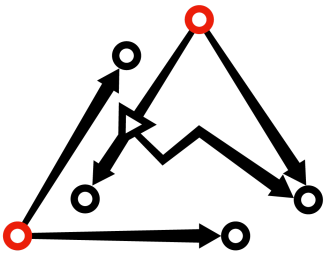

Least Authority

PRIVACY MATTERS

Permuto

Security Audit Report

Chia Network

Final Audit Report: 31 July 2025

Table of Contents

Overview

Background

Chia Network has requested that Least Authority perform security audits of Permuto. Permuto consists of certificates issued by the Trust, which is formed solely to hold a single class of stock in a specified publicly traded company, with the first being shares of the common stock of Microsoft Corporation (NASDAQ: MSFT). This review consisted of three audits on the *Permuto App and Chia Signer iOS Mobile App*, the *Enterprise Wallet and Chia Wallet SDK*, and the *CATS and Admin Tool*. The findings from all three audits are presented in this report.

Project Dates

Audit 1: Permuto App and Chia Signer iOS Mobile App

- **March 6, 2025 - April 16, 2025**: Initial Code Review *(Completed)*
- **April 18, 2025**: Delivery of Initial Audit Report *(Completed)*
- **July 31, 2025**: Verification Review *(Completed)*
- **July 31, 2025:** Delivery of Final Audit Report *(Completed)*

Audit 2: Enterprise Wallet and Chia Wallet SDK

- **March 6, 2025 - April 16, 2025**: Initial Code Review *(Completed)*
- **April 18, 2025**: Delivery of Initial Audit Report *(Completed)*
- **July 31, 2025**: Verification Review *(Completed)*
- **July 31, 2025:** Delivery of Final Audit Report *(Completed)*

Audit 3: CATs and Admin Tool

- **March 31, 2025 - April 16, 2025**: Initial Code Review *(Completed)*
- **April 18, 2025**: Delivery of Initial Audit Report *(Completed)*
- **July 31, 2025**: Verification Review *(Completed)*
- **July 31, 2025:** Delivery of Final Audit Report *(Completed)*

Review Team

- Nikos Iliakis, Security Researcher and Engineer
- Anna Kaplan, Cryptography Researcher and Engineer
- Paul Lorenc, Security Researcher and Engineer
- Michael Rogers, Security Researcher and Engineer
- Mirco Richter, Cryptography Researcher and Engineer
- Will Sklenars, Security Researcher and Engineer
- Dominic Tarr, Security Researcher and Engineer
- Burak Atasoy, Project Manager
- Jessy Bissal, Technical Editor

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of Permuto followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repositories are considered in scope for the review:

- See Appendix A.

Specifically, we examined the following Git revisions for our initial review:

- Audit 1: Permuto App and Chia Signer iOS Mobile App:
  - 9ce50d40b9dc810e82681252e522d6ac62ed46f8
  - 42dfe52c94a13730d2147bbe2c93a64b6fb9a9be

- Audit 2: Enterprise Wallet and Chia Wallet SDK
  - 33a8606ab94a8bd7c896a8cdfc96646d84252be
  - bbd411fe7e3ce773067ff6cc4db7b0cc315db01c

- Audit 3: CATs and Admin Tool
  - 7114078826d82e3405298a6ca771e861932e2a40

For the verification, we examined the following Git revisions:

- Audit 1: Permuto App and Chia Signer iOS Mobile App:
  - 9f91cbf42770c5f712d7562e227cbb4a5411b103
  - 580594f3c60ae2a9a55941216fb649cc8530c87a

- Audit 2: Enterprise Wallet and Chia Wallet SDK
  - 78305d9a63ded9589c2f48f8575c4eaef528808a

For the review, these repositories were cloned for use during the audit and for reference in this report:

- Audit 1: Permuto App and Chia Signer iOS Mobile App:
  - https://github.com/LeastAuthority/chia-permuto-app
  - https://github.com/LeastAuthority/chia-signer-ios

- Audit 2: Enterprise Wallet and Chia Wallet SDK
  - https://github.com/LeastAuthority/chia-ent-wallet
  - https://github.com/LeastAuthority/chia-wallet-sdk

- Audit 3: CATs and Admin Tool
  - https://github.com/LeastAuthority/chia-CAT-admin-tool

All file references in this document use Unix-style paths relative to the project's root directory.

In addition, any dependency and third-party code, unless specifically mentioned as in scope, were considered out of scope for this review.

Supporting Documentation

The following documentation was available to the review team:

- SEC Filing: https://www.streetinsider.com/SEC+Filings/Form+S-1+Permuto+Capital+MSFT/24208915.html
- Businesswire announcement: https://www.businesswire.com/news/home/20250114592626/en/Newly-Launched-Permuto-Capital-Announces-Filing-of-Registration-Statement-for-New-Type-of-Equity-Product
- Website: https://www.chia.net
- Clippy_Audit Scope.xlsx *(shared with Least Authority via email on 29 January 2025)*

- Clippy Requirements version 2 - LATEST.xlsx *(shared with Least Authority via email on 30 January 2025)*

In addition, this audit report references the following documents:

- Block Validation | Chia Documentation: https://docs.chia.net/block-validation
- Conditions | Chialisp: https://chialisp.com/conditions/#assert-before-seconds-absolute
- Principles of Blockchains | Transaction Ordering and Fairness: https://courses.grainger.illinois.edu/ece598pv/sp2021/lectureslides2021/ECE_598_PV_course_notes13_v2.pdf
- SET | Docs: https://redis.io/docs/latest/commands/set
- `cuid2` library: https://github.com/paralleldrive/cuid2
- Blog post, "There's `Math.random()`, and then there's `Math.random()`": https://v8.dev/blog/math-random

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the implementation;
- Vulnerabilities within each component and whether the interaction between the components is secure;
- Whether requests are passed correctly to the network core;
- Key management, including secure private key storage and management of encryption and signing keys;
- Denial of Service (DoS) and other security exploits that would impact the intended use or disrupt the execution;
- Protection against malicious attacks and other ways to exploit;
- Inappropriate permissions and excess authority;
- Data privacy, data leaking, and information integrity; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

Permuto utilizes the Chia blockchain to represent stock ownership via Chia Asset Tokens (CATs), categorized into Asset CATs and Dividend CATs. Users interact with the system through the Permuto App, a front-end administrative interface that facilitates stock deposits, certificate issuance, and redemption. CATs are controlled by a singleton coin held in a vault primitive, through which transactions such as trading, spending, and token movements can be authorized. Users manage vault interactions through the Chia Enterprise Wallet, which leverages the Chia Wallet SDK to manage the lower-level interactions with the blockchain. "Puzzles" are used to define how a coin can be spent on the blockchain. The Enterprise Wallet uses the Meta Inner Puzzle Specification (MIPS) custody layer to divide custody among a number of members and restrictions. The MIPS layer is used to implement the primitive functionality of a "vault." To ensure compliance and security, the system employs a revocable CAT mechanism, which is managed

via an inner puzzle and enables Permuto to revoke and reissue tokens during stock splits or other regulatory events.

Our team conducted security audits of Permuto's main components, comprising the Permuto App and Chia Signer iOS Mobile App, the Enterprise Wallet and Vault Implementation, and the CAT Admin Tool, in three separate reviews. Below, we detail our findings from each of these reviews.

Permuto App and Chia iOS Signer

During our review of the Permuto App and Chia Signer iOS, our team analyzed whether all signatures are produced with correct cryptographic primitives and appropriate parameter checks, and did not uncover any immediately exploitable vulnerabilities. However, we identified the use of a vulnerable version of `Next.js`, which has a known security issue (CVE-2025-29927) that could allow attackers to bypass authentication. This vulnerability may enable unauthorized access to exposed APIs, allowing malicious actions such as transferring funds or assets. We recommend promptly upgrading `Next.js` to the patched version (`15.2.3`) to mitigate this associated risk (Issue D). In addition, we identified an opportunity to strengthen user authentication requirements and recommend introducing biometric or device passcode verification when disabling key protection features, in order to prevent unintended or unauthorized configuration changes (Suggestion 3).

Enterprise Wallet and Vault Implementation

In our evaluation of the API subcomponent, we discovered two issues related to key-based authentication mechanisms. We identified a race condition in the Chia Enterprise Wallet's key-based authentication, where concurrent processing of API requests signed with identical nonces can result in nonce reuse, thereby undermining replay protection (Issue B).

Our team also identified a potential replay attack in the Chia Enterprise Wallet's key-based authentication, caused by a mismatch between nonce retention (five minutes) and timestamp validity (ten minutes), which may allow attackers to replay requests after nonce expiry if timestamps are set slightly ahead of the server's clock (Issue C).

When assessing high-level attack vectors, we also identified a timestamp manipulation vulnerability, whereby Chia farmers can opportunistically exploit the five-minute future-timestamp allowance to simultaneously capture dividend payouts and pre-dividend token prices, disadvantaging other traders (Issue A).

Our team also performed a manual review of the application’s front-end, focusing on user interactions and input processing. We did not identify any issues related to the handling or validation of user interface interactions.

Furthermore, we analyzed potential attack scenarios targeting the vault recovery mechanism, particularly those involving the loss of a user's mobile device containing spend keys and access credentials such as email. We identified areas of improvement that would enhance the overall security of the system. Specifically, we recommend improving the watchtower notification system by incorporating multiple independent communication channels. Additionally, enabling passphrase-based encryption for iCloud-stored keys would further mitigate risks associated with unauthorized remote vault recovery attempts (Suggestion 4).

Finally, we reviewed the Rust implementation of the MIPS driver, spend logic, vault binding wrapper, puzzle types, vault singletons, and Merkle tree structures. We did not identify any issues in these areas of investigation.

CAT Admin Tool and Vault Puzzles

Our team reviewed the CAT Admin Tool, a Python-based command-line interface (CLI) designed to facilitate the minting and management of CATs, including Credential Restricted CATs (CR-CATs). The minting functionality accurately implements the intended token issuance logic. We also examined the "Secure the Bag" feature for bulk token distributions via CSV input and the "Unwind the Bag" functionality for controlled token removals, and found both to be correctly implemented.

Additionally, we examined the vault puzzles, which extend the basic spending logic by integrating time locks, multi-signature authorization, and conditional spending mechanisms. We checked whether the puzzles correctly enforce strict and customizable spending conditions, and did not identify any issues.

Dependencies

Our audit of the `package.json` files in the Permuto App and the Chia Ent Wallet repositories revealed several vulnerable instances. We recommend improving dependency management practices to mitigate these risks (Suggestion 5).

Code Quality

We performed a manual review of the repositories in scope and found the code to be well-organized and aligned with best practices for TypeScript, Rust, and Python.

Tests

The Permuto server component includes an established testing suite; however, current test coverage is insufficient. In contrast, the Permuto client lacks a testing framework entirely. To address these gaps, we recommend implementing a testing framework using Jest's `Next.js` integration, which would enhance test coverage and support long-term reliability. The Enterprise Wallet repository, on the other hand, already contains a testing suite that covers key API functionalities. Similarly, the CAT Admin Tool includes tests that cover both key functionalities and critical features.

Documentation and Code Comments

The project documentation provided by the Chia Network team, along with code comments, sufficiently describes most intended functionalities. In particular, documentation for the Enterprise Wallet adequately facilitated the setup of the development environment; however, future enhancements should include comprehensive installation instructions specifically for Linux-based systems. The CAT Admin Tool repository also contains clear, descriptive comments covering critical logic and key functions. Similarly, the documentation and code comments for the Vault puzzles were found to be detailed and effective in conveying their intended purpose and behavior.

Scope

The scope of this review was sufficient and included all security-critical components.

Specific Issues & Suggestions

We list the issues and suggestions found during the review, in the order we reported them. In most cases, remediation of an issue is preferable, but mitigation is suggested as another option for cases where a trade-off could be required.

ISSUE / SUGGESTION	STATUS

Issue A: Timestamp Manipulation for Dividend Arbitrage

Synopsis

When a Chia farmer creates a block, the farmer has some flexibility in choosing the block’s timestamp. Although a farmer can use this flexibility to accept an offer to buy Permuto dividend tokens, where the offer is intended to expire before an upcoming dividend is issued, the block’s timestamp can be set to show that the offer was accepted after the dividend was issued. The farmer can thus receive the dividend while benefiting from the expected drop in the token’s price when the dividend is issued.

Impact

Medium.

The attacker can benefit financially from the attack at the expense of the party buying the attacker’s tokens.

Feasibility

High.

The attack is straightforward to execute if the preconditions are met. It can be performed opportunistically by anyone already farming Chia and holding Permuto dividend tokens, which are activities that are likely to offset their own costs, regardless of whether an opportunity to execute the attack arises.

Severity

Medium.

Preconditions

The attack is opportunistic. In order to have an opportunity to carry out the attack, the attacker must be an active Chia farmer and must own some Permuto dividend tokens for a stock that is due to receive a

dividend payment in the future. Crucially, the attacker must have the opportunity to create a block within five minutes of the time when the dividend will be issued. At the time the attacker has an opportunity to create a block, there must be at least one active offer to purchase the tokens.

Finally, there must be a reasonable expectation that the market price of the dividend token will fall when the dividend is issued. This is the typical market behavior for dividend-paying stock, as the price before the dividend is issued includes the expected amount of the dividend payment, whereas the price after the dividend is issued does not. The same market behavior is anticipated to apply to Permuto dividend tokens. Without this price difference, the attack could still be carried out but the attacker would not expect to profit from it.

Technical Details

A farmer who creates a Chia block has some flexibility in choosing the block's timestamp. The timestamp must be greater than that of the previous block, and it must not be more than five minutes in the future, based on the clocks of the nodes validating the block. Assuming most nodes have accurate clocks, a dishonest farmer can choose a timestamp up to five minutes ahead, and the network will still validate the block.

When a dividend is issued for a stock represented by a Permuto dividend token, Permuto's token tracker identifies the owner of each token at the time when the dividend was issued so that the dividend can be paid to that owner. The times at which changes of ownership occur are determined by the timestamps of the blocks containing the corresponding spend bundles. Thus, flexibility in choosing a block's timestamp can be turned into flexibility in determining the ownership of a token at a specific time, if there is a change in ownership close to that time.

As mentioned above, it is typical for the price of a dividend-paying stock to fall when a dividend is issued, and this behavior is expected to apply to Permuto dividend tokens also. Accordingly, traders offering to buy a dividend token at the higher market price before the dividend is issued are likely to cancel their offers shortly before the dividend is issued and place new offers at the lower market price after the dividend is issued. An attacker can profit if they are able to accept an offer at the higher market price before it is canceled, while simultaneously being recorded as the owner of the dividend token at the time the dividend is issued.

Consider the case where the dividend payment is scheduled for midnight, there is an offer to buy the tokens valid until `23:59`, and the attacker has an opportunity to create a block at `23:58`. The attacker sets the timestamp of the new block containing the attacker's acceptance of the offer to `00:01`, so the token tracker considers the attacker to have owned the tokens at midnight.

There are two ways that a trader who has offered to buy tokens may cancel that offer. The trader may issue a cancellation on-chain by spending the coin used to make the offer, or include an expiry time in the offer by outputting a condition such as `ASSERT_BEFORE_SECONDS_ABSOLUTE`. In either case, if the offer has not yet been canceled when the attacker has the opportunity to create a block, the attacker can accept the offer before it is canceled.

In the case of on-chain cancellation, the spend bundle containing the cancellation is either in the mempool or it has not yet been published. If it is in the mempool, then the attacker simply omits it from their block. Either way, once the attacker's block has been published, the cancellation will no longer be valid for inclusion in any subsequent block, as the offer will have already been accepted.

In the case of an expiry time using a condition such as `ASSERT_BEFORE_SECONDS_ABSOLUTE`, the attack exploits the fact that such conditions refer to the timestamp of the previous block, rather than the block containing the spend bundle that outputs the condition. In the example above, when the attacker creates a block at `23:58`, the offer's condition (that the expiry time of `23:59` has not yet been reached)

remains true, as evaluated against the timestamp of the previous block. This would not have been the case if it had been based on the timestamp of the attacker's block, which contains the acceptance of the offer.

A side effect of the attack is that any blocks following the attacker's block must have timestamps greater than the timestamp of the attacker's block. As a result, the timestamps of all subsequent blocks will be inaccurate until the timestamp of the attacker's block has passed. (In the example, this occurs at `00:01`, so all blocks created between `23:58` and `00:01` have inaccurate timestamps). These subsequent blocks cannot be used to carry out further instances of the attack against offers with expiry conditions, as any such conditions would be evaluated against the timestamp of the attacker's block or a subsequent block, resulting in the offers being treated as having expired.

The discussion so far has assumed the existence of a single attacker who was both a farmer and a token trader, but in principle, these roles could be separated. An off-chain market could arise in which farmers sell the opportunity to include a spend bundle in a block with an inaccurate timestamp. This would be similar to the markets for timestamp manipulation proposed for other blockchains, although the scope for such manipulation appears to be particularly broad on the Chia blockchain.

Mitigation

Traders can mitigate the attack by making any offers to buy dividend tokens expire at least five minutes before a dividend is scheduled to be issued. Consequently, this results in the loss of five minutes of trading during a period that is likely to be particularly significant for trading activity involving those tokens.

Remediation

We recommend reducing the permissible range for future timestamps, for example by decreasing the current five-minute allowance to a smaller value. This assumes that most farmers maintain accurate clocks and therefore will not legitimately produce blocks with significantly future-dated timestamps, and that most nodes likewise maintain accurate clocks and will not misclassify accurate timestamps as being too far in the future.

When a node receives a block with a timestamp that lies in the future relative to its local clock, it could delay validation and propagation until the timestamp becomes current. This remediation could be introduced incrementally, and once widely deployed, would prevent the propagation of blocks with future timestamps across the network.

While our team recommends the approach outlined above, we recognize that these remediations may be considered out of scope for Permuto, as they apply to the Chia network as a whole, whereas the attack is specific to dividend tokens.

Status

The Chia Network team had previously reduced the permissible range for future timestamps to 2 minutes, effective July 2, 2023. However, the documentation was only updated to reflect this change following its mention in the initial audit report.

We note that Chia offers provide users with conditions to prevent the described attack, provided the timing-based conditions are precisely aligned with the timestamp of dividend payment. However, if a user sets the offer to expire even a minute before the timestamp, an attacker could perform the attack. Therefore, Chia's conditions allow users to securely create offers but require users to set the correct timestamps.

Verification

Resolved.

Issue B: Race Condition in Key-Based Authentication

Location

`apps/api/src/utils/verifyAuthKey.ts#L93`

Synopsis

The Chia Enterprise Wallet allows API requests to be authenticated by signing them with a passkey or a key stored in the iOS secure element. To prevent replays, each signature includes a timestamp and a nonce. A race condition allows a nonce to be used twice if two requests using the same nonce are received concurrently.

Impact

Low.

Feasibility

Low.

Even if the preconditions for the issue are met, exploiting the race condition is probabilistic, as the attacker cannot control the order in which the API server handles concurrent requests.

Severity

Low

Preconditions

The attacker must be able to capture, delay, and replay requests signed by a client. This could be achieved either by compromising the security of the network connection between the client and the server or by compromising the client device itself.

It is not necessary for the attacker to compromise the client's passkeys or secure element. If the attacker were able to do so, the attack would be redundant, as they could simply sign a fresh request using the client's key rather than replaying a previously signed request.

Technical Details

The Chia Enterprise Wallet uses Redis to keep track of nonces that have already been used.

Separate Redis `get` and `set` operations are used to check whether a nonce has already been used and, if not, to record that it has been used. If two requests using the same nonce are processed concurrently, it is possible for both to complete their `get` operations before either performs its `set` operation. In that case, the code will fail to detect that the same nonce has been used twice.

Remediation

We recommend replacing the separate `get` and `set` operations with a single `set` operation using the NX flag. This operation atomically checks whether the nonce has been used before, returning an error if so or recording its use otherwise.

Status

The Chia Network team has resolved this issue by updating the implementation to use a single `set` operation with the NX flag, as shown in this commit.

Verification

Resolved.

Issue C: Potential Replay Attack Against Key-Based Authentication

Location

apps/api/src/utils/verifyAuthKey.ts#L88

apps/api/src/utils/verifyAuthKey.ts#L95

Synopsis

The Chia Enterprise Wallet allows API requests to be authenticated by signing them with a passkey or a key stored in the iOS secure element. To prevent replays, each signature includes a timestamp and a nonce. A timestamp is considered valid if it falls within five minutes of the server's clock in either direction. Used nonces are retained for five minutes to prevent reuse. As a result, if a request includes a timestamp slightly in the future relative to the server's clock, it may become replayable once its nonce has expired but its timestamp is still considered valid.

Impact

Low.

Feasibility

Medium.

The client's clock must be ahead of the server's clock for the attack to succeed. A discrepancy of as little as one second may be sufficient. If the attacker has compromised the client device, they may be able to manipulate the client's clock. Otherwise, the attack relies on timing conditions beyond the attacker's control.

Severity

Low.

Preconditions

The attacker must be able to capture, delay, and replay requests signed by a client. This could be done either by compromising the security of the network connection between the client and the server, or by compromising the client device.

It is not necessary for the attacker to compromise the client's passkeys or secure element. If the attacker were able to do so, the attack would be redundant, as they could simply sign a fresh request using the client's key rather than replaying a previously signed request.

Technical Details

A request's timestamp is considered valid if it falls within five minutes of the server's clock in either direction. Nonces are stored in Redis and expire after five minutes. This creates a vulnerability: the ten-minute timestamp validity window exceeds the five-minute nonce retention period, allowing a window for replay.

If a client's clock is ahead of the server's by even a small margin and it submits a signed request, the request's nonce may expire from Redis while the timestamp remains within the validity window. When the nonce expires, the timestamp will be just under five minutes in the past according to the server's clock,

having initially been slightly in the future. The request can then be replayed and accepted as valid, since the timestamp is still within bounds and no record exists of the nonce having been used.

Remediation

We recommend increasing the nonce expiry time to slightly exceed the length of the timestamp validity window (ten minutes).

Status

The Chia Network team has increased the nonce expiry time to 11 minutes, as indicated on line 97 of this commit.

Verification

Resolved.

Issue D: Next.js Authorization Bypass Vulnerability

Location

`chia-permuto-app/packages/client/package.json#L39`

Synopsis

A critical vulnerability in `Next.js` (CVE-2025-29927), which is used by the Permuto App, was publicly disclosed during the audit.

Impact

High.

Successful exploitation could allow attackers to bypass authentication mechanisms, enabling unauthorized control over critical API endpoints. This may lead to unauthorized transfers of funds or other assets managed by the application.

Feasibility

High.

Given the public disclosure of this vulnerability, attackers can readily identify and probe applications running vulnerable `Next.js` versions.

Severity

Critical.

Preconditions

For the vulnerability to be exploitable, the application must run a vulnerable version of `Next.js` and expose API endpoints that rely on middleware-based authentication mechanisms.

Technical Details

`Next.js` includes a nonstandard feature in the form of a custom `x-middleware-subrequest` header, which can instruct the middleware runner to skip specified middleware. While circumventing most middleware may have limited impact, bypassing authentication middleware in particular can allow attackers to gain unauthorized control over API endpoints that are publicly exposed by the application.

Mitigation

Users of the Permuto application have no available mitigations against this vulnerability.

Remediation

We recommend immediately updating `Next.js` to the patched version `15.2.3`, which addresses this vulnerability. The security fix has also been backported to earlier major versions to support rapid adoption. Given Permuto's current version (`15.1.7`), upgrading to version `15.2.3` is expected to require no additional compatibility adjustments.

Additionally, we strongly recommend auditing all other components or APIs that use `Next.js`, including those outside the original scope of this audit (such as web wallet APIs), to ensure comprehensive remediation.

Status

The Chia Network team has updated `Next.js` to `15.3.2`, which includes the security patch, as indicated in this commit.

Verification

Resolved.

Suggestions

Suggestion 1: Use a Strong Source of Randomness for Generating OAuth Refresh Tokens

Location

`api/src/services/oauth/server.ts#L51`

`packages/db-schema/src/utils/createId.ts#L4`

Synopsis

OAuth refresh tokens are generated by using the `Math.random` function as the source of randomness. This function is not designed for this purpose. Some implementations of this function may produce tokens that have low entropy, making it feasible for an attacker to guess the tokens by brute force.

The Chia Enterprise Wallet uses the `cuid2` library to generate unique identifiers for database records. This library is also used for generating OAuth refresh tokens, which clients use to authenticate to the Permuto API.

The `cuid2` library supports the use of a cryptographically secure source of randomness, but in the default configuration used by the Chia Enterprise Wallet, it relies on `Math.random`. Some implementations of `Math.random` have a small state space, making it possible to infer the internal state of the random number generator by observing its outputs and thereby predict future outputs. In the context of generating authentication tokens, this could allow an attacker to guess tokens issued to other users.

Mitigation

As the implementation of `Math.random` currently used by `Node.js` has a large state space, there is no immediate risk of exploitation.

We recommend migrating the code to use the `crypto.randomBytes` function instead of the `cuid2` library for generating OAuth refresh tokens. This will protect against any future changes to the implementation of `Math.random` that could compromise the security of tokens generated with it. The `cuid2` library can continue to be used for generating unique database identifiers, as these do not need to be resistant to brute-force guessing in the way authentication tokens do.

Status

The Chia Network team has migrated to the cryptographically secure `crypto.randomUUID` function.

Verification

Resolved.

Suggestion 2: Restrict Access to Metrics Endpoints

Location

`apps/api/src/metrics.ts`

`apps/blockchain-sync/src/metrics.ts`

`apps/metrics/src/metrics.ts`

Synopsis

Three of the servers in the Chia Enterprise Wallet repository (`api`, `blockchain-sync`, and `metrics` servers) have API endpoints that allow metrics to be fetched without authentication. For the `api` and `blockchain-sync` servers, the metrics include detailed information about the `Node.js` environment, such as memory and file descriptor limits, which could be useful to an attacker. For the `metrics` server, the metrics include aggregate information about Permuto's users and services.

Mitigation

We recommend that these API endpoints not be exposed to the Internet. Ideally, they should also require authentication and be accessible only to administrators.

Status

The Chia Network team has confirmed that this API endpoint is not exposed to the internet. However, our team did not directly verify this, as confirming it would require reviewing the deployment process, and server configuration, which falls outside the scope of the audit.

Verification

Resolved.

Suggestion 3: Require User Authentication to Disable the "Authorize Transactions with Biometrics or Device PIN" Toggle

Location

`ui/keygen/KeyGenConfigPanel.swift#L81`

Synopsis

The toggle determines whether a newly-generated `Secure Enclave` key will embed the `.userPresence` flag. If the flag is absent, the resulting key can sign silently, allowing any code within the

application's sandbox to execute `SecKeyCreateSignature` without invoking Face ID or a device passcode prompt.

Since disabling the toggle does not require user re-authentication:

- Users may disable the protection accidentally or without understanding its permanence.
- "UI-only" malware (such as tap injection or SwiftUI state manipulation) can silently downgrade the setting during key generation and create harvestable keys.

The vulnerability is confined to the key that is about to be created; existing keys remain unaffected. No other component currently compensates for this downgrade path.

Mitigation

We recommend requiring an immediate biometric or device passcode authentication when users attempt to disable this toggle. If authentication fails or is canceled, the toggle should remain enabled. Conversely, re-enabling the toggle (switching from OFF to ON) should remain straightforward and free from additional authentication prompts.

Status

The Chia Network team now requires user authentication to disable the "Authorize Transactions with Biometrics or Device PIN" toggle.

Verification

Resolved.

Suggestion 4: Improve Account Recovery Process

Synopsis

If an attacker gains unauthorized remote access to a vault user's iCloud BLS key and associated email account, they could initiate a vault recovery process while simultaneously preventing the user from receiving critical watchtower notifications. Although the initiation of recovery would be visible on-chain, the legitimate user would be unaware, leaving them unable to take timely action to secure their vault assets.

Mitigation

We recommend strengthening the watchtower node implementation by incorporating multiple independent notification channels, such as simultaneous notifications to both the user and a trusted third party (e.g., a family member or friend), thereby providing reliable alerts during vault recovery actions. Additionally, we suggest enabling vault users to encrypt their iCloud-stored BLS keys with a symmetric passphrase, thus increasing the difficulty for remote attackers to gain unauthorized access to the keys.

Status

The Chia Network team has added comments for their users suggesting the use of Apple's Advanced Data Protection system for securing their iCloud-stored keys, and has also implemented support for additional emails.

Verification

Resolved.

Suggestion 5: Update Vulnerable Dependencies

Location

`chia-ent-wallet/package.json`

`chia-permuto-app/packages/client/package.json`

`chia-permuto-app/packages/server/package.json`

Synopsis

Analyzing `package.json` for dependency versions using `npm audit` shows:

- `10` Vulnerabilities (7 moderate, 3 high) in the Enterprise Wallet repository;
- 2 Vulnerabilities (1 moderate, 1 critical) in the Permuto `client` and;
- 2 Vulnerabilities (2 moderate) in the Permuto `server`.

While the extent of their exploitability remains unclear, it is recommended to keep dependencies up to date in order to avoid importing vulnerable code.

Mitigation

We recommend adopting a process that emphasizes secure dependency usage to avoid introducing vulnerabilities to Permuto and to mitigate supply-chain attacks. This process should include:

- Manually reviewing and assessing currently used dependencies;
- Upgrading dependencies with known vulnerabilities to patched versions with fixes;
- Replacing unmaintained dependencies with secure and battle-tested alternatives, if possible;
- Pinning dependencies to specific versions, including pinning build-level dependencies in the `package.json` file to a specific version;
- Only upgrading dependencies upon careful internal review for potential backward compatibility issues and vulnerabilities; and
- Including Automated Dependency auditing reports in the project's CI/CD workflow.

Status

The Chia Network team has updated the dependencies across all projects (see here and here).

Verification

Resolved.

Appendix

Appendix A: In-Scope Components

Audit 1: Permuto App and Chia Signer iOS Mobile App

The following code repositories and features are considered in-scope for this review:

- Permuto App:
  - https://github.com/Chia-Network/permuto-app

- Chia Signer iOS Mobile App:
    - https://github.com/Chia-Network/chia-signer-ios

Audit 2: Enterprise Wallet and Chia Wallet SDK

The following code repositories and features are considered in-scope for this review:

- Enterprise Wallet:
    - https://github.com/Chia-Network/ent-wallet/
- Vaults:
    - Vault Implementation:
        - https://github.com/LeastAuthority/chia-wallet-sdk/tree/main/crates/chia-sdk-driver/src/primitives/vault
        - https://github.com/LeastAuthority/chia-wallet-sdk/blob/main/crates/chia-sdk-driver/src/primitives/vault/vault_launcher.rs
        - https://github.com/LeastAuthority/chia-wallet-sdk/tree/main/crates/chia-sdk-driver/src/primitives/mips
    - The MIPS and Vault binding wrapper::
        - https://github.com/LeastAuthority/chia-wallet-sdk/tree/main/crates/chia-sdk-bindings/src/mips
        - https://github.com/LeastAuthority/chia-wallet-sdk/blob/main/crates/chia-sdk-bindings/src/mips.rs
        - https://gist.github.com/Quexington/32936e9a03bbf4f1956f3538096f2a83
    - The section in the CLVM bindings that spends vault coins:
        - https://github.com/xch-dev/chia-wallet-sdk/blob/bbd411fe7e3ce773067ff6cc4db7b0cc315db01c/crates/chia-sdk-bindings/src/clvm.rs#L190-L220
    - Binding Tests:
        - https://github.com/LeastAuthority/chia-wallet-sdk/blob/main/napi/__test__/vaults.spec.ts
    - Puzzles:
        - https://github.com/Chia-Network/chia_puzzles/pull/22
        - https://github.com/xch-dev/chia-wallet-sdk/crates/chia-sdk-types/src/puzzles/mips
    - The Merkle Tree implementation:
        - https://github.com/xch-dev/chia-wallet-sdk/blob/bbd411fe7e3ce773067ff6cc4db7b0cc315db01c/crates/chia-sdk-types/src/merkle_tree.rs
- Token tracker
- Permuto API

Audit 3: CATs and Admin Tool

The following code repositories and features are considered in-scope for this review:

- Puzzles:
    - https://github.com/Chia-Network/chia_puzzles/pull/22
- CAT Admin Tool:
    - https://github.com/Chia-Network/CAT-admin-tool
    - Minting and melting of revocable CATs
    - Secure the bag payment
- CAT Revocation:
    - https://github.com/Chia-Network/chips/blob/c4ecddeeff2a9fad31dd8e69740e23e7dab9eb01/CHIPs/chip-0038.md

Out of Scope

The following items are considered out of scope for this review:

- Other unlisted actions defined within the CAT-admin-tool,
- CAT2 standard (in the CAT Revocation),
- Permuto processes,
- User brokerage,
- IRS Reporting, and
- Any dependency and third-party code, unless specifically included above.

The above in-scope audit target was provided by the Chia Network team to Least Authority and assessed for the purposes of this report.

In addition, any dependency and third-party code, unless specifically included above, were considered out of the scope of this audit.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, zero-knowledge protocols, and consensus protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's website to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. As we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present and possibly resulting in Issue entries, then for each, we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative and transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even before having verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate and comprehensive mitigations that live deployments can take, and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful

mitigation and remediation is an ongoing collaborative process after we deliver our Initial Audit Report, and before we perform a verification review.

Before our report, including any details about our findings and the solutions are shared, we like to work with your team to find reasonable outcomes that can be addressed as soon as possible without an overly negative impact on pre-existing plans. Although the handling of issues must be done on a case-by-case basis, we always like to agree on a timeline for a resolution that balances the impact on the users and the needs of your project team.

Resolutions & Publishing

Once the findings are comprehensively addressed, we complete a verification review to assess that the issues and suggestions are sufficiently addressed. When this analysis is completed, we update the report and provide a Final Audit Report that can be published in whole. If there are critical unaddressed issues, we suggest the report not be published and the users and other stakeholders be alerted of the impact. We encourage that all findings be dealt with and the Final Audit Report be shared publicly for the transparency of efforts and the advancement of security learnings within the industry.