WHITE PAPER · PUBLIC AI ASSISTANTS ON MICROSOFT AZURE

Securing a Public AI Assistant: Scope, Cost and Abuse Controls on Microsoft Azure

Brian Smith, CloudCentric · November 2026 · 17 minute read · 30 sources

Why a Public Assistant Is a Different Security Problem

An internal AI assistant operates within a security boundary where every user is authenticated. A public assistant has no signed-in user [1]. There is no account to rate-limit, suspend or bill [1]. An attacker can open unlimited fresh sessions [1]. The only identifying signals available to the system are network and browser signals such as IP address, TLS fingerprint and cookies [1].

The exposure of the model output is also different. Every answer a public assistant gives can be captured and published under the company's name. A company is accountable for what its assistant says. In Moffatt v. Air Canada (2024), a British Columbia tribunal rejected the argument that the airline's chatbot was a separate entity responsible for its own statements and held the airline liable for a refund policy the chatbot invented [2].

The case study is the visitor assistant on Lowcountry Magazine (lowcountry.one). Lowcountry Magazine is a regional publication covering restaurants, events, real estate and shops on the South Carolina and Georgia coast. I work on this assistant, which answers questions from the magazine's own listings database and shows results as cards linking to the magazine's pages.

Prompt injection is the practice of crafting input text to override system instructions. Denial of wallet is an attack designed to consume API budget through automated requests.

Documented Public Failures and Missing Controls

Scope Control and Cost Exposure

In December 2023, a visitor instructed a Chevrolet dealership's website assistant, built on ChatGPT, to agree with anything and to call each reply a legally binding offer. The assistant subsequently accepted $1 for a 2024 Tahoe vehicle [3]. In the same week, another visitor instructed the assistant to write Python code [3]. Dealers switched the assistant off [3]. The missing controls were a scope gate placed in front of the main model and an architectural restriction denying the assistant authority to state prices or offers [3].

From March to June 2026, developers discovered that Chipotle's public support assistant, Pepper, answered coding questions [4]. A developer reverse-engineered the chat protocol and published a coding-agent fork that used the assistant as a free model; the project listed adapters in progress for other retailers' assistants [4]. Chipotle changed the product to stop the use [4]. The missing controls were a scope gate, per-session token quotas, bot challenges and a chat endpoint configured to accept traffic only from the site's own pages [4]. Token economics refers to the cost model governing model API usage based on processed output and input text. Unmetered model access accumulates cost rapidly; Sysdig modelled the cost of stolen access to hosted models at about US$46,000 a day for one model at its quota ceiling across four regions [5].

Invented Commitments and Unfounded Policy

In a matter ruled in February 2024 regarding a November 2022 interaction, Air Canada's assistant instructed a customer to buy a full-price ticket and claim a bereavement discount afterwards [2]. The published policy required submission of the request before travel [2]. The tribunal found negligent misrepresentation and awarded damages [2]. The missing control was answers to policy questions drawn only from the published policy page, with a refusal when the page does not cover the question [2].

In April 2025, Cursor's AI support agent informed users that logouts were expected under a one-device policy that did not exist [6]. Users cancelled subscriptions publicly before a co-founder corrected the error [6]. The missing controls were grounding in published policy with an immediate escalation path when no supporting policy text is found, alongside an explicit disclosure that the agent is an AI [6].

In March 2024, reports highlighted that New York City's MyCity business assistant, reported to run on Microsoft Azure AI services, informed business owners that they could take a cut of workers' tips and refuse tenants with housing vouchers [7,8]. Disclaimers were added, but the errors continued [7]. The city shut the assistant down in February 2026; the mayor stated it cost around half a million dollars [8]. The missing controls were grounding in authoritative legal text with citations, mandatory refusal when no source supports an answer, and an evaluation set written by subject experts [7,8].

Unsafe Output Handling and Indirect Ingestion

In August 2025, researchers sent Lenovo's GPT-4 support assistant, Lena, a prompt of about 400 characters that caused it to reply in HTML containing an image tag pointing at an attacker's server [9]. When a support agent's console rendered the reply, the browser sent the agent's session cookie to that server [9]. Lenovo fixed the vulnerability after disclosure [9]. The missing controls were treating model output as untrusted data, plain-text rendering or strict sanitising, a Content Security Policy and HttpOnly session cookies [9].

Prompt injection refers to malicious input designed to override a model's system instructions. Indirect injection occurs when an assistant ingests third-party content containing malicious instructions. In December 2024, test pages with hidden text steered ChatGPT's search summaries toward positive reviews, even when visible reviews were negative [10]. Any assistant that reads content it did not write, such as listings or reviews, carries the same exposure [10].

Brand Damage and Over-Blocking

In January 2024, following a system update, DPD's parcel assistant swore at a customer and wrote a poem calling DPD the worst delivery firm in the world [11]. DPD switched the AI element off [11]. The missing controls were a red-team and regression run executed before each update, an output filter and a scope gate [11].

In January 2025, Virgin Money's assistant instructed a customer not to use 'words like that' after the customer typed the bank's own name [12]. The bank removed the assistant [12]. This incident represents over-blocking, where safety filters improperly flag legitimate interactions [12]. A test set containing the company's own brand and product names catches this specific failure before deployment.

DOCUMENTED PUBLIC FAILURES

Nine incidents, nine missing controls

2023

Chevrolet dealer

Agreed to sell a Tahoe for $1; wrote Python

Scope gate · no authority over prices

2026

Chipotle "Pepper"

Wrapped as a free coding model

Scope gate · token quotas · session tokens

2024

Air Canada

Invented a refund policy; airline held liable

Grounded answers · refuse when unsupported

2025

Cursor support

Invented a one-device policy

Grounding · escalation · AI disclosure

2024

NYC MyCity

Unlawful advice to businesses; shut down 2026

Cited sources · expert evaluation set

2025

Lenovo "Lena"

HTML reply leaked a staff session cookie

Plain-text rendering · CSP · HttpOnly

2024

DPD

Swore at a customer after an update

Regression and red-team runs · output filter

2025

Virgin Money

Blocked the bank's own name

Brand-vocabulary false-positive tests

2024

ChatGPT search

Hidden page text steered summaries

Untrusted-content screening · index hygiene

Anonymous Threat Vectors Mapped to OWASP

The OWASP Top 10 for Large Language Model Applications provides the reference taxonomy for public assistant risks [1]. An anonymous deployment alters the likelihood and impact of these vectors because the application accepts input from unidentified network endpoints.

Cost and Misuse

An anonymous assistant exposes model tokens to unauthenticated users, creating financial and resource risks. Off-topic misuse occurs when an attacker uses the assistant as an unbilled general-purpose model to generate code, write essays, or power an external API (LLM10: Unbounded Consumption) [1]. Scripted automated traffic presents a denial-of-wallet risk, where high volumes of long input prompts run up token costs or exhaust operational quotas for legitimate site visitors (LLM10: Unbounded Consumption) [1]. While no public incident with a reported bill was found, denial of wallet remains a recognised risk for public endpoints. Automated actors can also scrape an underlying catalogue one answer at a time, bypassing standard web anti-scraping controls (closest to LLM02 Sensitive Information Disclosure and LLM10 Unbounded Consumption).

Manipulation

Input manipulation risks increase when an application accepts untrusted text without prior identity verification. Direct prompt injection occurs when a user types instructions designed to override the system prompt, using techniques such as role-play scenarios or fake conversation turns (LLM01: Prompt Injection) [1]. Indirect prompt injection occurs when malicious instructions are hidden in content that the assistant retrieves, such as a property listing, user review, or vendor description (LLM01: Prompt Injection, LLM04: Data and Model Poisoning, LLM08: Vector and Embedding Weaknesses) [1]. On a site that publishes third-party content, any user who can post a listing can plant an indirect injection payload. Attackers may also use non-English text, Base64 encoding, leetspeak, or Unicode look-alikes to bypass safety checks, noting that Microsoft states Prompt Shields are primarily trained and evaluated on English (LLM01: Prompt Injection) [13].

Disclosure

Anonymous interfaces alter data disclosure boundaries. System prompt leakage occurs when an attacker extracts the hidden operational instructions of the assistant (LLM07: System Prompt Leakage) [1]. OWASP advises that the system prompt must not be treated as a secret or used as a security control [1]. Data leakage occurs when internal or non-public documents are indexed into the retrieval repository by mistake, making them retrievable by any site visitor (LLM02: Sensitive Information Disclosure) [1]. Unauthenticated visitors may also volunteer personal data into the chat window, which is subsequently written to application logs and transmitted to the model provider (LLM02: Sensitive Information Disclosure) [1].

Commitments and Harmful Output

An unauthenticated assistant can generate outputs that create organizational liability. The model may generate non-existent prices, refunds, discounts, or legal positions (LLM09 Misinformation, and LLM06 Excessive Agency if the assistant can act on them) [1]. The assistant may also output harmful or unlawful advice on regulated topics (LLM09: Misinformation) [1].

Output Handling and Actions

System integrations dictate the severity of output risks. Unsafe rendering occurs when an application displays model responses containing raw HTML or Markdown without sanitization, risking script execution in a visitor's browser or a staff administration console (LLM05: Improper Output Handling) [1]. When tools are configured to execute actions for anonymous users, prompt injection vulnerabilities translate into unauthorized system transactions (LLM06: Excessive Agency) [1].

Operational

Public assistants face operational stability risks related to model management and safety tuning. Vendor and model updates can shift live model behavior in public applications, as occurred in the documented DPD deployment failure (LLM03: Supply Chain) [1]. Over-blocking of legitimate customer prompts also presents an operational failure by degrading service utility, as occurred in the documented Virgin Money deployment, though this failure mode lacks an explicit OWASP category code.

THREAT MAP

Sixteen threats, their OWASP category and the main controls

THREAT	OWASP	MAIN CONTROLS
Off-topic use, token theft	LLM10	Scope gate, quotas, challenges
Denial of wallet	LLM10	Length cap, global quota, token alerts
Direct injection, jailbreak	LLM01	Prompt Shields, scope gate, output checks
Indirect injection via listings	LLM01 · 04 · 08	Index hygiene, document shields
System prompt leakage	LLM07	No secrets in the prompt, marker check
Data leakage	LLM02	Public-only index, read-only identity
Invented offers and policy	LLM09	Grounding with refusal, citations, commitment check
Harmful advice	LLM09	Grounding, content filters, domain tests
Visitor personal data	LLM02	PII filter, log retention
Catalogue scraping	closest LLM02 · 10	Per-session quotas, bot protection
Language and encoding bypass	LLM01	Shields, multilingual and encoding tests
Unsafe rendering	LLM05	Plain text, Content Security Policy
Malicious links	LLM01 · 05	Own-domain link allow-list
Tools acting for anonymous users	LLM06	No side-effect tools
Vendor and model changes	LLM03	Retest on every change
Over-blocking	none	Brand-vocabulary false-positive tests

Controls for Scope and Cost

Public AI assistants require explicit controls to define acceptable topics and limit consumption. An anonymous user interface exposes the underlying models and retrieval systems to arbitrary queries. Without operational boundaries, unexpected traffic patterns can exhaust service quotas or create unplanned financial liabilities.

Scope Gate

A scope gate classifies incoming messages before the application calls the primary model or queries a search index. This gate uses a smaller model to assign each message to one of three categories: on-topic, off-topic, or abusive. Off-topic messages receive a fixed reply.

Microsoft maintains no single managed scope-gate product, meaning this control is constructed from individual platform components. Recent deprecation timelines for Azure AI Content Safety custom categories point toward using Microsoft Foundry models for custom text classification [14]. The standard version of custom categories retires on 1 September 2026, and the rapid version retires on 30 April 2027 [14]. Processing incoming messages through an initial classification layer prevents off-topic prompts from generating unnecessary retrieval steps or long text outputs.

Input Limits

Input controls enforce bounds on message size and conversation length prior to model execution. Restricting character counts in a web browser input field provides user interface guidance, but it does not serve as a security boundary because automated scripts bypass front-end controls.

The API Management validate-content policy rejects requests larger than a set size with a 400 error [15]. The architecture sets a explicit maximum output token parameter on every call to the model. Enforcing fixed limits on input length and response generation caps the maximum processing effort allowed for any single interaction.

Rate Limits and Token Quotas

Azure API Management includes an `llm-token-limit` policy to control consumption based on token counts. The policy tracks usage across designated timeframes and returns an HTTP 429 status code when tokens per minute exceed set thresholds [16]. The policy also supports token quotas over longer periods, returning an HTTP 403 status code when usage reaches hourly, daily, weekly, monthly, or annual caps [16].

Quotas key on specific request attributes, such as the caller IP address or a signed session identifier. Counts are kept per gateway and are not aggregated across regions, and concurrent requests can briefly exceed the limit [16].

System stability relies on two distinct quota layers [16]. The first layer restricts tokens per individual IP address or session identifier [16]. The second layer applies a global token quota across the entire service by keying on a static constant value [16]. This global quota establishes the hard ceiling on spend on overall model consumption [16].

Bot Challenges and Session Tokens

Azure Front Door Web Application Firewall provides custom rate-limiting rules based on socket IP addresses over one-minute or five-minute evaluation windows [17]. Front Door Premium includes preview features that aggregate traffic by country, Autonomous System Number, or JA4 TLS fingerprint [17]. Each Front Door edge location evaluates these counters independently, which makes the effective global rate limit higher than the threshold configured on a single edge site [17].

Front Door JavaScript challenges and CAPTCHA tests do not execute on standard asynchronous XMLHttpRequests or Fetch API calls [18]. Web applications typically transmit chat interactions using these asynchronous patterns. The challenge belongs on the initial page request or the step that issues a short-lived chat session token; this placement is a design inference from the documented limit. Every subsequent chat message must submit this token for validation.

Spend Controls

Azure Budgets provide visibility into cloud spending, but they do not automatically halt service execution. Microsoft documentation states that Azure Budgets do not suspend consumption or alter resource availability [19]. Cost data arrives 8 to 24 hours after use, and budget evaluations execute every 24 hours [19].

A global token quota policy within API Management serves as the hard ceiling on spend on service execution [16]. To improve operational visibility, the `llm-emit-token-metric` policy streams token consumption metrics to Azure Application Insights [20]. Including custom metadata such as the client IP address or session identifier allows security teams to configure alerts that detect abnormal token burn within minutes [20].

DESIGNED DEFENCE PATH FOR AN ANONYMOUS VISITOR

Nine layers between the internet and the answer

ON-TOPIC
"Oyster roasts near Charleston this weekend?"

OFF-TOPIC
"Debug this Python function for me"

1	EDGE	**Front Door WAF**	Rate limit per IP, bot rules	● ●
2	SESSION	**Signed chat token**	Issued after a JavaScript challenge	● ●
3	GATEWAY	**API Management**	Size limit, token quota per visitor and in total	● ●
4	SCOPE GATE	**Small Foundry model**	On-topic, off-topic or abusive	● ●
5	SCREENING	**Prompt Shields**	Visitor message and retrieved listings	●
6	RETRIEVAL	**Foundry IQ**	Refuse below the relevance threshold	●
7	MODEL	**Main model**	No tools with side effects	●
8	OUTPUT	**Checks**	Marker, commitments, links, PII	●
9	RENDER	**Plain text**	Own-domain links only, CSP	●

Off-topic and abusive messages get a fixed reply at layer 4; the main model is never called.

Every layer logs to Log Analytics

Content Controls: Prompt Injection Screening, Groundedness, and Output Validation

Content controls protect the public assistant from prompt injection, ungrounded answers, and output manipulation. Prompt injection is the technique of submitting text that attempts to override the system instructions of a language model. Token economics is the cost structure based on the number of processed input and output units.

Screening Input and Retrieved Content

Public inputs and indexed documents require evaluation before reaching the generation model. Prompt Shields detects user prompt attacks, such as rule changes, fake turns, role-play, and encoding techniques. It also detects document attacks containing hidden instructions in documents, emails, or web pages. Microsoft Foundry enables user prompt attack detection by default. Document attack detection must be enabled explicitly in the settings. The standalone API checks one prompt and up to five documents per call [13].

Azure content filters classify prompts and completions across hate, sexual, violence, and self-harm categories at four severity levels. The service includes optional protected-material, PII, and custom blocklist filters. If the filtering system becomes unavailable, the request completes unfiltered and writes an error to the content_filter_results object. The receiving application must check for this error state and fail closed by blocking the response [21].

At the API layer, the Azure API Management llm-content-safety policy applies Content Safety and Prompt Shields at the gateway to both prompts and completions. The policy blocks non-compliant requests with a 403 status code. For streamed responses, the policy stops forwarding stream events on detection [22].

Public databases and third-party content require screening during data ingestion. Indexed content that the organization did not author directly, such as third-party listings, is treated as untrusted input. The ingestion pipeline strips hidden text at index time. The pipeline then scans remaining passages with Prompt Shields document detection at index time and at answer time.

Grounded Answers

A public assistant must restrict answers to retrieved enterprise facts. Azure OpenAI On Your Data, which contained a setting to limit responses to indexed data, was retired on 14 October 2026. Microsoft directs new architectural builds to Foundry IQ [23]. Refusal when retrieval finds no relevant data requires application logic. The application sets a minimum relevance threshold on search results. The system returns a fixed refusal string without calling the generation model when no retrieved passage clears the threshold.

Foundry IQ knowledge bases run retrieval workflows and return completed answers alongside source citations [24]. Groundedness detection in Content Safety compares a generated answer against its source documents. The Groundedness

detection feature supports English only and retires on 30 April 2027. Microsoft directs new implementations to use Foundry evaluation tooling for groundedness verification [25].

Output Checks and Rendering

Application logic must evaluate generated text before rendering it to the user browser. The application blocks any output containing a unique marker string placed inside the system prompt. The application blocks commitment language, including quoted prices, refunds, and guarantees, unless that exact language appears in a cited retrieval passage. The application strips links pointing outside the organization domain.

The user interface renders responses as plain text or strictly sanitised Markdown under a restrictive Content Security Policy. This prevents cross-site scripting and unauthorized script execution.

Controls for Actions, Monitoring and Testing

No Actions for Anonymous Users

A public assistant reads public content and performs no actions. It does not handle bookings, orders, refunds, emails or internal function calls. A visitor who needs an action must move to a web form or contact a person where identity is verified through standard methods. The managed identity that the assistant uses to query Azure AI Search holds read-only access limited strictly to the public search index. This architecture aligns with the OWASP LLM06 guidance for excessive agency [1].

Logging and Monitoring

Azure API Management sends LLM telemetry to Azure Log Analytics, populating the `ApiManagementGatewayLlmLog` table [26]. This table captures token usage metrics, the specific model deployment name, and optionally the full text of prompts and completions [26]. Logging full prompt text stores whatever personal data visitors enter into the chat interface. Data retention policies for these logs must enforce automatic deletion after a specified period, and role-based access control must restrict log viewing to authorized security personnel.

Microsoft Defender for Cloud AI threat protection monitors Azure OpenAI traffic to detect malicious activity [27]. Built on Azure AI Content Safety Prompt Shields and Microsoft threat intelligence, this service raises security alerts for jailbreak attempts, data leakage incidents and credential theft patterns [27].

Red-Team and Regression Testing

Security validation requires both automated security scanners and custom evaluation sets. The Microsoft Foundry AI Red Teaming Agent runs automated attack sets to evaluate system resilience [28]. The agent uses PyRIT, Microsoft's open-source Python Risk Identification Tool for generative AI, to execute attack strategies [28,29]. These strategies include Base64 encoding, ROT13 ciphering, leetspeak substitution, Unicode confusable characters, multi-turn conversations and Crescendo escalation techniques [28]. The tool calculates and reports an overall attack success rate metric [28].

Organizations can also run garak, an open-source vulnerability scanner developed by NVIDIA [30]. The garak scanner probes the deployment specifically for prompt injection vulnerabilities, successful jailbreaks, sensitive data leakage and toxic output generation [30].

Generic scanning tools must be complemented by a custom, site-specific test set. This test suite evaluates questions about the company's specific brand names and product lines. It tests responses to price inquiry edge cases, refund policy questions and regulated-topic inquiries. The suite executes tests across every language supported by the user interface. It also includes synthetic test documents containing planted hidden instructions to verify system prompt adherence. Security teams must run this custom test suite prior to initial system launch and repeat the full run following any change to the model version, system prompt text or underlying vendor services.

Case study: controls on the Lowcountry Magazine assistant and the Azure reference build

In place today

The Lowcountry Magazine visitor assistant functions as a specialized search interface. The assistant answers questions using a fixed set of search tools that query the magazine database for restaurants, events, homes, and shops. It returns results as visual cards that link to the magazine's own web pages. The assistant has no tool to book, buy, or change data.

Input constraints serve as the first layer of defense. Messages are limited to 300 characters. If a visitor submits a longer message, the system returns the text 'Question too long. 300 characters at most.' before any model call occurs. Control characters are stripped from all inputs. The instructions to the model treat tool output as data to prevent the model from executing text found in the database.

Traffic controls manage cost and availability. Each visitor IP address is limited to 20 questions per 10 minutes. Requests from known automated clients are refused. The endpoint accepts only same-origin JSON requests from the magazine's own pages. Prior turns in a conversation are signed to prevent a visitor from forging the dialogue history.

Output limits are defined at the application level. Each answer is limited to four tool steps, 12 result cards, 500 output tokens, and 1,200 characters. Model output is reduced to plain text. Markdown and URLs are stripped from the response.

The system displays only links to the magazine's own pages through the card interface.

Financial protection uses a monthly model budget of $100. This budget is tracked with reservations before each model call. When the system reaches the daily or monthly ceiling, the assistant switches to search-only answers and the site continues to function.

Data privacy is maintained through an anonymized logging process. The question log stores a salted hash of the visitor address. The system never stores the original address and strips personal details from the log.

A review identified specific gaps in this configuration. The current deployment has no prompt-injection classifier on input and no content filter on input or output. There is no explicit scope gate to handle off-topic requests. The rate limits are currently held in memory and reset if the application restarts.

The Azure reference build

The Azure reference build is a design that has not yet been deployed. It is designed to address the gaps identified in the current magazine assistant.

The reference build would add a scope gate using a small Microsoft Foundry model positioned in front of the main model. This smaller model would classify the intent of a message and refuse off-topic requests. The build would also incorporate Azure content filters on both input and output. These filters would address the current lack of content screening.

For security against injection, the reference build would add Prompt Shields with document detection. This would analyze retrieved listings for malicious instructions before they reach the model. A red-team run with the Foundry AI Red Teaming Agent and a site-specific test set would validate these defenses.

Traffic controls would be added at the network edge and at the gateway. Azure Front Door Web Application Firewall (WAF) would manage rate limits. This would include a JavaScript challenge on the page that issues the chat session token. Azure API Management would enforce llm-token-limit quotas per visitor and a global quota.

Monitoring would be centralized in the reference build. The system would send LLM logs to Log Analytics. It would also use llm-emit-token-metric alerts to notify administrators of unusual consumption patterns. These Azure services would provide a standardized way to manage the assistant as a production enterprise asset.

CASE STUDY · LOWCOUNTRY MAGAZINE VISITOR ASSISTANT

In place today, and what the Azure reference build would add

IN PLACE

- **Questions capped at 300 characters**
  Checked before any model call
- **20 questions per visitor per 10 minutes**
  Known bots refused; same-origin requests only
- **4 tool steps, 500 output tokens per answer**
  12 cards, 1,200 characters
- **$100 monthly model budget**
  Falls back to search at the ceiling
- **Read-only search tools**
  Nothing books, buys or changes
- **Plain-text output**
  Only links to the magazine's own pages
- **Signed prior turns**
  Forged history rejected
- **Hashed visitor address in logs**
  Personal details stripped

REFERENCE BUILD ADDS

- **Scope gate**
  Small Foundry model rejects code, essays and other off-topic requests
- **Prompt Shields**
  On visitor messages and on retrieved third-party listings
- **Content filters**
  On input and output
- **Token quotas**
  API Management, per visitor plus a global ceiling
- **Edge rate limits**
  Front Door WAF; challenge where the chat token is issued

- **Durable limits**
  Counters that survive a restart
- **Token-burn alerts**
  llm-emit-token-metric to Application Insights
- **Red-team runs**
  AI Red Teaming Agent plus a site-specific test set

CloudCentric Services for Public AI Assistants

CloudCentric designs, builds and tests public-facing and internal AI assistants on Microsoft Azure. Services include the deployment of scope gates, quotas, prompt-injection screening, grounded retrieval architectures, operational monitoring and automated red-team testing. Organizations implementing or securing an anonymous web assistant can contact CloudCentric at (844) 422-7000 or review service details at https://cloudcentric.tech/ai/ai-governance/.

Sources

1. OWASP Top 10 for LLM Applications 2025
2. CBS News: Air Canada chatbot discount ruling (Feb 2024)
3. VentureBeat: A Chevy for $1? Car dealer chatbots show perils of AI for customer service (19 Dec 2023)
4. Gizmodo: Should you hijack a corporate AI chatbot for free tokens? (4 Jun 2026)
5. Sysdig: LLMjacking, stolen cloud credentials used in new AI attack (6 May 2024)
6. The Register: Cursor AI support bot invents policy (18 Apr 2025)
7. The Markup: NYC's AI chatbot tells businesses to break the law (29 Mar 2024)
8. The Markup: Mamdani to kill the NYC AI chatbot (30 Jan 2026)

9. CSO Online: Lenovo chatbot breach highlights AI security blind spots (20 Aug 2025)
10. Search Engine Journal: ChatGPT search manipulated with hidden instructions (29 Dec 2024)
11. The Register: DPD AI chatbot swears, calls itself useless (23 Jan 2024)
12. Fortune: Virgin Money chatbot scolds customer (30 Jan 2025)
13. Microsoft Learn: Prompt Shields
14. Microsoft Learn: Content Safety custom categories
15. Microsoft Learn: API Management validate-content policy
16. Microsoft Learn: API Management llm-token-limit policy
17. Microsoft Learn: Azure Front Door WAF rate limiting
18. Microsoft Learn: WAF JavaScript challenge
19. Microsoft Learn: Create and manage budgets
20. Microsoft Learn: API Management llm-emit-token-metric policy
21. Microsoft Learn: Azure OpenAI content filtering
22. Microsoft Learn: API Management llm-content-safety policy
23. Microsoft Learn: Azure OpenAI On Your Data (retired 14 Oct 2026)
24. Microsoft Learn: What is Foundry IQ
25. Microsoft Learn: Groundedness detection
26. Microsoft Learn: API Management LLM logs
27. Microsoft Learn: Defender for Cloud AI threat protection
28. Microsoft Learn: AI Red Teaming Agent
29. Microsoft PyRIT (GitHub)
30. NVIDIA garak (GitHub)