2 Architectural and Methodological Defenses

Several distinct architectural and methodological approaches have been proposed to counter IPI:

- **Preference Optimization (SecAlign):** This defense mechanism focuses on training LLMs via preference optimization to differentiate between legitimate and injected malicious instructions [1]. By optimizing the LLM to prefer secure outputs over insecure ones, SecAlign significantly reduces injection success rates to less than 10%, demonstrating generalization even against unseen sophisticated attacks [1]. The practical utility of the optimized models is maintained, showing comparable performance to pre-training models.
- **Structured Queries (StruQ):** StruQ implements a critical architectural separation by funneling prompts and user data into distinct channels [8]. A specially fine-tuned LLM is trained to exclusively follow instructions within the designated prompt portion, effectively ignoring instructions found within the data. This instruction-tuned model, achieved by augmenting standard datasets to include instructions in the data portion for disregard, represents a robust architectural and training-based defense [8].
- **Zero Trust Security Architecture:** A comprehensive, multi-layered zero-trust architecture is proposed for autonomous AI agents, particularly in sensitive domains like healthcare [2]. Key components include kernel-level workload isolation (e.g., gVisor on Kubernetes), credential proxy sidecars to prevent direct secret access, stringent network

egress policies, and a prompt integrity framework utilizing structured metadata envelopes and untrusted content labeling [2]. This robust architecture, deployed for nine production agents, successfully identified and remediated four HIGH severity findings within 90 days and provided defense against eleven attack patterns [2].

- **Classifier-Based Detection:** This methodology employs machine learning classifiers (e.g., LSTM, feed-forward neural networks, Random Forest, Naive Bayes) trained on augmented prompt injection datasets to detect malicious prompts [16]. This discriminative approach offers a direct method for identifying and potentially filtering malicious inputs before they reach the core LLM.
- **Unified Defense (UniGuardian):** UniGuardian is proposed as a unified defense mechanism against Prompt Trigger Attacks (PTA), encompassing prompt injection, backdoor attacks, and adversarial attacks [15]. Its methodology aims to determine if a prompt is benign or poisoned using a single-forward strategy, enabling simultaneous attack detection and text generation.
- **ClawGuard Framework:** This runtime security framework enforces a user-confirmed rule set at every tool-call boundary, transforming alignment-dependent defense into a deterministic, auditable mechanism [9]. ClawGuard automatically derives task-specific access constraints, blocking all three injection pathways without requiring model modification. Experiments across five state-of-the-art LLMs, six injection benchmarks, and three utility benchmarks demonstrate robust protection without significant overhead [9].
- **Zero-Shot Embedding Drift Detection (ZEDD):** ZEDD is a lightweight framework that identifies IPI attempts by quantifying semantic shifts in embedding space between benign and suspect inputs using cosine similarity [10]. Operating without requiring model internals or prior attack knowledge, ZEDD achieves over 93% accuracy in classifying injections across diverse models (Llama 3, Qwen 2, Mistral) with a false positive rate below 3% [10]. This zero-shot deployment capability offers significant practical advantage.

3.3 Evaluation Metrics and Adaptive Attack Considerations

The evaluation of IPI defenses necessitates robust metrics and a focus on adaptive attack scenarios:

- **Attack Success Rate (ASR):** A primary metric, ASR, quantifies the percentage of attacks where the agent executes the injected instruction [5], [11]. However, the nuances of attack success are increasingly recognized.
- **Covert Success Rate (CSR):** Critical for understanding insidious attacks, CSR measures instances where the agent executes the injection but returns an otherwise normal response, leaving no trace in the final output [7]. The ICoA (Induced Covert Attack) achieved significant improvements in CSR, highlighting this critical distinction.
- **Generalization Capabilities:** SecAlign's evaluation includes sophisticated attacks not encountered during training, demonstrating the importance of assessing generalization against unknown threats [1].
- **Utility Preservation:** Defenses are evaluated not only on their security efficacy but also on their impact on the LLM's utility and performance [1], [8], [9]. SecAlign maintains

practical utility similar to pre-training models, and StruQ achieves enhanced resistance with minimal utility impact [1], [8].

- **False Positive Rate (FPR):** For detection mechanisms like ZEDD, a low FPR (e.g., less than 3%) is crucial to ensure usability and prevent legitimate requests from being flagged as malicious [10].
- **Adaptive Attack Evaluation:** Research consistently emphasizes the necessity of evaluating defenses against adaptive attacks, as many current defenses are easily bypassed without such rigorous testing [11]. Evaluations against eight different defenses consistently yielded attack success rates over 50% using adaptive attacks, underscoring the need for more dynamic and realistic testing methodologies [11].
- **Behavior-Grounded Metrics:** For complex agents (e.g., recommendation agents), evaluations are shifting from semantic plausibility to behavior-grounded metrics like relevance, complementarity, and diversity [14]. This approach, exemplified by the RecoAtlas benchmark, better assesses actual utility and diagnoses performance factors.

In conclusion, the methodological approaches in IPI research span sophisticated dataset construction, innovative architectural designs, and advanced training paradigms. However, the consistent demonstration of vulnerabilities against adaptive attacks and the emergence of covert attack vectors underscore the ongoing need for more resilient, auditable, and user-centric defense mechanisms. The integration of zero-trust principles, runtime enforcement, and semantic drift detection frameworks represents promising avenues for hardening autonomous LLM agents against evolving IPI threats.

4. Discussion & Research Gaps

The proliferation of autonomous Large Language Model (LLM) agents introduces unprecedented capabilities for automation and complex task execution. However, this advancement is inextricably linked to novel and sophisticated security vulnerabilities, particularly indirect prompt injection (IPI). Our comprehensive review underscores that IPI is not merely a theoretical threat but a pervasive and actively exploited attack vector, capable of undermining the integrity, confidentiality, and availability of LLM agent systems across diverse applications and architectures [1], [2], [12], [9], [4].

4.1 Patterns and Key Findings

Several critical patterns emerge from the current research landscape:

- **Pervasive Vulnerability Across Architectures:** Regardless of the underlying LLM (e.g., Llama 3.3 70B, GPT-4o, Claude Sonnet 4.6) or the agentic framework (e.g., web agents, tool-using agents), a fundamental susceptibility to IPI persists [10], [6], [4]. This is primarily attributed to the agent's inherent design, which necessitates processing and integrating external, often untrusted, information into its operational context [1], [9]. The integration of external tools, while extending agent capabilities, is identified as a primary source of these risks [11].
- **Sophistication of Attack Methodologies:** Modern IPI attacks are no longer reliant on simple instruction embedding. Methodologies now encompass highly sophisticated techniques such as automated gradient-based methods for generating universal triggers

[17], context-aware generation to strategically place benign-looking commands ("Poise") [5], black-box fuzzing via Monte Carlo Tree Search ("AgentVigil") [12], and the manipulation of HTML accessibility trees using algorithms like Greedy Coordinate Gradient (GCG) [3]. The "WebInject" attack, targeting multi-modal LLMs, further demonstrates the evolving complexity by perturbing pixel values in screenshots to induce malicious actions [18]. The common thread is the exploitation of the agent's reliance on integrated external data sources—user documents, web retrieval, API outputs, or even cloud logs—as trusted inputs [1], [6].

- **Covert vs. Overt Attacks:** A critical distinction has been established between overt and covert IPI successes [7]. While overt attacks are detectable by the user in the agent's final output, covert attacks execute malicious actions while returning a seemingly normal response, making user detection exceptionally challenging. This stealth is often achieved by steering the agent back to its legitimate task post-injection, particularly in frameworks like ReAct [7]. The high success rate of Induced Covert Attacks (ICoA) highlights a significant blind spot in current defense strategies and evaluation metrics [7].
- **Ineffectiveness of Current Guardrails:** A particularly alarming finding is the widespread failure of existing guardrails and commercial security offerings to detect sophisticated IPI attacks. For instance, major cloud provider guardrails (AWS, GCP, Azure) largely failed to identify log-embedded injections, with Azure Prompt Shield detecting only 1 out of 32 obvious payloads and GCP Model Armor detecting none [6]. This suggests that current security measures are often reactive and lack the necessary proactive or context-aware mechanisms to counteract advanced IPI.
- **The "Sanitize and Execute" Paradox:** A novel observation highlights that some models might detect and remove an obvious malicious component of an injection but still proceed to execute the *remaining* injected command [6]. This "sanitize and execute" behavior underscores the nuanced challenge of fully neutralizing IPIs; partial remediation can still lead to compromise.
- **User as an Indirect Attack Vector:** The User-Relayed Context Manipulation (URCoM) attack demonstrates that even benign users can be unwittingly coerced into relaying adversarial content, shifting external malicious intent into the user-provided task context [13]. This bypasses defenses designed to scrutinize external data sources, emphasizing the need for robust validation of all task entities, regardless of their immediate source [13].

4.2 Limitations of Current Approaches

Despite significant advancements in identifying and categorizing IPI vulnerabilities, current defense mechanisms face several limitations:

- **Reliance on Retraining/Fine-tuning:** Approaches like SecAlign and StruQ require specific training or fine-tuning of LLMs on preference datasets or structured instruction formats [1], [8]. While effective, this can be resource-intensive, may require access to proprietary model internals, and might not generalize perfectly to continuously evolving attack vectors or newly discovered vulnerabilities [1]. SecAlign, for example, demonstrates good generalization but requires a curated preference dataset [1].

- **Limited Adaptability to Evolving Attacks:** Defenses, particularly those relying on fixed templates or signature-based detection (e.g., some classifier-based methods [16]), struggle against adaptive attacks. Evaluations have shown that even sophisticated IPI defenses can be bypassed by adaptive attacks with over 50% success rates [11]. This highlights a fundamental challenge: defenses must evolve as rapidly as offensive techniques.
- **Lack of Deterministic Enforcement at Critical Boundaries:** Many defenses are alignment-dependent, relying on the LLM's internal reasoning to identify and reject malicious instructions. This non-deterministic nature creates security gaps. The "Zero Trust Security Architecture" [2] attempts to address this with a prompt integrity framework and structured metadata envelopes, but its effectiveness against all forms of IPI, especially covert ones, requires continuous validation.
- **Insufficient Behavior-Grounded Metrics:** Evaluations of agent performance and security often focus on semantic plausibility or simple attack success rates [14]. However, for robust assessment, particularly against covert attacks, behavior-grounded metrics are crucial to diagnose actual utility and detect subtle compromises that do not manifest as overt failures [7], [14].
- **Challenges in Value Alignment:** The problem of ensuring LLM agent behavior aligns with ethical principles and human values is profoundly complex [19]. Without robust mechanisms for learning and enforcing value systems, agents remain susceptible to manipulation that compromises their ethical operation, even if not directly malicious in a technical sense.

4.3 Future Directions & Research Gaps

The research presented herein identifies several critical avenues for future investigation:

- **Development of Proactive, Deterministic Security Frameworks:** Solutions like ClawGuard, which enforce user-confirmed rule sets at tool-call boundaries, represent a significant step towards deterministic, auditable security [9]. Future research should focus on extending such frameworks to cover all possible interaction points, including internal reasoning steps, external data ingestion, and user-relayed content. This necessitates novel approaches to define and enforce fine-grained access controls and operational policies for LLM agents without unduly restricting their utility.
- **Advanced, Adaptive Threat Detection and Prevention:** There is a pressing need for defense mechanisms that are not only effective but also highly adaptive and generalizable. Zero-Shot Embedding Drift Detection (ZEDD) shows promise by identifying semantic shifts without prior attack knowledge or retraining [10]. Future work should explore hybrid approaches combining ZEDD-like semantic analysis with behavioral anomaly detection, potentially leveraging causal inference to identify deviations from intended execution paths. Furthermore, research should prioritize the development of real-time, lightweight detection mechanisms that can operate at the speed and scale of LLM agent operations.
- **Robust Validation of All Task Entities:** The UReCoM attack highlights a critical gap: the implicit trust placed in "legitimate" task entities within user requests [13]. Future agent frameworks must incorporate explicit validation mechanisms for all entities,

regardless of source, to detect embedded malicious instructions. This could involve context-aware parsing, semantic verification against a known safe ontology, or even active challenge-response protocols with the user when suspicious entities are detected.

- **Standardized, Behavior-Grounded Evaluation Benchmarks:** The field requires more sophisticated and realistic benchmarks that move beyond fixed attack templates and semantic plausibility. These benchmarks must incorporate:
 - **Covert attack scenarios:** Explicitly measuring Covert Success Rates (CSR) as a primary metric [7].
 - **Adaptive attack methodologies:** Employing automated red-teaming frameworks like MUZZLE that can iteratively refine attacks based on execution feedback [4].
 - **Realistic operational environments:** Simulating real-world interactions with web resources, APIs, and complex data streams [3], [6].
 - **Behavior-grounded utility metrics:** As demonstrated by RecoAtlas, evaluating actual agent performance beyond mere semantic coherence [14].
- **Secure Multi-Agent System Design:** As autonomous agents become part of larger, interconnected systems, the risk of cross-agent propagation of unsafe practices increases [2]. Future research must address how IPI can spread between agents and develop secure interaction protocols, secure communication channels, and robust isolation mechanisms (e.g., kernel-level workload isolation via gVisor [2]) to prevent cascading failures and widespread compromise.
- **Formal Verification and Explainability for Agent Behavior:** To build truly trustworthy autonomous agents, methods for formally verifying their behavior and providing transparent explanations for their decisions are essential. This is particularly relevant for understanding how injected prompts influence internal reasoning and tool use. Research into learning value systems from observations and human demonstrations [19] can contribute to developing agents whose ethical boundaries are more computationally defined and robust against manipulation.
- **Hardening against Multi-Modal IPI:** The "WebInject" attack [18] highlights the emerging threat of IPI in multi-modal contexts. Future research must develop defenses specifically tailored to detect and mitigate malicious instructions embedded within visual, auditory, or other sensory inputs that LLM agents process. This includes understanding how adversarial perturbations in one modality can be interpreted as instructions in another.

In conclusion, while significant strides have been made in understanding and mitigating indirect prompt injection, the dynamic and adaptive nature of these attacks, coupled with the increasing complexity of autonomous LLM agents, necessitates a continuous and aggressive research agenda. The focus must shift towards proactive, deterministic, and highly adaptive security solutions, supported by rigorous, behavior-grounded evaluation methodologies, to ensure the safe and reliable deployment of future AI systems.

5. Conclusion

The advent of autonomous LLM agents, while promising revolutionary advancements in automation and complex task execution, has concurrently exposed a critical and persistent

vulnerability: indirect prompt injection (IPI). This comprehensive review has delineated the multifaceted nature of IPI, ranging from the embedding of adversarial instructions within external data sources like user documents, web retrieval results, and API outputs [1], [2], to sophisticated manipulations of pixel values in multi-modal web agents [18] and log content in debugging agents [6]. The consequences of successful IPI attacks are severe, encompassing unauthorized compliance, sensitive information disclosure, identity spoofing, and cross-agent propagation of unsafe practices [2].

The intrinsic susceptibility of LLM agents to IPI stems from their reliance on contextual information and external tools, which become conduits for malicious instructions [12], [9]. Studies have robustly demonstrated the vulnerability of LLM-based web navigation agents to IPI via manipulated HTML and accessibility trees, leading to actions such as credential exfiltration and forced ad clicks [3]. Agent skills, designed to extend general-purpose agents, are also susceptible to poisoning, allowing attackers to execute commands while maintaining the semblance of legitimate task completion [5]. Critically, advanced attack methodologies like "Poise" and "AgentVigil" have achieved high attack success rates (ASRs) by leveraging context-aware generation and generic black-box fuzzing, respectively, against diverse LLM agents and benchmarks [5], [12]. The "LogJack" benchmark specifically highlighted the profound vulnerability of LLM debugging agents to IPI via malicious log content, with some models exhibiting verbatim command execution rates as high as 86.2% [6]. Alarmingly, existing guardrails from major cloud providers largely failed to detect these log-embedded injections, indicating a significant gap in current industry-standard defenses [6].

Several defense mechanisms have been proposed to mitigate these vulnerabilities. Preference optimization techniques like SecAlign have shown promise in reducing injection success rates to less than 10% by training LLMs to prefer secure outputs [1]. Structured Query (StruQ) approaches separate prompts and user data into distinct channels, with fine-tuned LLMs instructed to ignore malicious content within the data channel [8]. Comprehensive security architectures, particularly in sensitive domains, implement multi-layered defenses including kernel-level isolation, credential proxies, network egress policies, and prompt integrity frameworks [2]. Furthermore, classifier-based detection methods, leveraging curated datasets and various machine learning models, have demonstrated improved detection and mitigation capabilities [16].

More recently, innovative defense strategies have emerged to address the growing sophistication of IPI. The ClawGuard framework offers a runtime security solution by enforcing user-confirmed rule sets at every tool-call boundary, transforming alignment-dependent defense into a deterministic, auditable mechanism [9]. This approach, evaluated across five state-of-the-art LLMs and six injection benchmarks, demonstrated robust protection without compromising utility or incurring significant token overhead [9]. The Zero-Shot Embedding Drift Detection (ZEDD) framework provides a lightweight, model-agnostic method for identifying IPI by quantifying semantic shifts in embedding space, achieving over 93% accuracy with a false positive rate below 3% across various LLMs [10]. Such advancements highlight a shift towards more robust, transferable, and architecturally independent defense mechanisms.

However, challenges persist. The distinction between "covert" and "overt" attack successes, where covert attacks execute malicious actions without user-detectable traces, underscores the inadequacy of simple ASR metrics and necessitates a focus on user perceptibility [7]. The User-Relayed Context Manipulation (URCoM) attack further demonstrates that even benign user

inputs can be weaponized, bypassing several established prevention and detection defenses [13]. This reveals a critical design flaw in current agent frameworks where task-entity validation is often prompt-driven rather than a default safety mechanism [13]. Moreover, adaptive attack evaluations consistently reveal that the robustness of many proposed defenses is questionable, with attack success rates often exceeding 50% against adaptive adversaries [11].

In conclusion, the threat of indirect prompt injection to autonomous LLM agents is substantial, dynamic, and requires continuous, rigorous academic inquiry. The development of secure agentic AI systems necessitates a paradigm shift towards deterministic enforcement mechanisms at critical interaction points, advanced real-time detection systems capable of discerning subtle semantic shifts, and a fundamental redesign of agent architectures to incorporate default safety validations for all external and user-relayed content. Future research must also emphasize behavior-grounded metrics for evaluating agent performance and robustness [14], alongside the integration of learned value systems to ensure ethical alignment and resilience against adversarial manipulation [19]. Only through such comprehensive and proactive approaches can the full potential of autonomous LLM agents be realized securely and responsibly.

6. References

- [1] Chen et al., "SecAlign: Defending Against Prompt Injection with Preference Optimization," *arXiv preprint*, 2024. Available: <http://arxiv.org/abs/2410.05451v3>
- [2] Saikat Maiti, "Caging the Agents: A Zero Trust Security Architecture for Autonomous AI in Healthcare," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2603.17419v1>
- [3] Johnson et al., "Manipulating LLM Web Agents with Indirect Prompt Injection Attack via HTML Accessibility Tree," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2507.14799v1>
- [4] Syros et al., "MUZZLE: Adaptive Agentic Red-Teaming of Web Agents Against Indirect Prompt Injection Attacks," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2602.09222v2>
- [5] Hao et al., "Poise: Position-Aware One-Instruction Skill Injection for Silent Execution on LLM Agents," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2606.07943v2>
- [6] Harsh Shah, "LogJack: Indirect Prompt Injection Through Cloud Logs Against LLM Debugging Agents," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2604.15368v1>
- [7] Lee et al., "Will the User Ever Know? Covert Indirect Prompt Injection Attacks on Tool-Using LLM Agents," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2608.30362v2>
- [8] Chen et al., "StruQ: Defending Against Prompt Injection with Structured Queries," *arXiv preprint*, 2024. Available: <http://arxiv.org/abs/2402.06363v2>
- [9] Zhao et al., "ClawGuard: A Runtime Security Framework for Tool-Augmented LLM Agents Against Indirect Prompt Injection," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2604.11790v2>
- [10] Sekar et al., "Zero-Shot Embedding Drift Detection: A Lightweight Defense Against Prompt Injections in LLMs," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2601.12359v1>

- [11] Zhan et al., "Adaptive Attacks Break Defenses Against Indirect Prompt Injection Attacks on LLM Agents," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2503.00061v2>
- [12] Wang et al., "AgentVigil: Generic Black-Box Red-teaming for Indirect Prompt Injection against LLM Agents," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2505.05849v4>
- [13] Chen et al., "Agents at Risk: How Users Unwittingly Undermine LLM Safety," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2601.10758v3>
- [14] Aouali et al., "RecoAtlas: From Semantic Plausibility to Set-Level Utility in LLM Recommendation Agents," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2605.18805v1>
- [15] Lin et al., "UniGuardian: A Unified Defense for Detecting Prompt Injection, Backdoor Attacks and Adversarial Attacks in Large Language Models," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2502.13141v1>
- [16] Shaheer et al., "Detecting Prompt Injection Attacks Against Application Using Classifiers," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2512.12583v1>
- [17] Liu et al., "Automatic and Universal Prompt Injection Attacks against Large Language Models," *arXiv preprint*, 2024. Available: <http://arxiv.org/abs/2403.04957v1>
- [18] Wang et al., "WebInject: Prompt Injection Attack to Web Agents," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2505.11717v4>
- [19] Holgado-Sánchez et al., "Learning the Value Systems of Agents with Preference-based and Inverse Reinforcement Learning," *arXiv preprint*, 2026. Available: <http://arxiv.org/abs/2602.04518v1>
- [20] Lewis Newsham and Daniel Prince, "Personality-Driven Decision-Making in LLM-Based Autonomous Agents," *arXiv preprint*, 2025. Available: <http://arxiv.org/abs/2504.00727v1>