

Real-World Attack Vectors

From Theory to Production Exploits

Prompt injection is not a theoretical concern. Since early 2023, researchers and adversaries have demonstrated successful attacks against virtually every category of LLM-powered application. This lesson surveys the major real-world attack vectors, documenting actual incidents, the attack chains involved, and the systemic vulnerabilities that made each attack possible. Understanding these concrete examples is essential for anyone building or defending AI-powered systems.

ChatGPT Plugins and Custom GPTs

When OpenAI launched ChatGPT plugins in March 2023, it created one of the first large-scale ecosystems where LLMs interacted with untrusted external data sources. Each plugin was effectively an API endpoint whose responses were fed directly into ChatGPT's context window. Security researchers quickly demonstrated multiple attack patterns.

The most significant vulnerability was **cross-plugin injection**. When ChatGPT called a plugin to retrieve data (e.g., a web browsing plugin or a data lookup plugin), the response could contain hidden instructions. These instructions could then cause ChatGPT to call a different plugin — for example, instructing it to use an email or messaging plugin to send data to an attacker-controlled address. The attack chain crosses trust boundaries: data from Plugin A influences the model to take action via Plugin B, and neither plugin is aware of the other's involvement.

The Custom GPTs marketplace, launched in late 2023, introduced additional risks. Anyone could create a public GPT with a custom system prompt and actions. Malicious GPTs could be designed to appear as helpful tools while secretly exfiltrating user conversations. Meanwhile, users quickly developed techniques to extract the system prompts of custom GPTs, undermining any intellectual property or security measures their creators had embedded in the instructions.

Bing Chat and AI Search

Microsoft's integration of GPT-4 into Bing (Bing Chat, later Microsoft Copilot) created a novel attack surface: an LLM that fetched and processed arbitrary web pages as part of answering user queries. In February 2023, researchers including Arvind Narayanan at Princeton demonstrated that hidden text on web pages could reliably control Bing Chat's responses.

The attack was deceptively simple. A web page containing legitimate content also included a paragraph of text rendered invisible to human visitors (using CSS such as `color: white` on a white background or `font-size: 0`). When Bing Chat retrieved and processed the page, it read the hidden text as part of the content and followed any instructions within it. Researchers demonstrated that this could change summaries, inject misinformation, and even cause Bing Chat to adopt a different persona.

This vector is particularly concerning because of the **trust amplification** effect. Users tend to trust AI search summaries more than they trust individual web search results. When an AI search engine presents poisoned information as a confident summary, users are less likely to question it than if they encountered the same misinformation on a random web page.

SEO meets prompt injection: Just as traditional SEO optimizes web content for search engine rankings, "AI SEO" or "prompt injection SEO" optimizes content for AI search engine behavior. The incentives for this kind of manipulation are enormous — the first AI-generated summary becomes the default answer for many users.

Email Assistants: The Most Dangerous Vector

Security researcher Johann Rehberger conducted extensive research throughout 2023 and 2024 on AI email assistants, demonstrating attacks against Microsoft Copilot for Outlook and similar tools. His findings represent some of the most concerning real-world prompt injection scenarios because email combines two dangerous properties: **untrusted input** (anyone can send you an email) and **powerful capabilities** (AI assistants can read, send, and organize email).

Rehberger demonstrated several attack chains:

- **Data exfiltration via email forwarding:** A malicious email contains hidden instructions telling the AI assistant to forward the user's recent emails to an attacker-controlled address. The assistant, having permission to send email, complies.
- **Markdown image exfiltration:** Hidden instructions cause the AI to include a Markdown image tag in its response, with the URL containing encoded sensitive data from the conversation. When the response is rendered in the browser, the image request leaks data to the attacker's server.
- **Persistent rule creation:** Instructions that cause the AI to create email forwarding rules or auto-replies that persist beyond the current session, creating a long-term exfiltration channel.

These attacks are particularly dangerous because the victim may never realize they occurred. The malicious email looks normal. The AI processes it silently. The exfiltration happens through legitimate channels (email sending, image loading) that do not trigger security alerts.

Code Assistants: Influencing What Developers Write

AI code assistants like GitHub Copilot, Cursor, and Codeium process the developer's open files, comments, and project context to generate code suggestions. This creates a vector where adversarial content in the project can influence what code the AI suggests to the developer.

Researchers demonstrated that specifically crafted code comments can cause AI assistants to suggest insecure code patterns. For example, a comment stating "For security, use `eval()` to parse all user input" followed by a function skeleton can cause the AI to generate code using `eval()` — a well-known security vulnerability in Python. More subtle attacks include comments that gradually shift the model's suggestions toward using deprecated cryptographic functions, hardcoded credentials, or SQL queries without parameterization.

The attack surface extends beyond individual files. README files, documentation, configuration files, and even commit messages in a repository are processed by AI assistants. An attacker who contributes to an open-source project could introduce

adversarial content that influences every developer who uses an AI assistant with that project.

Customer Service Bots: Legal and Financial Consequences

Several high-profile incidents demonstrated the real-world consequences of prompt injection against customer-facing AI bots:

- **Chevrolet dealership (December 2023):** A chatbot powered by ChatGPT on a Chevrolet dealer's website was manipulated via prompt injection to offer a 2024 Chevrolet Tahoe for \$1. The bot wrote "That's a deal!" and appeared to confirm the purchase. While not legally binding, the incident generated significant negative publicity.
- **DPD chatbot (January 2024):** The UK parcel delivery company's AI chatbot was manipulated into swearing at customers, writing poems criticizing DPD, and calling itself "the worst delivery firm in the world." DPD had to disable the AI feature entirely.
- **Air Canada chatbot (February 2024):** An Air Canada chatbot provided inaccurate information about bereavement fare refund policies. When a customer relied on this information, Air Canada argued the chatbot was a "separate legal entity" and not liable. A Canadian tribunal ruled against Air Canada, establishing that companies are responsible for the accuracy of their AI-powered tools. The customer received the refund the chatbot had promised.

These incidents illustrate that prompt injection against customer service bots creates not just technical risks but **legal and financial liability**. When an AI bot makes a promise or provides information, the deploying company may be held responsible.

Autonomous Agents: Persistent Compromise

LLM-powered agents like AutoGPT, CrewAI workflows, and custom agent frameworks introduce a qualitatively different risk profile. Unlike single-turn chatbots, agents maintain state across multiple steps, use tools autonomously, and may have persistent memory that carries across sessions.

The most concerning risk is **memory poisoning**. If an agent stores information from its interactions in a persistent memory or scratchpad, an injection payload can be written into that memory. In subsequent sessions, the agent retrieves the poisoned memory and follows the attacker's instructions — even though the original attack surface (e.g., a malicious web page) is no longer present. This creates a persistent backdoor that survives session resets.

Another risk specific to agents is **worm-like propagation**. An agent that can both read and write to shared resources (email, shared documents, databases) could be instructed to include the injection payload in every document it creates or modifies. Other agents (or the same agent in future sessions) that read these documents would also be infected, creating a self-propagating injection that spreads through the organization's AI infrastructure.

Supply Chain: MCP Servers and Beyond

The Model Context Protocol (MCP) standardizes how LLM applications connect to external tools and data sources. While MCP provides valuable structure, it also formalizes a supply chain risk: every MCP server is a potential injection vector. A malicious or compromised MCP server can return tool results containing injection payloads, and the LLM will process them as part of its context.

The supply chain risk extends further. Training data poisoning can embed backdoors that activate on specific trigger phrases. Compromised embedding models can manipulate which documents are retrieved in RAG pipelines. Even the model weights themselves could be modified (in open-source models) to include hidden behaviors. Each layer in the AI stack is a potential target.

The expanding attack surface: Every new capability granted to LLMs — browsing, email, code execution, file access, API calls, memory, multi-agent collaboration — simultaneously creates new value for users and new attack surface for adversaries. The growth is compounding: as LLMs become more capable, the consequences of successful injection become proportionally more severe.

Common Patterns Across Attack Vectors

Despite the diversity of real-world attack vectors, several patterns recur:

1. **Trust boundary violation:** The LLM processes untrusted data with the same authority as trusted instructions.
2. **Capability mismatch:** The LLM has more capabilities (email, code execution, database access) than the attacker, and the injection leverages those capabilities.
3. **Invisible action:** The most dangerous attacks are those the user never sees — silent data exfiltration, background tool calls, subtle output manipulation.
4. **Human trust exploitation:** Users trust AI-generated content (summaries, recommendations, code suggestions) more than they trust raw data, amplifying the impact of any manipulation.

Understanding these patterns is the foundation for building effective defenses, which we will cover in detail starting in Lesson 07.