Tucker Payment Processing · Standards self-assessment

PCI DSS v4.0.1

How the store and support chat are built to the payment card industry's security standard · Self-assessment · November 24, 2026

8 met **4 partly met**

About this document

This is a self-assessment written by Michael Cunningham, who spent more than 25 years in payments security running PCI programs. It is not a Report on Compliance or a Self-Assessment Questionnaire, and no Qualified Security Assessor has reviewed it. Tucker Payment Processing is fictional and handles no real cards; every card on the site is invented. This document shows how the site would stand against the PCI Data Security Standard (version 4.0.1, the current version) if those cards were real.

The most important PCI decision is made before any control: keep card data out. A full card number is never typed into this site, never sent by a browser, never stored and never shown to the AI. Cards exist only as the last four digits and a random token standing in for a vaulted card, the way a merchant using a payment processor's tokenization works. That design shrinks the cardholder data environment to almost nothing, which is the cheapest and strongest PCI control there is.

The twelve requirements

Requirement 1: Install and maintain network security controls — MET

The Google Cloud firewall accepts web traffic only from Cloudflare's published addresses, so nobody can reach the server directly. Admin access (SSH) is open only to Google's Identity-Aware Proxy, never to the internet, and the server does not even answer pings. A host firewall (ufw) runs on the server as well.

Evidence: `deploy/gcp-firewall.md`, `deploy/auto-maintenance.sh`.

Requirement 2: Apply secure configurations to all system components — MET

The web server hides its version, refuses TRACE, PUT and DELETE, lists no directories, serves no backup or configuration files, and limits request sizes. The API's built-in documentation pages are switched off. The configuration lives in version control, and Cipher's Lair checks these settings on every run.

Evidence: `deploy/apache-hardening.conf`, `app/main.py`; Cipher's nine security misconfiguration attacks.

Requirement 3: Protect stored account data — MET

No full card number (PAN) is stored anywhere. The cards table holds only the brand, the last four digits, the expiry date, the status and a random vault token. Sensitive authentication data (the CVV and PIN) is never collected. Card numbers are masked to the last four digits everywhere they appear.

Evidence: `db/schema.sql`, `app/tools.py`, `app/store.py`, `policies/card-security-and-privacy.md`.

Requirement 4: Protect cardholder data with strong cryptography in transit — MET

Only TLS 1.2 and 1.3 are accepted; TLS 1.0 and 1.1 were switched off and confirmed by SSL Labs (A+ on every address). HSTS is set for two years with preload, so browsers will only ever use HTTPS, and the domain has been submitted to the browsers' built-in preload list.

Evidence: `deploy/apache-hardening.conf`, `deploy/dns.md`, `docs/scans/2026-11-24/results.json`.

Requirement 5: Protect all systems and networks from malicious software

PARTLY MET

The server installs operating system security updates automatically every day, fail2ban blocks repeated attacks, and nothing on the site accepts file uploads. No anti-malware software runs on the Linux server. PCI allows that for systems not commonly affected by malware, but only with a documented, periodic evaluation (requirement 5.2.3), which a real company would keep on file.

Evidence: `deploy/auto-maintenance.sh`.

Requirement 6: Develop and maintain secure systems and software

MET

Secure development: every change on GitHub is scanned by CodeQL, secret scanning and Dependabot, dependencies are pinned, and more than 70 automated tests check the security behavior. Public-facing protection: Cloudflare's managed web application firewall sits in front of the site. Payment page scripts (requirement 6.4.3): the store and checkout pages run only the site's own scripts, enforced by a strict Content-Security-Policy, with no third-party code at all. Prices and totals are computed on the server only.

Evidence: `requirements.txt`, `tests/`, `deploy/apache-hardening.conf`, `app/store.py`, `tests/test_store.py`.

Requirement 7: Restrict access by business need to know

MET

Least privilege at every layer. The chat's database login can only read company data and write to its own logs. The store's login can call exactly one database function, which re-prices every order itself. The AI's tools only ever return the verified customer's own records, and each chat page gets only the tools it needs.

Evidence: `db/roles.sql`, `app/tools.py`, `tests/test_store.py`, `tests/test_page_chats.py`.

Requirement 8: Identify users and authenticate access

MET

Administrator access requires a Google identity with 2-step verification through the Identity-Aware Proxy, so multi-factor authentication protects every admin login. The application's database logins use the operating system's own identity rather than passwords, so there is no database password to steal. Customer identity checks lock after three failures in a session and ten per network per hour, and the session ID changes the moment a customer is verified.

Evidence: `deploy/gcp-firewall.md`, `deploy/postgres-store-login.md`, `app/tools.py`, `app/session.py`, `tests/test_limits.py`.

Requirement 9: Restrict physical access to cardholder data

MET

There is no office and no paper. The server runs in Google Cloud's data centers, and physical security is inherited from Google Cloud, which is assessed as a PCI DSS service provider. A real company would keep the provider's Attestation of Compliance and a responsibility matrix on file.

Requirement 10: Log and monitor all access

PARTLY MET

Monitoring is real: a nightly security report covers every visitor, guardrail event and attack, and real-time alerts reach the owner through Google Cloud Monitoring. But PCI requires keeping audit logs for at least 12 months, with three months immediately available. This demo deliberately keeps web logs for only two days because they hold visitors' IP addresses and there are no real cards to protect. A real deployment would send logs to a protected central store and keep them for the full year.

Evidence: `app/report.py`, `app/alerts.py`, `deploy/ops-agent-config.yaml`, `deploy/logrotate-apache2`.

Requirement 11: Test security of systems and networks regularly

PARTLY MET

Testing is frequent: Cipher's Lair runs 50 real attacks against the live site, and five independent public scanners checked it on November 24, 2026. PCI requires more formal testing: quarterly external scans by an Approved Scanning Vendor (ASV), internal vulnerability scans, an annual penetration test by a qualified independent tester, and detection of unauthorized changes to payment pages (requirement 11.6.1). A real company would contract for those.

Evidence: `app/cipher/attacks.py`, `docs/scans/`.

Requirement 12: Support information security with policies and programs — PARTLY MET

Published security policies cover card security, data retention and AI use, and a vulnerability disclosure policy is posted. A real program would add an annual risk assessment, security awareness training, service provider management and a tested incident response plan.

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

Where the AI fits

PCI DSS has no AI-specific requirement, but an AI that can see card data is in scope like any other system. Tucker is kept out of the card data environment by design: he never receives a full card number, his replies are filtered for anything that looks like one, and he cannot move money or change a card. The principle is the one PCI applies to any connected system: treat the AI as untrusted and control what it can reach.

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