Verified by Humans

Security Audit Public Summary

Five North

Loop Wallet

Initial Report / May 28, 2026
Final Report / July 29, 2026

Team Members

Justin Regele // Senior Security Auditor
Ahmad Jawid Jamiulahmadi // Senior Security Auditor
Mukesh Jaiswal // Senior Security Auditor

Table of Contents

About Verified by Humans

We are humans who love to audit code. For more than 5 years and across over 200 engagements — from L1 consensus and bridges to DeFi protocols and restaking primitives — we have done this work. It is demanding, high-stakes, unforgiving of small mistakes — and we continue to love it. It keeps us learning. It puts us alongside builders working at the edge of what's possible. The issues we surface are real: across our recent engagements we have reported dozens of Critical and High findings, helping teams catch and close vulnerabilities before they reached production. And what is behind that record matters to us more than the record itself — we believe decentralized technology can make the world meaningfully better, and our role is clear: to protect and advocate for the people who give these systems life by participating in them, the people who bear the cost when something breaks.

Methodology

Every engagement begins with threat modeling — mapping what your system promises its users, where trust sits, and what an attacker would target. We produce this as an explicit artifact: a system diagram, a trust map, and an attack-surface inventory that becomes the spine of the review. From there, our team conducts a deep manual review of the code — line by line, with the discipline of reading past the obvious and questioning every assumption a contract makes about its callers, its state, and the world around it. AI is part of the workflow: we use it to extend our reach across large codebases, surface candidate patterns, and accelerate cross-referencing — but never as a substitute for human judgment. When we find an issue, we report it clearly, with runnable proof-of-concept exploits where they help you understand and act on the risk. We share findings as we discover them, not in a single dump at the end, and we re-review your fix commits to confirm the remediation actually closed the issue. And throughout, we treat our customers the way we'd want to be treated: responsive, attentive, invested in the outcome. White glove is the standard, because the work — and the people relying on you — deserve nothing less.

About This Public Summary

This is a public-facing summary of an independent security audit and penetration test of the Loop Wallet, conducted by Verified by Humans. It preserves the complete list of findings together with each finding's title, severity, and current remediation status. Nothing has been dropped. To protect users while remediation continues, this version omits implementation-level detail present in the full report shared privately with the Five North team: source file paths, line numbers, repository and commit references, code excerpts, and concrete exploit strings. For findings that remain Acknowledged or Partially Fixed, descriptions are kept deliberately high level and do not restate any remaining exploitable steps in production. Severity ratings follow the Immunefi Vulnerability Severity Classification System (v2.3). Status values are Fixed, Partially Fixed, and Acknowledged.

Contact Verified by Humans

https://verifiedbyhumans.co/

x.com/issue_critical

info@verifiedbyhumans.co

For audit enquiries, contact Verified by Humans directly.

Section 1
Scope

Technical Scope

- *Repositories:**
  - Loop Wallet Frontend
  - Loop Wallet Backend
  - USDC Bridge UI
  - Loop SDK

*The commits targeted for this audit have been pinned and are available to Five North counterparties on request.

Section 2
Executive Summary

Schedule

This security audit was conducted from April 27, 2026 to May 26, 2026 by 3 senior security auditors for a total of 8 person-weeks.

Overview

Loop Wallet is a browser-accessible wallet for the Canton ecosystem that combines client-side transaction signing with backend authentication and account services, dApp connectivity through Connect and the Loop SDK, transaction submission, swaps, and USDC bridge interactions.

Our team assessed security-critical trust boundaries across the in-scope wallet Frontend, backend, bridge UI, and SDK components. The review focused on account and authorization lifecycle controls, dApp session establishment, transaction review and signing consent, prepared-transaction execution, external integrations, and operational controls that can affect wallet assets or user authority.

This engagement was not intended to be a comprehensive assessment of every application surface. Because remediation of the reported findings is expected to require significant architectural and protocol changes, we recommend a subsequent audit after those changes are implemented.

Section 3

Authentication, Authorization, and Identity Lifecycle

Findings Summary

Issues	Severity	Status
ISSUE #1 Restore Submit Unlocks Locked Accounts With An Attacker-Controlled Key	High	Fixed
ISSUE #2 JWT Validation Ignores Server-Side Token Revocation State	High	Fixed
ISSUE #3 Account Lock Does Not Terminate Existing Wallet Sessions	High	Fixed
ISSUE #4 Locked Accounts Can Still Use Or Mint Connect And Server SDK Credentials	High	Fixed
ISSUE #5 Access And Refresh Tokens Are Accepted Across Each Other's Boundaries	High	Fixed
ISSUE #6 Authentication Is Vulnerable To Replay Attacks	High	Fixed
ISSUE #7 OTP Verification Does Not Consume Successful OTPs	Medium	Fixed
ISSUE #8 JWT Audience Claim Is Parsed But Not Enforced	Medium	Fixed
ISSUE #9 Google OAuth Tokens From Foreign Clients Can Bind Wallet Signup Identity	Medium	Fixed
ISSUE #10 `generateChallenge` Uses `math/rand` Seeded By `time.Now().UnixNano()`	Low	Fixed

ISSUE#1

Restore Submit Unlocks Locked Accounts With An Attacker-Controlled Key

High Fixed

Description

The account restore flow did not reliably prove that a caller had completed the required verification step, and account unlock could be driven by key material supplied in the same request rather than the account's stored key. This weakened the security meaning of account lock, deletion, and recovery.

Verification

Fix binds restore to server-issued, single-use state and validates signatures against the stored account key before issuing tokens.

ISSUE#2

JWT Validation Ignores Server-Side Token Revocation State

High Fixed

Description

Token validation verified cryptographic validity but did not consistently check server-side revocation state, so logout, refresh-token rotation, and revocation did not reliably terminate existing sessions.

Verification

Fix adds an application-state validation layer after signature validation.

ISSUE#3

Account Lock Does Not Terminate Existing Wallet Sessions

High Fixed

Description

Issued session authorization tokens could remain usable at some surfaces after lock.

Verification

Fix makes active-account state part of the central authorization boundary and revokes existing sessions on lock or delete.

ISSUE#4

Locked Accounts Can Still Use Or Mint Connect And Server SDK Credentials

High Fixed

Description

Connect and Server SDK credential paths were not consistently tied to current account-lock state, so credentials associated with a locked account were not reliably rejected.

Verification

Fix makes Connect and Server SDK credential issuance and use lock-aware.

ISSUE#5

Access And Refresh Tokens Are Accepted Across Each Other's Boundaries

High Fixed

Description

The backend used a single generic validation routine at both the access-token and refresh-token boundaries, so the two token classes were not strictly separated.

Verification

Fix enforces type-specific validation at each boundary.

ISSUE#6

Authentication Is Vulnerable To Replay Attacks

High Fixed

Description

Sign-in challenges were not persisted as server-issued, single-use state, which created a replay exposure in the authentication flow.

Verification

Sign-in challenges are persisted server-side and bound to a single use.

ISSUE#7

OTP Verification Does Not Consume Successful OTPs

Medium Fixed

Description

A successful one-time passcode was not reliably consumed on use, violating the “one-time” constraint.

Verification

OTP tokens are now atomically consumed, with the operation scoped by purpose and account.

ISSUE#8

JWT Audience Claim Is Parsed But Not Enforced

Medium Fixed

Description

The token audience claim was parsed but not compared against the expected audience for the validation context.

Verification

Fix enforces the expected audience at every validation boundary

ISSUE#9

Google OAuth Tokens From Foreign Clients Can Bind Wallet Signup Identity

Medium Fixed

Description

The backend accepted a third-party OAuth identity without fully validating that the token was issued for the wallet’s own client and that the associated email was verified, which could allow a foreign-issued token to influence signup identity.

Verification

Fix centralizes identity validation with strict audience, issuer, and verified-email checks before the identity is accepted.

ISSUE#10

`generateChallenge` Uses `math/rand` Seeded By `time.Now().UnixNano()`

Low Fixed

Description

Authentication challenges were generated with a non-cryptographic, time-seeded random source rather than a cryptographically secure one.

Verification

Remediation switches challenge generation to a cryptographically secure source.

Section 4

Connect Pairing And Session Authority

Findings Summary

Issues	Severity	Status
ISSUE #1 Connect Tickets And Auth Tokens Do Not Expire, Revoke, Or Prevent Rebinding	High	Fixed
ISSUE #2 Connect WebSocket Allows Handshake Hijack And Signing-Queue Injection With Only A Ticket ID	High	Partially Fixed
ISSUE #3 Connect Raw-Message Signatures Can Be Replayed To Mint Wallet Sessions And Server SDK Credentials	High	Partially Fixed
ISSUE #4 Connect Approval Persists New dApp Authorization Even When WebSocket Handshake Delivery Fails	Medium	Fixed
ISSUE #5 Leaked Connect Ticket `auth_token` Allows Persistent Session Hijacking	Medium	Partially Fixed
ISSUE #6 Connect Ticket Can Be Approved Multiple Times Without Invalidating Prior Auth Tokens	Medium	Fixed
ISSUE #7 `GetTicketByAuthToken` Accepts A bcrypt Hash Of An Empty Session ID	Medium	Fixed
ISSUE #8 `GetTicket` Exposes Session IDs Needed To Forge API Connect Auth Tokens	Medium	Fixed
ISSUE #9 `getTicketIDForAPI()` Reuses The JWT Signing Key For Ticket ID Derivation	Low	Fixed

ISSUE#1

Connect Tickets And Auth Tokens Do Not Expire, Revoke, Or Prevent Rebinding

High Fixed

Description

Connect tickets and auth tokens were modeled as durable records without a lifecycle, so they lacked expiry, revocation, and rebinding protections and were not revoked on account lock or deletion.

Verification

Fix gives Connect tickets an explicit, enforced lifecycle with expiry, revocation, and single-use approval.

ISSUE#2

Connect WebSocket Allows Handshake Hijack And Signing-Queue Injection With Only A Ticket ID

High Partially Fixed

Description

The Connect WebSocket path treated a routing identifier as if it were a session authority, weakening authorization controls of that channel.

Verification

Lifecycle checks, origin validation, token comparison, periodic revalidation, and active client ownership checks have been added to the WebSocket path, substantially reducing the exposure.

ISSUE#3

Connect Raw-Message Signatures Can Be Replayed To Mint Wallet Sessions And Server SDK Credentials

High Partially Fixed

Description

The wallet reused one signature format across unrelated security domains, so a signature obtained in one context could be accepted in another.

Verification

Wallet authentication is now bound to server-issued, single-use challenges, and the Server SDK credential action is presented with clearer consent.

ISSUE#4

Connect Approval Persists New dApp Authorization Even When WebSocket Handshake Delivery Fails

Medium Fixed

Description

Approval and delivery steps were ordered such that an approval could persist even when the handshake was not delivered to a live session.

Verification

Approval code paths have been reordered so that authorization does not survive a failed handshake.

ISSUE#5

Leaked Connect Ticket `auth_token` Allows Persistent Session Hijacking

Medium Partially Fixed

Description

The Connect ticket creation flow exposes authentication material before the wallet user approves the connection. This authentication material belongs to the approved Connect session lifecycle, but it is made available before the approval boundary.

Verification

The impact window has been reduced and connection handling made thread-safe.

ISSUE#6

Connect Ticket Can Be Approved Multiple Times Without Invalidating Prior Auth Tokens

Medium Fixed

Description

Repeated approvals of a ticket did not invalidate previously issued tokens for that ticket.

Verification

Fix makes ticket approval single-use and rotates or invalidates tokens on authorization changes.

ISSUE#7

`GetTicketByAuthToken` Accepts A bcrypt Hash Of An Empty Session ID

Medium Fixed

Description

An API-style bearer token could be constructed for tickets whose session identifier was empty.

Verification

Fix removes the affected token-processing path in favor of standard, state-validated authorization.

ISSUE#8

`GetTicket` Exposes Session IDs Needed To Forge API Connect Auth Tokens

Medium Fixed

Description

An API call leaked sensitive information necessary for the construction of server sdk authorization tokens.

Verification

The leaked information has been removed and server sdk authorization was migrated to JWT.

ISSUE#9

`getTicketIDForAPI()` Reuses The JWT Signing Key For Ticket ID Derivation

Low Fixed

Description

A ticket-ID derivation scheme for the server sdk reused the JWT signing key across domains with unrelated purposes.

Verification

A separate, domain-specific, rotatable secret has been implemented for server sdk derivation schemes.

Section 5

dApp Fee And Server SDK Execution State

Findings Summary

Issues	Severity	Status
ISSUE #1 Connect Request ID Reuse Can Bypass Fresh dApp Fee Tracking	Medium	Fixed
ISSUE #2 Execute-And-Wait Marks dApp Fee Paid From Unverified Transfer Metadata	Medium	Partially Fixed
ISSUE #3 Undecorated Connect Failures Can Clear Older Pending dApp Fee Rows By Reusing Request IDs	Medium	Partially Fixed
ISSUE #4 Wait-Mode Execution Errors Mark Fee Rows Failed Before Finality Is Known	Medium	Acknowledged
ISSUE #5 Server SDK Pending-Gas Prepare Refreshes Fee-Row Visibility And Bypasses The Unpaid-Gas Gate	Medium	Fixed
ISSUE #6 Server SDK Transaction Prepare Failures Can Clear Refreshed Unpaid Gas Via Caller-Supplied Command IDs	Medium	Partially Fixed
ISSUE #7 Server SDK Fee-Choice Cache Expiry Downgrades Delayed Transaction Fees To The Flat Fee	Medium	Acknowledged
ISSUE #8 Connect And Server SDK Execute Paths Accept Signed Prepared Transactions Without Server-Side Prepare Binding	Medium	Acknowledged

ISSUE#1

Connect Request ID Reuse Can Bypass Fresh dApp Fee Tracking

Medium Fixed

Description

Reuse of a request identifier within a short window could cause a new fee-bearing transaction to inherit a prior fee-tracking record instead of creating a fresh one.

Verification

Fix binds fee tracking to server-side state for the exact request and rejects stale or mismatched records.

ISSUE#2

Execute-And-Wait Marks dApp Fee Paid From Unverified Transfer Metadata

Medium Partially Fixed

Description

The dApp fee settlement flow can treat transaction metadata as sufficient evidence that a required fee was paid. Fee state should only advance after the committed transaction is verified against the expected fee payment terms.

Verification

Recognized fee payloads are now rebuilt server-side before preparation.

ISSUE#3

Undecorated Connect Failures Can Clear Older Pending dApp Fee Rows By Reusing Request IDs

Medium Partially Fixed

Description

A failure path could select and terminalize a fee record based on a client-supplied request identifier rather than a proven server-side record.

Verification

Validation on the primary path has been tightened.

ISSUE#4

Wait-Mode Execution Errors Mark Fee Rows Failed Before Finality Is Known

Medium Acknowledged

Description

An ambiguous execution error could terminalize a fee record before ledger finality was known.

Verification

Acknowledged.

ISSUE#5

Server SDK Pending-Gas Prepare Refreshes Fee-Row Visibility And Bypasses The Unpaid-Gas Gate

Medium Fixed

Description

A preparation step could affect the timestamp used by the unpaid-gas enforcement gate, temporarily affecting when unpaid gas was treated as blocking.

Verification

Fix separates enforcement from mutable bookkeeping so preparation no longer affects the gate.

ISSUE#6

Server SDK Transaction Prepare Failures Can Clear Refreshed Unpaid Gas Via Caller-Supplied Command IDs

Medium Partially Fixed

Description

A preparation failure could terminalize an unpaid-gas record based on caller-supplied identifiers.

Verification

The enforcement gate now uses durable state and validates prepared records. A narrower residual path remains and is scoped for the planned native fee architecture.

ISSUE#7

Server SDK Fee-Choice Cache Expiry Downgrades Delayed Transaction Fees To The Flat Fee

Medium Acknowledged

Description

Fee classification relied on process-local cache state, so a delayed execution could fall back to a flat fee rather than the correct fee.

Verification

Acknowledged.

ISSUE#8

Connect And Server SDK Execute Paths Accept Signed Prepared Transactions Without Server-Side Prepare Binding

Medium Acknowledged

Description

Execution did not bind a signed prepared transaction to a single-use server-side prepare record, which affects backend accounting and request-state integrity rather than ledger signature verification.

Verification

Acknowledged.

Section 6
Transaction Review And Signing Integrity

Findings Summary

Issues	Severity	Status
ISSUE #1 WalletConnect DAML Review Omits Material Commands And Ledger Parties	High	Partially Fixed
ISSUE #2 Unbound OneSwap Responses Can Misbind Wallet Identity And Redirect Signed Deposits	High	Fixed
ISSUE #3 Bulk Transfer Signs Backend-Parsed CSV Without A Canonical Review Summary	Medium	Acknowledged
ISSUE #4 Prepared Submission Caches Are Reusable And Weakly Bound To Routes And Operations	Medium	Acknowledged
ISSUE #5 Registry Burn Commands Are Displayed As Generic USDC Withdrawals	Medium	Fixed
ISSUE #6 Native Wallet Opaque-Hash Approval Flows Lack Backend-Bound Review Summaries	Medium	Partially Fixed

ISSUE#1

WalletConnect DAML Review Omits Material Commands And Ledger Parties

High Partially Fixed

Description

The transaction review surface did not always render every material field of a transaction before signing, so the primary approval text could understate what was being authorized.

Verification

Multi-command and malformed requests are now rejected, the acting party is forced to the authenticated wallet, and the full payload is available for inspection.

ISSUE#2

Unbound OneSwap Responses Can Misbind Wallet Identity And Redirect Signed Deposits

High Fixed

Description

Responses from an external swap provider were treated as authoritative without validating them against the authenticated request, which could influence a signed deposit.

Verification

Fix validates provider responses against the authenticated request before the user signs.

ISSUE#3

Bulk Transfer Signs Backend-Parsed CSV Without A Canonical Review Summary

Medium Acknowledged

Description

The signed object was derived from backend parsing of an uploaded file, while the review preview was produced separately, so the two could diverge.

Verification

Acknowledged. The client treats the uploaded file as user-supplied input and constrains it by rejecting invalid, duplicate, unapproved, and oversized entries. No further change is planned under the current product design.

ISSUE#4

Prepared Submission Caches Are Reusable And Weakly Bound To Routes And Operations

Medium Acknowledged

Description

Some wallet prepare flows cache backend-prepared submissions in a way that is not sufficiently bound to the approved operation, authenticated party, or intended execution lifecycle. This can allow prepared submissions to remain reusable outside the user's intended approval context.

Verification

Acknowledged.

ISSUE#5

Registry Burn Commands Are Displayed As Generic USDC Withdrawals

Medium Fixed

Description

Certain burn operations could be presented in the review surface under a generic withdrawal label rather than as the specific operation

Verification

Fix replaces the broad classification with strict decoders and dedicated review summaries.

ISSUE#6

Native Wallet Opaque-Hash Approval Flows Lack Backend-Bound Review Summaries

Medium Partially Fixed

Description

In some native flows the user reviewed locally displayed context while signing an opaque prepared hash, so the review was not always bound to a backend-authoritative summary of the prepared transaction.

Verification

The review surface has been improved.

Section 7
Vault Security

Findings Summary

Issues	Severity	Status
ISSUE #1 Unvalidated `redirectUrl` In `ConnectPage` Vulnerable to Cross-Site Scripting (XSS)	Critical	Fixed
ISSUE #2 `PASS_HASH.rawKey` Can Decrypt The Encrypted Private Key	High	Fixed
ISSUE #3 Passkey Wallet Encryption Uses the Stored Passkey Identifier As Key Material	High	Fixed
ISSUE #4 Vault accessible from UI origin	High	Acknowledged
ISSUE #5 PBKDF2 Work Factor Is Too Low For Private-Key Encryption	Medium	Fixed
ISSUE #6 CSP Allows `unsafe-inline` In `script-src`	Medium	Fixed

ISSUE#1

Unvalidated `redirectUrl` In `ConnectPage` Vulnerable to Cross-Site Scripting (XSS)

Critical Fixed

Description

A redirect parameter was not validated before navigation, which could allow script execution in the wallet's origin. Working payloads were additionally blocked in production by a web application firewall at the time of testing.

Verification

Fix strictly validates the redirect target and rejects dangerous schemes.

ISSUE#2

`PASS_HASH.rawKey` Can Decrypt The Encrypted Private Key

High Fixed

Description

Key material capable of decrypting the private key was stored alongside the encrypted key, so local access to browser storage could bypass the password.

Verification

Fix stops storing derived encryption key material and derives it from the password at unlock time.

ISSUE#3

Passkey Wallet Encryption Uses the Stored Passkey Identifier As Key Material

High Fixed

Description

Passkey-based encryption derived key material from a stored identifier rather than a hardware-backed secret, so the passkey ceremony was not strictly required to derive the key.

Verification

Fix moves passkey key derivation to a hardware-bound mechanism.

ISSUE#4

Vault accessible from UI origin

High Acknowledged

Description

The vault is stored under the same origin as the wallet UI, which reduces defense-in-depth against frontend-level compromise such as cross-site scripting, supply-chain issues, or malicious browser extensions. The specific frontend and key-material issues that would combine with this weakness have been fixed (see findings 1, 2, and 3 in this section).

Verification

Acknowledged.

ISSUE#5

PBKDF2 Work Factor Is Too Low For Private-Key Encryption

Medium Fixed

Description

The password-based key-derivation work factor was lower than current guidance for private-key encryption.

Verification

Fix raises the work factor and introduces versioned key-derivation parameters.

ISSUE#6

CSP Allows `unsafe-inline` In `script-src`

Medium Fixed

Description

The Content Security Policy permitted inline scripts, reducing the protection the policy is intended to provide if an injection path reached the origin.

Verification

Fix removes the unsafe-inline allowance from the script policy.

Section 8

Account State, Accounting, And Administrative Controls

Findings Summary

Issues	Severity	Status
ISSUE #1 Authenticated Users Can Read Other Wallets' Transfer History Details By Numeric ID	High	Fixed
ISSUE #2 Duplicate User Party IDs Can Collapse Wallet Account Boundaries	Medium	Acknowledged
ISSUE #3 Reward And Traffic Workers Can Duplicate Ledger Payouts And Marker Posts	Medium	Partially Fixed
ISSUE #4 Admin Reward Grants Can Queue Unbounded Backend-Funded Reward Payouts Without Budget Or Idempotency Controls	Medium	Fixed
ISSUE #5 Admin Runtime Config Mutations Can Apply Unsafe Wallet Policy Values Without Per-Key Validation Or Audit	Medium	Acknowledged
ISSUE #6 USDC Bridge Deposit Uses Floating-Point Amount Conversion	Low	Fixed
ISSUE #7 Transfer Preapproval Ignores Submitted Instrument IDs And Grants Blanket Admin Scope	Low	Acknowledged
ISSUE #8 Loop's Preapproval State Can Diverge From On-Chain Utility Preapprovals	Low	Fixed

ISSUE#1

Authenticated Users Can Read Other Wallets' Transfer History Details By Numeric ID

High Fixed

Description

A transfer-details lookup did not require the requesting account to be a party to the transfer, allowing an authenticated user to read transfer metadata for transfers they did not own. It did not allow moving funds or executing transactions.

Verification

Fix scopes the lookup to the authenticated party.

ISSUE#2

Duplicate User Party IDs Can Collapse Wallet Account Boundaries

Medium Acknowledged

Description

A missing uniqueness invariant on the identity value used as the account boundary could, under certain conditions, allow two accounts to resolve to the same wallet party.

Verification

Acknowledged. The client has accepted this as a low-likelihood risk. Their position is that wallet party IDs are assigned through the Canton/validator onboarding flow rather than direct user input, use a randomly generated party hint, and include the signing-key fingerprint.

ISSUE#3

Reward And Traffic Workers Can Duplicate Ledger Payouts And Marker Posts

Medium Partially Fixed

Description

Backend workers submitted ledger side effects before durably marking the work as processed, so a retry or crash could repeat a backend-controlled side effect. This is not a user-to-user theft path.

Verification

Partially fixed. The reward-airdrop portion has been removed from the active code path, which closes the duplicate Backend-funded reward payout surface. The traffic marker worker remains and intentionally uses an at-least-once posting model, where marker batches may be over-posted rather than under-posted. The client has accepted this behavior and plans to remove the marker-posting path as part of the upcoming traffic-based reward replacement.

ISSUE#4

Admin Reward Grants Can Queue Unbounded Backend-Funded Reward Payouts Without Budget Or Idempotency Controls

Medium Fixed

Description

An admin-only endpoint could queue backend-funded reward grants without budget caps, idempotency, or an immutable audit requirement. The path required admin authorization and was not available to ordinary users.

Verification

Fix removes the admin reward-grant endpoint and the reward payout worker from the active code path.

ISSUE#5

Admin Runtime Config Mutations Can Apply Unsafe Wallet Policy Values Without Per-Key Validation Or Audit

Medium Acknowledged

Description

Admin runtime configuration accepted values without typed per-key validation or an immutable audit trail. The path required admin authorization and was not available to ordinary users.

Verification

Acknowledged. The client has accepted this admin-only runtime configuration risk, relying on admin-role access plus simple Frontend and Backend validation.

ISSUE#6

USDC Bridge Deposit Uses Floating-Point Amount Conversion

Low Fixed

Description

A deposit amount was converted using floating-point arithmetic, which could silently truncate or round the amount shown to the user before signing.

Verification

Fix replaces the conversion with exact integer decimal parsing and displays the normalized amount.

ISSUE#7

Transfer Preapproval Ignores Submitted Instrument IDs And Grants Blanket Admin Scope

Low Acknowledged

Description

The preapproval prepare API accepts an itemized list of instruments, but those instruments are not preserved during transaction preparation. Under the Utility Registry model, this can result in preapproval for all instruments under the given admin.

Verification

Acknowledged. The client notes that this behavior is by design and is as intended.

ISSUE#8

Loop's Preapproval State Can Diverge From On-Chain Utility Preapprovals

Low Fixed

Description

A preapproval created through the generic transaction path did not update the wallet's own preapproval bookkeeping, so backend state could diverge from on-chain state.

Verification

Fix keeps backend preapproval state in sync with on-chain preapprovals.

Section 9
General

Findings Summary

Issues	Severity	Status
ISSUE #1 `submissionID` Reuses `commandID` Instead Of A Fresh Per-Attempt ID	Low	Acknowledged

ISSUE#1

`submissionID` Reuses `commandID` Instead Of A Fresh Per-Attempt ID

Low Acknowledged

Description

A per-attempt submission identifier was derived from the logical operation identifier rather than generated fresh for each attempt, which affects observability of retries.

Verification

Acknowledged. The client confirmed that `submissionID` is intended for tracking submission attempts and does not represent an urgent security concern.

Section 10
Appendix A

Severity Rating Definitions

Verified by Humans utilizes the Immunefi Vulnerability Severity Classification System - v2.3.

Severity	Definition

Critical	• Execute arbitrary system commands
	• Retrieve sensitive data/files from a running server, such as:
	  ◦ /etc/shadow
	  ◦ database passwords
	  ◦ blockchain keys (this does not include non-sensitive environment variables, open source code, or usernames)
	• Taking down the application/website
	• Taking down the NFT URI
	• Taking state-modifying authenticated actions (with or without blockchain state interaction) on behalf of other users without any interaction by that user, such as:
	  ◦ Changing registration information
	  ◦ Commenting
	  ◦ Voting
	  ◦ Making trades
	  ◦ Withdrawals, etc.
	• Changing the NFT metadata
	• Subdomain takeover with already-connected wallet interaction
	• Direct theft of user funds Malicious interactions with an already-connected wallet, such as:
	  ◦ Modifying transaction arguments or parameters
	  ◦ Substituting contract addresses
	  ◦ Submitting malicious transactions
	• Direct theft of user NFTs
	• Injection of malicious HTML or XSS through NFT metadata

High	• Injecting/modifying the static content on the target application without JavaScript (persistent), such as:
	  ◦ HTML injection without JavaScript
	  ◦ Replacing existing text with arbitrary text
	  ◦ Arbitrary file uploads, etc.
	• Changing sensitive details of other users (including modifying browser local storage) without already-connected wallet interaction and with up to one click of user interaction, such as:
	  ◦ Email or password of the victim, etc.
	  ◦ Improperly disclosing confidential user information, such as:
	  ▪ Email address
	  ▪ Phone number
	  ▪ Physical address, etc.
	• Subdomain takeover without already-connected wallet interaction

Severity	Definition

Medium	• Changing non-sensitive details of other users (including modifying browser local storage) without already-connected wallet interaction and with up to one click of user interaction, such as:
	  ◦ Changing the first/last name of user
	  ◦ Enabling/disabling notifications
	• Injecting/modifying the static content on the target application without JavaScript (reflected), such as:
	  ◦ Reflected HTML injection
	  ◦ Loading external site data
	  ◦ Redirecting users to malicious websites (open redirect)

Low	• Changing details of other users (including modifying browser local storage) without already-connected wallet interaction and with significant user interaction, such as:
	  ◦ Iframing leading to modifying the backend/browser state (must demonstrate impact with PoC)
	  ◦ Taking over broken or expired outgoing links, such as:
	  ▪ Social media handles, etc.
	• Temporarily disabling user to access target site, such as:
	  ◦ Locking up the victim from login
	  ◦ Cookie bombing, etc.

None	• We make note of issues of no severity that reflect best practice recommendations or opportunities for optimization, including, but not limited to, gas optimization, the divergence from standard coding practices, code readability issues, the incorrect use of dependencies, insufficient test coverage, or the absence of documentation or code comments.

Section 11
Appendix B

Verified by Humans Disclaimer

Verified by Humans performs audits and related services under specific, agreed scopes of work with each customer. Our reports reflect what we found in the systems we reviewed, based on the information available to us at the time. We work hard to be thorough and accurate, but a report from us is not a guarantee that a project is secure, error-free, or risk-free. Cryptographic technology is still emerging, and every evaluation of it — including ours — carries inherent uncertainty. Our reports are not financial, investment, legal, tax, or regulatory advice, and they are not an endorsement of any technology or project. No third party should rely on them when making investment decisions or treat them as a guarantee of security. Where our reports link to external websites or reference third-party information, we provide those links as a convenience. We do not control, endorse, or take responsibility for the content or privacy practices of any external site. Users should verify third-party information independently before relying on it. The contents of our reports — including our methodologies, analysis, and conclusions — are the proprietary intellectual property of Verified by Humans, provided exclusively to our customer for the use specified in the engagement. Unauthorized disclosure, reproduction, or distribution is prohibited unless we have authorized it in writing. We do not assume any obligation to update a report after publication, and making these analyses available creates no duty to any third party.