Tucker Payment Processing · Standards self-assessment

SOC 2 Trust Services Criteria

How the site's controls line up with the AICPA criteria a SOC 2 auditor tests · Self-assessment · November 24, 2026

10 met **3 partly met**

About this document

This is a self-assessment written by Michael Cunningham, who has led SOC 2 Type II audits. It is not a SOC 2 report: only an independent CPA firm can issue one. It works like a readiness review, describing the design of each control and the evidence an auditor would ask for. A Type II report would also test that the controls operated over a period of months, which a new demonstration site cannot show.

The criteria are the AICPA's 2017 Trust Services Criteria (with the 2022 revised points of focus), the version auditors use today. In late 2026 the AICPA proposed updated criteria, including new attention to AI, for examinations starting on or after December 31, 2027. The common criteria (CC1 to CC9) make up the Security category every SOC 2 report covers; the other four categories are optional and are covered briefly at the end.

Security: the common criteria

CC1: Control environment — PARTLY MET

Integrity and accountability are set from the top: one named owner is responsible for the site, the policies are published, and the code is open for anyone to inspect. What one person cannot show: board oversight, separation of duties, background checks and an org chart, which an auditor would test in a real company.

Evidence: `README.md`, `policies/`.

CC2: Communication and information — MET

Information about the system is documented and shared. Inside: deployment and configuration guides, design notes in the code, and a nightly security report to the owner. Outside: the AI Policies, the About page, the security scorecard, the privacy notes and a published way to report vulnerabilities (SECURITY.md and security.txt).

Evidence: `deploy/README.md`, `app/report.py`, `site/about.html`, `SECURITY.md`.

CC3: Risk assessment — MET

Risks are identified and mapped to controls: the OWASP Top 10 for LLM Applications and the OWASP web Top 10 are each answered control by control, and the NIST AI RMF write-up covers AI risk. Fraud risk is designed against: prices, card and shipping address are set by the server, never the browser, and the AI cannot move money.

Evidence: `docs/standards/`, `app/store.py`, `tests/test_store.py`.

CC4: Monitoring activities — MET

Controls are checked continuously and independently. More than 70 automated tests check the controls. Cipher's Lair attacks the live site on demand and counts what the real defenses blocked. Five independent public scanners graded the site, and every gap they found was fixed the same day.

Evidence: `tests/`, `app/cipher/detect.py`, `docs/scans/`.

CC5: Control activities — MET

Controls are built into code and configuration rather than relying on people to remember them: identity checks, read-only AI tools, output masking, rate limits, spend caps, database permissions and security headers. Each is documented next to the code that enforces it.

Evidence: `app/chat.py`, `app/guardrails.py`, `app/config.py`, `db/roles.sql`, `deploy/apache-hardening.conf`.

CC6: Logical and physical access

MET

Admin access needs a Google identity with 2-step verification through the Identity-Aware Proxy; there is no public SSH. The firewall admits web traffic only from Cloudflare. Database logins use the operating system's identity, with no passwords, and least privilege (the store's login can call one function and nothing else). Secrets live only on the server, never in the repository. All traffic is encrypted with TLS 1.2 or 1.3. Physical security is inherited from Google Cloud.

Evidence: `deploy/gcp-firewall.md`, `deploy/postgres-store-login.md`, `db/roles.sql`, `app/config.py`.

CC7: System operations

MET

Detection: code rules watch attack traffic, alert after 15 blocked attacks from one source and block it for 24 hours after 50. Real-time alerts reach the owner by email through Google Cloud Monitoring, at most once per day per kind so they stay meaningful. Response: the AI SOC Squirrel writes an incident report for each attack run, and every real issue found is fixed and tested. Recovery: the server updates and restarts itself, and the demo data resets to a known state every night.

Evidence: `app/cipher/detect.py`, `app/alerts.py`, `deploy/alert-policy.json`, `deploy/auto-maintenance.sh`, `db/nightly_reset.sql`.

CC8: Change management

PARTLY MET

Every change goes through GitHub, where it is recorded, scanned by CodeQL and secret scanning, and checked for vulnerable dependencies by Dependabot. Deployment is one scripted command that pulls the reviewed code and restarts the app only when the app itself changed, waiting for any attack run in progress to finish first. The gap is independence: the builder also approves and deploys. An auditor would expect a second reviewer to approve changes before release.

Evidence: `deploy/update.sh`, `requirements.txt`.

CC9: Risk mitigation

MET

Business disruption is limited: hard spend caps stop runaway AI cost, the site degrades gracefully when the AI is down or over budget, and Cloudflare absorbs attacks. Vendors are known and bounded: Google (model and hosting), Cloudflare (DNS, TLS and protection) and Resend (alert email). Each receives only what it needs.

Evidence: `app/spend.py`, `app/chat.py`, `site/about.html`.

The optional categories

Availability

PARTLY MET

Services start on boot and come back on their own after updates, Cloudflare absorbs traffic floods, and limits stop any one visitor from exhausting the AI budget. There is one server and no failover or backup site, which is acceptable for a demo but would not meet a commitment to customers.

Evidence: `deploy/auto-maintenance.sh`, `app/limits.py`.

Processing integrity

MET

Orders are processed completely and correctly by design: prices come only from the database, quantities are capped, the database function re-prices every order itself, tampered requests are refused with nothing saved, and orders can never be edited or deleted. Tests prove each of these.

Evidence: `app/store.py`, `db/schema.sql`, `tests/test_store.py`.

Confidentiality

MET

Confidential information is protected and kept only as long as needed: no full card numbers anywhere, account data only for a verified customer, sensitive-looking output masked, demo data reset nightly and web logs deleted after two days.

Evidence: `db/schema.sql`, `app/guardrails.py`, `db/nightly_reset.sql`, `deploy/logrotate-apache2`.

Privacy

MET

A published data retention policy states what is collected and for how long. The one cookie is strictly necessary, there is no tracking or advertising, visitor IP addresses are never sent to the AI, and limit counters store only an anonymous code that changes every day, never a raw IP address.

Evidence: `policies/data-retention.md`, `app/limits.py`, `tests/test_limits.py`.

Where AI fits

SOC 2 does not yet have AI-specific criteria, but auditors already test AI systems through the existing ones: who can change the model's instructions (CC8), what data it can reach (CC6), how its failures are detected (CC7) and how its vendor is managed (CC9). The design here gives an auditor clean answers to each: the AI's instructions are in version control, it can only read, its guardrail events are logged and reported, and its vendor never receives IP addresses or card data.

Source: https://github.com/mcunningham-infosec/tucker-payment-processing/blob/main/docs/standards/soc-2.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.