Phoenix5981 Security — Security Research Summary

Contact: willworachat@gmail.com | Report date: 2026-07-03 | Report type: Audit Coverage Summary

Target	Stripe
Platform	HackerOne (managed program)
Engagement dates	2026-07-02 to 2026-07-03
Reports submitted	0
Outcome	NO FINDINGS — COMPREHENSIVE CLEAN RESULT
Estimated coverage	~70% of documented in-scope surface

Scope Tested

Core payments API (`api.stripe.com`) — publishable-key privilege boundary, webhook endpoint registration and URL validation, idempotency-key scoping, Search API query parser, cross-account object authorization (Customer, PaymentMethod, Charge). MCP server (`mcp.stripe.com`) OAuth 2.1 / Dynamic Client Registration flow. Stripe Agent Toolkit (open-source SDK, `github.com/stripe/ai`). Smokescreen (open-source egress-filtering proxy, `github.com/stripe/smokescreen`). Connect platform surface — connected-account creation, platform/connected-account authorization boundary, Standard OAuth authorize flow. Dashboard team-member role-based access control (View Only role).

Methodology Applied

- P10 systematic recon (subdomain enumeration, live-host triage across 316 discovered `*.stripe.com` hosts)
- P14 CVE/CWE prior-art research applied to the specific tech stack (MCP OAuth 2.1/DCR, Stripe webhook signing) before each testing pass
- H12 authentication/session checklist — OAuth redirect_uri validation (16 distinct bypass techniques total across two independent OAuth implementations: exact-match control, trailing slash, wrong host, subdomain confusion, protocol-relative URL, path traversal, HTTP downgrade, duplicate-parameter pollution, state-parameter smuggling)
- H13 cloud/infrastructure SSRF checklist — webhook-URL destination validation, cross-checked against source code of Stripe's own egress-filtering proxy
- P4 source-code-informed testing — read and analyzed Smokescreen's IP classification and ACL domain-matching logic directly rather than relying on black-box inference alone
- Two independent, isolated test accounts used to test genuine cross-tenant authorization boundaries (read and write paths) rather than single-account self-testing
- New rule (G26) applied: multi-tenant role-boundary results independently verified against the exact account ID under test before being treated as findings, after an initial false-positive was caught and corrected mid-engagement

Coverage

Twelve distinct attack-surface leads were carried to a definitive conclusion. Not yet covered: the live Connect OAuth token-exchange step (paused by mutual agreement at a routine test-mode onboarding step), Billing/Subscriptions edge cases, Radar, Issuing, Financial Connections, mobile applications (iOS/Android), and several smaller acquisition properties within scope.

Outcome / Findings Summary

No vulnerabilities of any severity were identified. Notable negative results, each independently verified rather than assumed:

- OAuth redirect_uri validation held under all tested bypass techniques on both the MCP server and the Connect Standard OAuth flow — two separate implementations, both robust.
- A webhook-URL destination-validation gap at the API input layer (accepting non-public IP literals that a stricter validator would reject) was traced to source and confirmed compensated by network-layer egress filtering in Stripe's own open-source Smokescreen proxy — not independently exploitable.
- Comprehensive two-account authorization testing (read and write, across Customer, PaymentMethod, and Charge objects, plus idempotency-key cache isolation) found no cross-tenant leakage.
- Connect platform/connected-account confused-deputy testing (five sub-tests spanning object access, account enumeration, login-link generation, and OAuth de-authorization) found no boundary violation.
- Dashboard role-based access control correctly restricted a View Only team member from API key visibility and team-member visibility on the account where that role was actually assigned.

$0. No reports submitted to the platform.
