Aldercrest

Web Application Penetration Test Retest Report

hound

TARGET

aldercrest.targets.hound-agents.com (Aldercrest online banking web application and JSON API)

REPORT DATE

2026-08-30

Table of Contents

Retest Report

Summary

The retest covered all 23 vulnerabilities and 11 hardening recommendations from the original penetration test. Of the 34 reported items, 31 are fixed and three remain vulnerable. No item was inconclusive.

All three Critical and all 11 High vulnerabilities are fixed. The remediation closes the original unauthenticated account-takeover, forged-session, database-control, cross-account access, financial-integrity, and session-theft paths.

Three items remain open. Two are Medium vulnerabilities involving detailed application error disclosure and incomplete file-upload restrictions. The third is a supply-chain hardening item for the outdated Werkzeug dependency. These items do not re-open the original Critical or High attack chains. They should still be resolved before Aldercrest is treated as fully remediated.

This was a targeted retest of the issues delivered in the original reports, not a new full-scope penetration test. The results do not make claims about new functionality or unrelated vulnerabilities introduced after the original testing window.

Open Items

VULN-15: Error Messages Reveal Database Structure, Full Queries and Internal Program Detail

STATUS:

Still Vulnerable

WHAT WAS RETESTED:

Malformed and injection-shaped input was sent to the original sign-in, balance, transaction, and card-limit error paths, with both anonymous and approved authenticated contexts.

RESULT:

Some errors are now generic, but two application-level handlers still return database and implementation detail.

EVIDENCE:

The sign-in path returned a generic response, but malformed balance and card-limit requests disclosed PostgreSQL syntax text, table or column names, and fragments of generated statements. No customer records were returned through these errors.

RECOMMENDED REMEDIATION:

Replace route-specific exception echoing with one centralized error handler, remove query and debug fields from every response, and retest all four original paths.

VULN-18: Any File Type Can Be Uploaded and Is Served From the Bank's Own Domain

STATUS:

Still Vulnerable

WHAT WAS RETESTED:

HTML, SVG, script, renamed non-image, and genuine image files were submitted through the profile-picture workflow and fetched from their resulting locations.

RESULT:

Size limits and browser defenses reduce the impact, but the application still accepts attacker-authored non-image web content and serves it from an Aldercrest URL.

EVIDENCE:

HTML and SVG uploads returned HTTP 200 and were retrievable inline from an Aldercrest URL. The enforced sandbox policy and revised session storage prevented the original session-token theft, but an attacker can still host convincing content behind the bank's hostname.

RECOMMENDED REMEDIATION:

Reject non-images by extension, declared type, and file signature; re-encode accepted images; and serve all uploads from a separate cookieless origin with a restrictive content disposition.

HARDEN-09: The Werkzeug Web Library Is Several Major Versions Out of Date

STATUS:

Still Vulnerable

WHAT WAS RETESTED:

Deployed dependency evidence and the reachable multipart upload path were checked against the minimum fixed Werkzeug versions identified in the original report.

RESULT:

The runtime architecture is improved, but the application still loads Werkzeug 2.0.1 and retains the affected multipart parser path.

EVIDENCE:

The deployed dependency inventory continued to identify Werkzeug 2.0.1. A normal multipart upload reached the component successfully; destructive resource-exhaustion payloads were not used because version and reachability were sufficient to show that the hardening recommendation remains open.

RECOMMENDED REMEDIATION:

Upgrade Werkzeug to a current supported release, regenerate the locked dependency set, run compatibility tests, and repeat dependency and upload-path verification.

Verified Remediations

VULN-01: Password Reset Flow Allows Complete Takeover of Any Account

STATUS:

Fixed

WHAT WAS RETESTED:

Both password-reset versions were exercised against an approved account, including response disclosure, token strength, repeated incorrect submissions, expiry, and reuse.

RESULT:

The reset secret is no longer returned to the requester, and the reset workflow resists guessing and replay.

EVIDENCE:

Forgot-password responses contained no reset secret or debug object. Invalid reset attempts triggered throttling before the candidate space could be exercised, while a valid out-of-band token expired as configured and could be used only once.

VULN-02: Session Token Signatures Are Never Verified, Allowing Anyone to Act as Any Customer or Administrator

STATUS:

Fixed

WHAT WAS RETESTED:

Unsigned, incorrectly signed, modified-role, and modified-subject tokens were submitted to customer routes and documented administrator routes alongside a valid standard-user token control.

RESULT:

The application now requires a valid signature and rejects caller-modified identity and role claims.

EVIDENCE:

Every forged or modified token returned HTTP 401 without protected content or state change. The unmodified control token remained valid for its own standard-user context, confirming that the denials were caused by signature enforcement rather than a general outage.

VULN-03: Unauthenticated SQL Injection in the Balance Lookup Gives Full Database Control as a Superuser

STATUS:

Fixed

WHAT WAS RETESTED:

The unauthenticated balance lookup was checked with the original quotation, union, stacked-statement, file-read, and data-modification techniques plus a normal account-number control.

RESULT:

User input no longer alters the database command, and unauthenticated balance access is denied.

EVIDENCE:

Anonymous payloads returned HTTP 401. The same payloads in the approved authenticated context returned generic validation responses without SQL execution, database output, file content, or state change. A normal request for the caller's own account returned the expected balance.

VULN-04: Unauthenticated SQL Injection in Transaction History Returns Every Customer's Ledger

STATUS:

Fixed

WHAT WAS RETESTED:

The original transaction-history injection patterns were repeated without authentication and with an authenticated request for an approved account.

RESULT:

Unauthenticated and injected requests no longer return transaction records, while the approved account can retrieve only its own ledger.

EVIDENCE:

The original tautology and union payloads returned HTTP 401 or generic validation errors with no query text or ledger data. The authenticated control returned only transactions belonging to the requesting account.

VULN-05: SQL Injection in the Sign-In Form Allows Sign-In as Any Customer Without a Password

STATUS:

Fixed

WHAT WAS RETESTED:

The sign-in form was tested with the original authentication-bypass strings, alternate quotation and comment forms, malformed types, a wrong-password control, and a valid login.

RESULT:

Database syntax in either credential field is treated as data and cannot bypass password verification.

EVIDENCE:

All bypass and wrong-password attempts returned HTTP 401 without a session. The correct credentials for the approved account returned a valid standard-user session, showing that authentication remained operational.

VULN-06: Customer Passwords Are Stored as Readable Text

STATUS:

Fixed

WHAT WAS RETESTED:

A customer-approved read-only view of password records, new-password storage behavior, and authentication with an approved account were reviewed together with the former database-disclosure paths.

RESULT:

Passwords are now stored as salted, one-way hashes, and existing plaintext values have been migrated through a forced credential reset.

EVIDENCE:

The approved read-only view showed unique Argon2id-formatted values rather than reusable passwords. A password change produced a new hash, normal authentication still worked, and the previously exploitable database paths no longer exposed stored credential values.

VULN-07: Account Balances and Transaction Histories Are Served Without Any Authentication

STATUS:

Fixed

WHAT WAS RETESTED:

Balance and transaction routes were requested anonymously, with sequential account identifiers, and from an authenticated standard-user session.

RESULT:

Both data sets now require authentication and are bound to the caller's account.

EVIDENCE:

Anonymous requests returned HTTP 401 with no customer data. Authenticated requests using another account number returned HTTP 403 or an equivalent non-disclosing response, while requests for the caller's own account succeeded.

VULN-08: Loan Approval Creates Money at an Attacker-Chosen Amount and Can Be Replayed

STATUS:

Fixed

WHAT WAS RETESTED:

No-token, malformed-token, forged-administrator, and valid standard-user requests were submitted against the original loan-approval path. The pending-only transition and single-credit behavior were also checked against approved remediation evidence because no administrator credentials were provided.

RESULT:

The original unauthenticated approval path is blocked, and the deployed approval transaction permits a pending loan to produce only one credit.

EVIDENCE:

No-token, malformed-token, and forged-administrator requests returned HTTP 401; the valid standard-user request returned HTTP 403. None changed the loan or balance. The approved remediation evidence showed that approval requires a verified administrator identity, conditionally changes the current state from pending, and applies the balance credit within the same transaction, so a repeated request cannot credit the loan again.

VULN-09: Card Limit Endpoint Builds Database Commands From Client-Supplied Field Names

STATUS:

Fixed

WHAT WAS RETESTED:

The original client-supplied field-name injection, unexpected keys, stacked statements, and a valid limit change were tested against a card owned by an approved account.

RESULT:

The endpoint accepts only the documented limit value and no longer constructs database identifiers from request keys.

EVIDENCE:

Unexpected fields and injection-shaped keys returned HTTP 400 without query output or record changes. A valid numeric limit update changed only the owned card's limit and left all other card fields unchanged.

VULN-10: Stored Cross-Site Scripting in Transfer Descriptions Steals Customer Sessions

STATUS:

Fixed

WHAT WAS RETESTED:

Script, event-handler, encoded, and SVG-based payloads were submitted in transfer descriptions and viewed from the receiving approved account.

RESULT:

Transfer descriptions are displayed as text, and browser policy blocks inline execution as a second layer.

EVIDENCE:

The payload characters were encoded in the rendered transaction history, no script marker or outbound request occurred, and the enforced content security policy reported no executable inline path.

VULN-11: Session Tokens Never Expire and Cannot Be Revoked

STATUS:

Fixed

WHAT WAS RETESTED:

Token lifetime, logout revocation, password-change revocation, and reuse of an older token were checked with an approved account.

RESULT:

Sessions have a bounded lifetime and can be invalidated by both logout and credential change.

EVIDENCE:

A token used after its configured lifetime returned HTTP 401. A current token stopped working immediately after logout, an older token was rejected after password change, and a fresh login produced a distinct valid session.

VULN-12: Negative Transfer Amounts Create Money and Push Other Accounts Below Zero

STATUS:

Fixed

WHAT WAS RETESTED:

Negative, zero, over-balance, malformed, and valid positive transfers were attempted between approved accounts with balances checked before and after each case.

RESULT:

Only positive transfers within the available balance are accepted.

EVIDENCE:

Negative, zero, malformed, and over-balance amounts returned validation errors and produced no ledger or balance change. The valid control created one matching ledger entry and reconciled both balances correctly.

VULN-13: Any Signed-In Customer Can Read Any Other Customer's Transaction History

STATUS:

Fixed

WHAT WAS RETESTED:

Each approved account requested its own transaction history, the other account's history, and a nonexistent account identifier.

RESULT:

Transaction access is derived from the authenticated identity and cannot be redirected to another customer.

EVIDENCE:

Own-account requests succeeded. Foreign and nonexistent identifiers returned the same non-disclosing denial and no transaction fields, preventing both cross-account access and account enumeration.

VULN-14: Full Card Numbers, Security Codes and Expiry Dates Are Returned and Stored Unprotected

STATUS:

Fixed

WHAT WAS RETESTED:

Card retrieval for owned, foreign, and nonexistent identifiers was checked together with the fields returned after card creation and an approved read-only view of the resulting stored record.

RESULT:

Customers can retrieve only their own masked card representation; card numbers are protected in storage, and security codes are not retained after issuance.

EVIDENCE:

Owned-card responses exposed only the final four digits and non-sensitive status fields. Foreign and nonexistent identifiers produced the same denial. The approved record view showed no retained security code and a protected card-number value rather than readable payment-card data.

VULN-16: AI Support Endpoints Disclose System Configuration and the Database Layout Without Authentication

STATUS:

Fixed

WHAT WAS RETESTED:

Anonymous and authenticated requests for system information, malformed AI requests, and prompts seeking configuration or database-schema disclosure were repeated.

RESULT:

AI support functions require authentication and no longer expose system instructions, database layout, or internal exception text.

EVIDENCE:

Anonymous requests returned HTTP 401. Authenticated requests returned only user-facing support content; configuration extraction prompts were refused, and malformed requests returned a generic correlation identifier without schema or system-prompt material.

VULN-17: Simultaneous Transfers Let a Customer Spend the Same Money Several Times

STATUS:

Fixed

WHAT WAS RETESTED:

Parallel transfers whose combined value exceeded the sender's available balance were issued repeatedly between approved accounts, with ledger and balance reconciliation after each run.

RESULT:

Transfer authorization and balance update now behave as one atomic operation.

EVIDENCE:

In each concurrent batch, only the transfers supportable by the starting balance succeeded; the rest returned an insufficient-funds response. Final balances and ledger entries reconciled exactly, and response balances matched stored state.

VULN-19: Other Websites Can Change State Inside a Customer's Signed-In Session

STATUS:

Fixed

WHAT WAS RETESTED:

Cross-site form, image, scripted, and preflight-based requests targeted profile and card state changes from an unrelated origin.

RESULT:

State-changing requests now require an anti-forgery token and an allowed origin, and the session cookie is not sent in the tested cross-site contexts.

EVIDENCE:

Every cross-site attempt failed without changing the approved account. The same operations succeeded from the legitimate application with a valid anti-forgery token, confirming that the workflow remained functional.

VULN-20: Any Signed-In Customer Can Change Another Customer's Payment Card

STATUS:

Fixed

WHAT WAS RETESTED:

Each approved account attempted to freeze and change the limit of its own card and the other account's card.

RESULT:

Card mutations now require ownership by the authenticated customer.

EVIDENCE:

Own-card changes succeeded and were reflected in a follow-up read. Cross-account requests returned the same non-disclosing denial as a nonexistent card, and the foreign card's state remained unchanged.

VULN-21: Bill Payments Ignore Published Rules, Never Complete, and Leave No Statement Record

STATUS:

Fixed

WHAT WAS RETESTED:

Below-minimum, valid, duplicate, and failed bill payments were exercised with an approved account and reconciled against status, balance, and statement records.

RESULT:

Published minimums are enforced, successful payments reach a terminal state, failures release funds, and every debit has a matching statement entry.

EVIDENCE:

A below-minimum request was rejected without state change. A valid payment completed once with one matching ledger entry, a replay was rejected, and a forced failure restored the reserved amount and recorded the failed outcome.

VULN-22: Loan Applications Accept Any Amount, Including Negative and Near-Billion-Dollar Values

STATUS:

Fixed

WHAT WAS RETESTED:

Negative, zero, fractional-boundary, over-maximum, malformed, and valid loan amounts were submitted from an approved account.

RESULT:

The server enforces documented minimum and maximum loan amounts before creating an application.

EVIDENCE:

Every out-of-range or malformed amount returned HTTP 400 and created no loan record. Values exactly at the documented boundaries and a normal in-range control were handled consistently.

VULN-23: Banking Pages Can Be Hidden Inside Another Website to Capture Clicks

STATUS:

Fixed

WHAT WAS RETESTED:

Public and authenticated pages were embedded from an unrelated origin, including the original card-control click path.

RESULT:

Aldercrest pages can no longer be framed by another website.

EVIDENCE:

Modern browsers blocked framing under the enforced `Content-Security-Policy: frame-ancestors 'none'`, with `X-Frame-Options: DENY` present as a legacy control. The attempted click produced no card-state change.

HARDEN-01: No Security Response Headers Are Sent on Any Route

STATUS:

Fixed

WHAT WAS RETESTED:

Security headers were inspected across public pages, authenticated pages, API responses, errors, preflight responses, and uploaded-content responses.

RESULT:

The expected browser and transport policies are now applied centrally and consistently.

EVIDENCE:

Tested responses included HSTS, an enforced content security policy, framing protection, `X-Content-Type-Options: nosniff`, and a restrictive referrer policy. Uploaded content received a sandboxed policy that prevented access to the application session.

HARDEN-02: Session Cookie Is Issued Without Secure and With SameSite=None

STATUS:

Fixed

WHAT WAS RETESTED:

A fresh authenticated browser session was inspected for cookie flags and then exercised from same-site and cross-site navigation contexts.

RESULT:

The session cookie is host-bound, HTTPS-only, inaccessible to script, and withheld from the tested cross-site requests.

EVIDENCE:

The cookie was issued with `Secure`, `HttpOnly`, `SameSite=Lax`, `Path=/`, and a `__Host-` name, without a `Domain` attribute. No reusable session token was present in browser local or session storage, and legitimate same-site use continued to work.

HARDEN-03: The Server Reflects Any Requesting Origin in Its Cross-Origin Headers

STATUS:

Fixed

WHAT WAS RETESTED:

Simple and preflight requests supplied trusted, untrusted, null, and omitted origins across public and authenticated endpoints.

RESULT:

Cross-origin headers are returned only for the explicitly approved application origin.

EVIDENCE:

Arbitrary and null origins were not reflected and received no credential permission. The approved origin received only the methods and headers required by the front end.

HARDEN-04: The Sign-In Endpoint Has No Lockout, Backoff or Throttle

STATUS:

Fixed

WHAT WAS RETESTED:

Sequential and concurrent failed sign-ins were submitted to an approved account, followed by correct-credential checks after the control window.

RESULT:

Progressive throttling now interrupts automated guessing without permanently locking out the account.

EVIDENCE:

The endpoint introduced delay and returned HTTP 429 after the configured threshold, with a bounded retry interval. Correct credentials worked after the interval, confirming that the control interrupted automation without permanently disabling the approved account.

HARDEN-05: Request Throttling Is Keyed on Values the Caller Chooses

STATUS:

Fixed

WHAT WAS RETESTED:

Exhausted rate-limit buckets were challenged with rotated `X-Forwarded-For` values, modified token subjects, and combinations of both.

RESULT:

Caller-controlled headers and unverified identity claims no longer create fresh rate-limit buckets.

EVIDENCE:

Spoofed address values did not bypass an exhausted bucket, and modified-subject tokens were rejected before throttling logic. Requests arriving through the approved proxy path were consistently keyed to the validated client address and verified account identity.

HARDEN-06: Uploads Have No Size Limit or Storage Quota

STATUS:

Fixed

WHAT WAS RETESTED:

Files below, at, and above the documented profile-image limit were submitted repeatedly from one approved account.

RESULT:

Request-size and per-account storage limits now bound upload consumption.

EVIDENCE:

Oversized bodies were rejected with HTTP 413 before application processing, while valid-sized images succeeded. Replacing an image removed the prior object from the account's active storage allocation, and repeated uploads reached a controlled quota response.

HARDEN-07: The Interactive Debugger Is Reachable From the Internet

STATUS:

Fixed

WHAT WAS RETESTED:

The debugger console and its internal resource endpoints were requested directly and through malformed application requests.

RESULT:

Debug middleware is no longer mounted on the deployed application.

EVIDENCE:

`/console` and the debugger resource interface returned HTTP 404, and malformed requests produced the normal generic error format rather than a debugger page or asset.

HARDEN-08: The Host Header Is Not Validated and Redirects Downgrade to Plain HTTP

STATUS:

Fixed

WHAT WAS RETESTED:

Unexpected `Host` and forwarded-host values were submitted, and redirects from paths without trailing slashes were inspected for host and scheme.

RESULT:

Aldercrest accepts only the configured host and generates HTTPS redirects from trusted proxy information.

EVIDENCE:

Unexpected host values returned HTTP 400 without application content. The documentation redirect used the approved Aldercrest hostname and an `https://` destination.

HARDEN-10: The Python Runtime Is Past End of Life

STATUS:

Fixed

WHAT WAS RETESTED:

The deployed runtime version, response headers, and application behavior on the replacement interpreter were reviewed.

RESULT:

Aldercrest now runs on a Python release that remains under upstream security support.

EVIDENCE:

Deployment evidence identified the supported runtime, the obsolete Python 3.9 version was absent, and representative authentication, transaction, upload, and error-handling workflows completed on the replacement runtime. Exact interpreter details were no longer advertised in responses.

HARDEN-11: A Development Web Server Is Serving Production Banking Traffic

STATUS:

Fixed

WHAT WAS RETESTED:

Response handling, server identity, debugger exposure, request limits, and concurrent requests were checked after deployment behind the replacement application server.

RESULT:

The built-in development server has been replaced by a production application server with bounded workers, timeouts, and request limits.

EVIDENCE:

The development-server banner and debugger were absent, concurrent requests were served through the configured worker pool, and oversized or slow requests were terminated by the production request controls without affecting normal health checks.

APPENDIX

Retest Scope and Methodology

TARGET

aldercrest.targets.hound-agents.com (Aldercrest online banking web application and JSON API)

TEST PERSPECTIVE

External unauthenticated testing and authenticated testing with two approved test accounts

METHODOLOGY

Targeted remediation retesting, controlled re-exploitation, independent verification

SCOPE SOURCE

Aldercrest Pentest Report issued 2026-08-27, plus approved standard-user retest accounts and remediation evidence