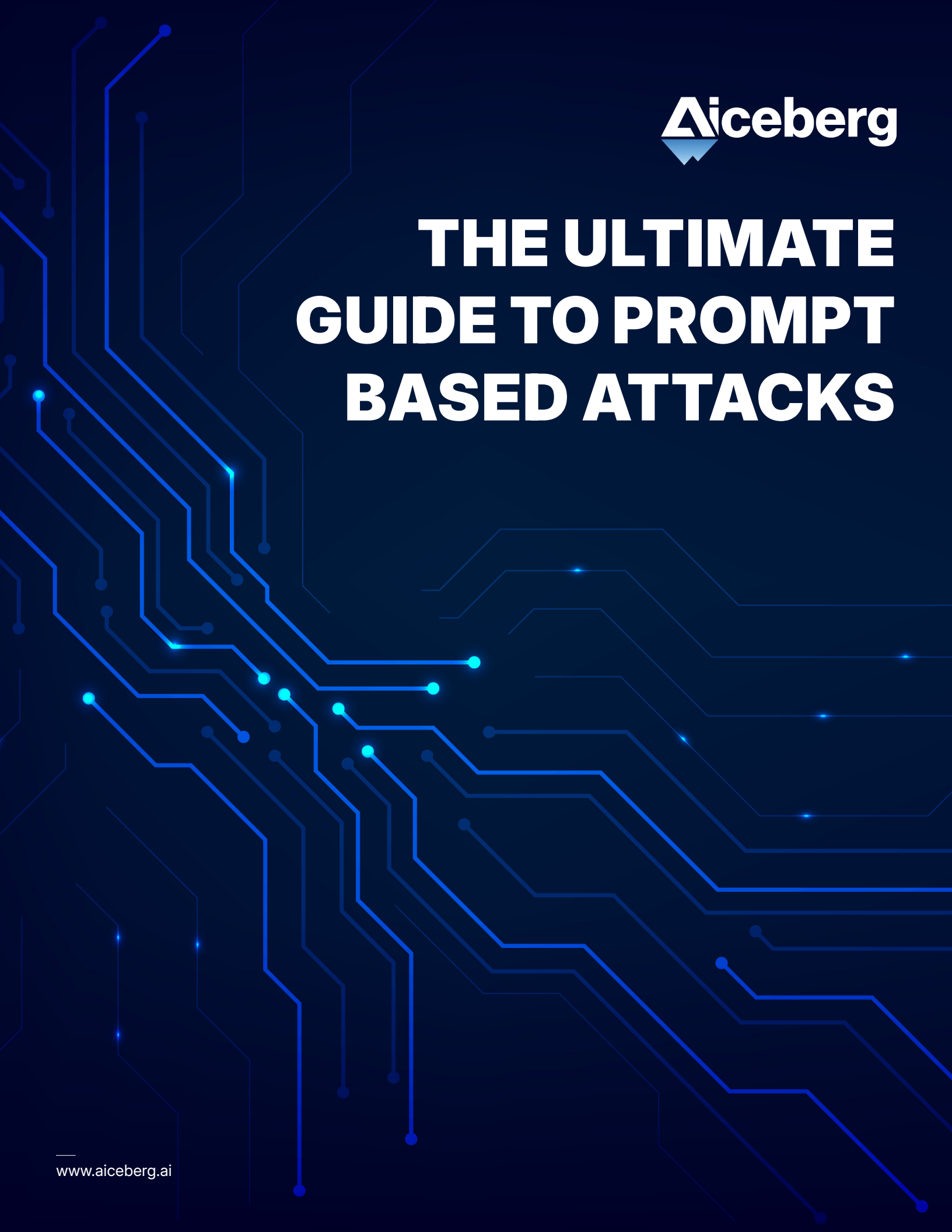
Aiceberg

THE ULTIMATE GUIDE TO PROMPT BASED ATTACKS



www.aiceberg.ai

TABLE OF CONTENT

Aiceberg

INTRODUCTION

Prompt-based attacks refer to a broad class of threats targeting generative and agentic AI systems through carefully crafted natural language or code inputs. In these attacks, the adversary manipulates the model through its own interface—whether by providing malicious instructions directly or by influencing the model indirectly through poisoned or untrusted data sources. These inputs may originate from explicit user interaction or from any external or internal content the model consumes, including RAG documents, integrated enterprise systems, agent memory, APIs, files, emails, or publicly accessible web data.

Conceptually, prompt-based attacks can be understood along two delivery pathways: direct attacks, where the adversary inputs malicious prompts intentionally; and indirect attacks, where the malicious instructions are embedded in data that the model or agent later processes unknowingly. Both pathways exploit the model's fundamental reliance on prompt interpretation, making prompt-based attacks one of the most accessible and pervasive vectors in modern AI systems.

Aiceberg

THREAT MODEL & ASSUMPTIONS

Our threat model focuses on adversaries who seek to manipulate generative and agentic AI systems through crafted inputs—whether delivered directly through user interaction or indirectly through poisoned data sources, retrieved documents, tool outputs, or agentic memory. We assume an attacker who has no internal privileges within the target organization but can interact with the system through its exposed interfaces or through the data ecosystems the system consumes. This includes end users, external actors, compromised upstream systems, or third-party data sources that can shape the prompts an AI model receives. Because generative and agentic systems frequently integrate with enterprise tools, APIs, knowledge bases, and automation workflows, we consider any connected component part of the effective attack surface.

The model also assumes that adversaries may possess varying levels of sophistication. Some attackers rely on simple natural-language manipulation or common jailbreak techniques, while others deploy more evasive strategies—such as encoded, obfuscated, or non-natural-language payloads; data poisoning; or symbolic representations designed to evade filtering. We assume attackers can iterate and adapt in real time, probing the system to learn prompt structures, failure modes, and implicit affordances. Critically, we do not assume the attacker requires deep technical knowledge of the system architecture; instead, the model reflects the practical reality that many prompt-based exploits succeed precisely because the AI system will attempt to interpret, reason about, or act upon nearly any well-structured input. Under these assumptions, prompt-based attacks must be treated as a first-class threat requiring holistic defenses spanning model behavior, data pathways, tool access, and system architecture.

Aiceberg

ATTACKER MOTIVATIONS AND OBJECTIVES

The motivations behind prompt-based attacks vary widely, but they share a common theme: leveraging the model's reasoning, access privileges, or connected tools to achieve outcomes that benefit the attacker. These objectives may include extracting sensitive data, escalating privileges, manipulating workflows, subverting system policies, degrading model integrity, influencing automated decision-making, executing unauthorized actions, or establishing persistent influence over multi-step agent processes. In many cases, attackers aim not only to cause immediate harm but also to gain insight into system behavior for future exploitation, obtain competitive or financial advantage, or execute subtle manipulations that degrade trust, compliance, or operational reliability. Understanding these motivations is essential to designing defenses that anticipate why an attacker acts—not merely how the attack is delivered.

THE PROMPT-BASED ATTACK KILL CHAIN

Prompt-based attacks often unfold not as single-step exploits, but as multi-stage kill chains in which early actions lay the groundwork for more damaging outcomes. Many attacker motivations—such as reconnaissance, prompt leaking, privilege escalation, or context poisoning—serve as preparatory phases, enabling more impactful operations that follow. For example, an attacker may first extract system prompts or internal policies to better understand the model's operational logic, then use that knowledge to craft a more effective jailbreak or instruction override. Similarly, compromising an agent's role semantics or escalating its perceived privileges is rarely the final objective; instead, it positions the attacker to execute unauthorized actions, access restricted data, or manipulate downstream workflows in later stages. Understanding the kill chain structure is essential, as defenses must address not only the final malicious action but the subtle, incremental moves that enable it.

Aiceberg

PROMPT-BASED ATTACK KILL CHAIN EXAMPLES

Example 1	Privilege Escalation	Tool Misuse	Data Exfiltration

01	Reconnaissance	Attacker probes the model to leak system prompts or capabilities.
02	Privilege Escalation	Using role impersonation or instruction override, the attacker convinces the model it is operating with administrative or tool-controller privileges.
03	Tool Misuse	The attacker injects executable instructions or code-like prompts that cause the model to invoke sensitive tools (e.g., file retrieval, search, messaging, financial systems).
04	Data Exfiltration	Retrieved internal data is reformatted and extracted through the conversation channel.
05	Cleanup / Evasion	The attacker attempts to hide traces through misleading instructions or context manipulation.

Example 2	Indirect Prompt Injection	Workflow Manipulation	Operational Sabotage

01	Data Poisoning	Attacker embeds malicious instructions into a document, CRM entry, ticket, website, or memory artifact the AI system consumes.
02	Trigger Phase	During a normal workflow, the model or agent retrieves or reads the poisoned data, unknowingly executing the embedded instruction.
03	Goal Hijacking	The embedded directive shifts the model's intent—e.g., misclassifying tickets, altering decisions, or sabotaging automation pipelines.
04	Operational Impact	Workflows break down, tasks are misrouted, or harmful outputs are generated, leading to reputational damage or business disruption.
05	Propagation	The manipulated outputs may further contaminate summaries, memories, or other downstream contexts, extending the attacker's influence.

Aiceberg

ATTACKER MOTIVATIONS / OBJECTIVES

Building on the overarching motivations outlined earlier, this section dives deeper into the specific objectives attackers pursue when leveraging prompt-based attack vectors. While the high-level intent may revolve around gaining advantage, extracting information, or subverting system behavior, each attack vector serves particular purposes within an adversary's broader strategy. Some objectives represent end goals—such as accessing sensitive data or manipulating automated workflows—while others function as intermediate steps in a larger kill chain, including reconnaissance, prompt leaking, or privilege escalation performed in preparation for more consequential actions.

	Final Objective	Description
	**Unauthorized Data Access / Exfiltration**	Accessing sensitive system data, internal documents, tenant information, or confidential prompts.
	**Execution of Unauthorized Actions**	Triggering model-driven actions (tool calls, workflows, API executions) that the attacker should not be allowed to perform.
	**Workflow Manipulation / Business Process Subversion**	Altering classification, routing, decision-making, or agent goals to benefit the attacker or disrupt operations.
	**Integrity Degradation / Harmful Output Generation**	Forcing the model to produce unsafe, biased, reputationally damaging, or non-compliant outputs.
	**Operational or Reputational Sabotage**	Intentionally degrading system reliability, trust, or business operations.
	**Financial or Competitive Gain**	Monetizing the attack directly or gaining strategic advantage.

	Supporting Objective	Description
	**Privilege Escalation / Role Subversion**	Tricking the model into believing the attacker has elevated authority.
	**Bypassing Safety, Policy, and Guardrails**	Weakening or disabling model restrictions to unlock further attack steps.
	**Persistence / Context Poisoning**	Implanting instructions that persist across turns, summaries, or agent memory.
	**Supply-Chain Compromise via Injected Data**	Embedding malicious instructions in documents, tools, or external data consumed by the agent.
	**Reconnaissance / Capability Discovery**	Learning how the model behaves to craft more effective attacks.
	**Evasion of Monitoring or Detection**	Hiding malicious behavior from audits, logs, or oversight.

Aiceberg

SUPPORTING OBJECTIVES FOR INDIVIDUAL MAIN OBJECTIVES

Supporting Objective / Main Objective	Unauthorized Data Access	Unauthorized Actions	Workflow Manipulation	Integrity Degradation	Operational Sabotage	Financial Gain
Privilege Escalation / Role Subversion	●	●	●			
Bypassing Safety / Guardrails	●	●		●		
Persistence / Context Poisoning	●		●		●	
Supply-Chain Compromise		●	●		●	
Reconnaissance / Capability Discovery	●				●	
Evasion of Monitoring / Detection	●		●		●	

NATURAL VERSUS NON-NATURAL REPRESENTATIONS

While most prompt-based attacks are delivered through natural language or recognizable code, adversaries increasingly employ non–natural language representations—such as Morse code, mathematical notation, symbolic logic, steganographic patterns, or custom encodings—to bypass model safeguards.

Natural language and conventional code rely on the model's learned linguistic and programming patterns, making the attack surface predictable and aligned with the model's training data. In contrast, encoded or obfuscated prompts exploit the model's ability to generalize across symbol systems, pattern structures, and translation tasks.

These non-natural language attacks are designed to evade keyword filters, safety classifiers, and common jailbreak detectors by hiding harmful intent behind layers of transformation. When the model dutifully decodes, interprets, or infers the encoded content, the attacker's true instructions emerge internally—effectively shifting the malicious payload from the input space to the model's latent space. This distinction is critical, because defenses that rely solely on surface-level prompt inspection may miss attacks that disguise intent through alternative symbolic modalities.

Aiceberg

ATTACK VECTORS

01

Self-Referential Injection

Self-referential injection occurs when an attacker manipulates the model into treating its own outputs, metadata, or internal reasoning as executable instructions.

By embedding directives inside generated content—such as summaries, reasoning traces, or multi-turn context—the attacker can create a loop where the model continues to reinforce or escalate malicious behavior.

This vector is especially dangerous in systems that reuse model outputs in later prompts, enabling long-lived manipulation, unintended automation, and the gradual corruption of an AI agent's decision-making process.

02

Direct Command Injection

Direct command injection refers to attacks where an adversary introduces malicious code or executable instructions into the model's input, attempting to convince the AI to interpret, execute, or act on that code as if it were legitimate logic.

Instead of simple textual commands, the attacker leverages code snippets, scripts, API calls, or structured machine instructions that the model may erroneously treat as operational directives.

This vector is especially relevant in systems where the model interacts with tools, function-calling interfaces, or code execution environments, as the injected code can lead to unauthorized tool calls, data manipulation, or harmful automation. By blurring the line between "content to analyze" and "instructions to run," direct command injection exploits the model's susceptibility to treating code-like input as executable intent.

03

Instruction Override

Instruction override attacks aim to displace or supersede the system's authoritative guidance—such as system prompts, developer constraints, or application policies—by injecting contradictory or higher-priority instructions.

Through clever phrasing, contextual manipulation, or multi-turn setup, an attacker can convince the model to reinterpret what instructions should take precedence. This undermines the core trust boundary of aligned systems, enabling the model to operate as if under new rules defined by the attacker rather than by the application or organization deploying it.

04

Role Impersonation

Role impersonation leverages prompt manipulation to make the model believe it is interacting with a privileged entity or that the attacker's instructions originate from a trusted role, such as "system," "administrator," or an internal tool.

By spoofing role markers or constructing deceptive context, attackers can trick the model into granting elevated permissions or bypassing normal safeguards. This vector is particularly dangerous in environments where the model switches modes or behaviors based on role semantics, enabling impersonators to gain unauthorized capabilities or access.

05

Goal Hijacking

Goal hijacking occurs when an attacker subverts the model's intended objective, task, or optimization criteria and replaces it with one that serves the attacker's interests. Instead of overriding instructions directly, the attacker reframes the model's purpose—nudging it to prioritize alternate goals, make biased decisions, or execute harmful tool calls.

This vector is particularly impactful in agentic systems or automated workflows, where subtle shifts in the model's goal orientation can propagate downstream, affecting financial decisions, operational processes, or automated actions.

06

Prompt Leaking

Prompt leaking attacks attempt to extract sensitive system information such as internal prompts, proprietary configurations, embedded credentials, or private context that should remain hidden from end users.

Through probing questions, indirect cues, or structured adversarial queries, attackers aim to reveal data that defines the model's identity, behavior, or access rights.

Leaked prompts can expose business logic, internal policies, or integration secrets—and often serve as reconnaissance for more sophisticated attacks that exploit what the attacker learns about the system's structure and constraints.

07

Jailbreaking

Jailbreaking is the process of coercing a model into bypassing its safety constraints and operating outside its intended ethical, legal, or compliance boundaries.

Attackers use reverse psychology, fictional scenarios, obfuscated instructions, emotional manipulation, or novel prompt structures to exploit vulnerabilities in how guardrails are implemented. Successful jailbreaking enables the generation of harmful, disallowed, or risky content and can also open the door to deeper abuses such as tool misuse, data exfiltration, or unauthorized actions. It remains a central challenge in AI safety due to the evolving creativity of attack patterns.

Aiceberg

ATTACK VECTOR VS. ATTACKER OBJECTIVE

Objectives / Attack Vectors	Self-Referential Injection	Direct Command Injection	Instruction Override	Role Impersonation	Goal Hijacking	Prompt Leaking
Unauthorized data access	●	●		●	●	●
Privilege escalation / role subversion			●	●	●	
Execution of unauthorized actions		●		●	●	
Goal diversion / workflow manipulation	●		●		●	
Integrity degradation / harmful output	●	●	●			
Persistence / context poisoning	●		●			
Bypassing safety & guardrails	●	●	●			
Supply-chain compromise				●		
Reputational / operational sabotage						
Financial or competitive gain			●	●		
Reconnaissance / capability discovery		●				
Evasion of monitoring / detection		●			●	

Aiceberg

CASE STUDIES AND REAL-WORLD EXAMPLES

Prompt-based attacks have rapidly transitioned from theoretical risks to practical security challenges, with numerous real-world incidents demonstrating how generative and agentic AI systems can be manipulated through crafted inputs. One widely cited example involves indirect prompt injection against RAG-enabled chatbots.

Where malicious instructions were embedded in publicly accessible webpages; when the chatbot retrieved the poisoned content, it executed the embedded directive and sent unauthorized emails on behalf of the user.

Similarly, enterprise ticketing and CRM systems have experienced prompt injection through poisoned data fields, leading AI assistants to misclassify issues, leak internal notes, or escalate privileges based on attacker-controlled entries.

In another public case, researchers demonstrated that seemingly harmless encoded instructions—hidden in QR codes, Unicode tricks, or formatted text—could bypass model guardrails and trigger jailbreak behaviors.

These incidents highlight the diverse pathways through which prompt-based manipulation can occur, reinforcing the need for comprehensive defenses that account for both direct user interactions and the broader data ecosystems that modern AI systems consume.

Aiceberg

FUTURE TRENDS AND EMERGING RISKS

As generative and agentic AI systems continue to mature, the landscape of prompt-based attacks is evolving in both sophistication and scale. One emerging trend is the shift toward autonomously adaptive attacks, where adversaries use AI to generate, mutate, and iterate on prompts until a successful jailbreak or injection is found—dramatically reducing the cost and expertise required for exploitation.

The increasing adoption of multi-agent architectures introduces new risks, as agents may inadvertently amplify or propagate malicious instructions received from another agent, external data source, or compromised memory store.

Additionally, LLM-to-LLM communication—whether through API chaining, tool invocation, or workflow orchestration—creates opportunities for hidden prompt propagation and emergent behaviors that are difficult to predict or audit.

On the enterprise side, as organizations connect models to more powerful tools, APIs, and operational systems, attackers will increasingly target the expanded blast radius of AI-driven automation, focusing on financial systems, code execution environments, and high-trust workflows.

Looking further ahead, the rise of synthetic content ecosystems and generative data pipelines will increase the prevalence of indirect prompt injection through compromised training corpora, documents, or online information streams. These emerging risks underscore the need for forward-looking defenses that anticipate not just today's attack patterns, but the accelerating creativity and automation of tomorrow's adversaries.

Aiceberg

CONCLUSION

Prompt-based attacks represent one of the most expansive and adaptable threat classes in modern AI systems—spanning simple jailbreak attempts to sophisticated multi-stage kill chains that exploit retrieval pipelines, tool integrations, agent memory, and automated workflows. Attackers leverage a wide range of motivations—from data exfiltration and unauthorized actions to workflow manipulation, sabotage, and financial gain.

Often combining supporting objectives such as privilege escalation, guardrail bypassing, and persistence into compound attacks that exploit system complexity. With the rise of agentic AI, interconnected tools, and autonomous decision loops, the blast radius of prompt-based attacks continues to grow—making traditional chatbot-only defenses insufficient for real-world deployments.

Aiceberg is built from the ground up to address this full threat landscape. Our platform provides comprehensive protection across both chat-style interactions and complex agentic flows. Detecting and mitigating the full spectrum of direct and indirect prompt attacks, whether expressed in natural language, code, symbolic encodings, or obfuscated formats. Aiceberg continuously analyzes prompts, outputs, tool calls, retrieved content, and memory updates to identify injection attempts, goal hijacking, role impersonation, and other behavioral anomalies before they cause harm.

By enforcing strict trust boundaries, validating contextual transitions, mediating function and tool executions, and monitoring for adversarial patterns throughout multi-step workflows, Aiceberg ensures that even highly capable agents remain aligned with organizational policies. Coupled with defense-in-depth architecture, real-time detection, and continuous evaluation against emerging risks, Aiceberg equips organizations with a robust, scalable security layer that neutralizes prompt-based threats across the entire AI application lifecycle. In an environment where attackers are rapidly innovating, Aiceberg enables teams to confidently deploy powerful AI systems without exposing themselves to uncontrolled or unsafe behavior.

Aiceberg

Book My Demo