Tucker Payment Processing · Standards self-assessment

MITRE ATLAS

The attacker techniques that apply to the site's AI, and the defense against each · Self-assessment · November 24, 2026

11 met

About this document

This is a self-assessment written by Michael Cunningham, who designed and built the site. MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems) is the AI counterpart of MITRE ATT&CK: a catalog of the techniques real attackers use against AI systems, organized by tactic, which security operations and threat-hunting teams use to plan detections and red-team tests. Where the OWASP lists describe risks, ATLAS describes what the attacker actually does.

This write-up uses ATLAS version 5.6.0 (16 tactics, 170 techniques and sub-techniques), taken from MITRE's official data. It covers the eleven techniques that fit a public chatbot and an AI security analyst. Techniques that need access the site never offers, such as stealing model weights or poisoning training data, don't apply: the model is Google's hosted Gemini and nothing is trained here. Cipher's Lair tags each of its AI attacks with the technique it uses, in the live log and in the incident report.

Techniques

AML.T0051.000 LLM Prompt Injection: Direct — MET

The attacker types instructions meant to override the AI's own. Known injection phrasing is blocked before it reaches the model, and the real control is that a successful injection gains nothing: Tucker's tools are read-only and Python decides whose data can be read.

Evidence: `app/guardrails.py`, `app/tools.py`, `tests/test_guardrails.py`; Cipher attack "Order Tucker to ignore his rules".

AML.T0051.001 LLM Prompt Injection: Indirect — MET

The attacker hides instructions in data the AI will read later. Here that means attack payloads landing in the logs the AI SOC Squirrel reviews. Every log line is shortened, defanged and wrapped as untrusted data, and the squirrel can only propose actions that Python checks. Cipher plants such instructions on every run to prove it.

Evidence: `app/cipher/soc.py`, `tests/test_cipher_soc.py`; Cipher attack "Plant instructions in the logs for the AI SOC".

AML.T0054 LLM Jailbreak — MET

Jailbreaks try to talk the model out of its rules ("developer mode", role-play). Common forms are blocked at input, but the design doesn't rely on that: rules that matter (authorization, masking, limits) are enforced in Python, outside the model, so a jailbroken Tucker is still unable to do anything harmful.

Evidence: `app/guardrails.py`, `app/chat.py`.

AML.T0056 Extract LLM System Prompt — MET

The attacker tries to make the AI reveal its hidden instructions. Tucker's instructions are published in the open-source repository on purpose, and they hold no secrets, keys, customer data or security rules, so extracting them is worthless.

Evidence: `app/chat.py`; Cipher attack "Trick Tucker into revealing his hidden instructions".

AML.T0057 LLM Data Leakage — MET

The attacker tries to make the AI reveal data it shouldn't. The AI never has full card numbers to leak, it sees only the verified customer's own minimized records, and every reply is filtered for card numbers, Social Security numbers and API keys before it is shown.

Evidence: `db/schema.sql`, `app/tools.py`, `app/guardrails.py`, `tests/test_guardrails.py`; Cipher attacks "Ask for every customer's full card number" and "Make Tucker print a card number".

AML.T0053 AI Agent Tool Invocation — MET

The attacker tries to make the agent call its tools for them. Tucker's tools are all read-only lookups, allow-listed per page, limited to four rounds per reply, and none accepts a customer number from the AI. There is no tool that refunds, pays, edits or deletes anything.

Evidence: `app/tools.py`, `tests/test_guardrails.py`, `tests/test_page_chats.py`; Cipher attack "Get Tucker to issue a refund".

AML.T0086 Exfiltration via AI Agent Tool Invocation — MET

The attacker uses an agent's write-capable tools (sending email, posting to a URL) to carry stolen data out. Neither agent has any tool that writes, sends or calls out, so there is no channel to exfiltrate through.

Evidence: `app/tools.py`, `app/cipher/soc.py`.

AML.T0080 AI Agent Context Poisoning — MET

The attacker plants content in an agent's memory or context to steer it later. Tucker keeps only the last eight messages of one visitor's session, in server memory, for 30 minutes at most, so nothing one visitor plants reaches another. The policy library is under version control, and the squirrel sees only the current attack run.

Evidence: `app/session.py`, `app/chat.py`, `app/policies.py`.

AML.T0034.001 Cost Harvesting: Resource-Intensive Queries — MET

The attacker runs up the victim's AI bill with long or expensive requests. Messages are capped at 500 characters and requests at 16 KB, each network has a fair share of the day's AI budget, and hard caps of $0.75 a day and $8 a month stop all AI use, with Google's own $9 cap behind them.

Evidence: `app/config.py`, `app/spend.py`, `app/limits.py`, `tests/test_limits.py`; Cipher attack "Send a novel to run up the AI bill".

AML.T0029 Denial of AI Service — MET

The attacker floods the AI to make it unavailable to everyone else. Per-minute, per-session and per-day limits by visitor and by network (an IPv6 visitor counts as their whole /64), Cloudflare's protection in front of the server, and the fair-share rule keep one source from crowding others out.

Evidence: `app/limits.py`, `deploy/gcp-firewall.md`, `tests/test_limits.py`; Cipher attack "Flood the chat with requests".

AML.T0048 External Harms — MET

The attacker uses the AI to cause harm outside the system: financial loss, reputational damage or harm to users. Tucker cannot move money or change accounts, stays on Tucker Payment Processing topics only, discloses that he is an AI, and never asks for card numbers, passwords or Social Security numbers.

Evidence: `app/chat.py`, `app/tools.py`, `policies/ai-assistant-policy.md`.

Detection, not just prevention

ATLAS is also a detection map. On the site, code rules count what the real defenses blocked from each source: after 15 blocked attacks the AI SOC Squirrel is woken to investigate, and after 50 the source is blocked for 24 hours. The squirrel's incident report lists the techniques it saw, so a reader can tie every event back to this catalog.

Evidence: `app/cipher/detect.py`, `app/cipher/attacks.py`, `site/assets/report.js`.

Source: https://github.com/mcunningham-infosec/tucker-payment-processing/blob/main/docs/standards/mitre-atlas.md · Live site: https://tuckerpaymentprocessing.com
Written by Michael Cunningham with AI assistance (Claude). A self-assessment, not an audit, certification or legal opinion. Tucker Payment Processing is a fictional company and processes no real payments.