NF ID Wallet

NFID Wallet

Audit Report

Audit by:

Solidstate

Audit Report Body

General Disclaimer

This audit report aims to detail the current state of security and potential weaknesses of the audited app within a limited scope. It does not make any recommendations regarding the app's user experience, investment potential, or other factors.

Scorecard

This scorecard is a broad overview of all our findings. You can also view the Summary of Findings from our initial audit review and then see how the NFID Labs team updated their code in response to those initial findings by viewing Post-Audit Review #1.

Area	Description
Access controls	**Strong:** Operational access controls are separated between admin (to be given to SNS governance), operator (which can perform backups and non-critical/sensitive state changes). User access controls are correctly enforced.
Architecture/Design	**Strong:** Despite some scalability issues that could be addressed with sharding in the long term, the architecture serves its purpose efficiently and doesn't introduce unnecessary complexity for the most part.
SNS Readiness	**Strong:** Canisters are ready to be upgraded and managed by an SNS. Proper roles have been added. Defensive switches (like pausing account creation) have also been added. Only missing reproducible build instructions, which could be easily added.
Scalability	**Sufficient:** The size of the identity_manager canister is the only limiting factor from a scalability standpoint, since the delegation_factory canister is reset on every upgrade.
Upgradeability	**Sufficient:** Both IM and DF canisters are upgradeable, there are some protections in place from state bloat that would prevent upgrades. More granular state size monitoring and clear limits would make this guaranteed.
Documentation	**Sufficient:** Documentation has been greatly upgraded during the course of the audit. There are now good instructions on how to build the canister and setup dev environments. Point of improvement would be to add more documentation about the client-server interactions and flows.

Area	Description
Code Quality & Safety	**Sufficient:** Server-side code has been upgraded to the latest best practices in terms of concurrency and state management and libraries have been upgraded. General practice of using unwrap to force traps and rollback state has been upgraded with meaningful error messages. Server code is very readable, low cyclomatic complexity and straightforward, with appropriate use of macros. Tests help understand functionality further. Client side code review was limited in scope, but could use more comments and is significantly harder to follow.
Testing & Verification	**Sufficient:** There are adequate tests both on the client and the server.

Scorecard levels

Poor	The scored area does not meet best practices. The area is very deficient, and improvements to it as a whole must addressed immediately.
Moderate	The scored area almost meets best practices. The area has some core deficiencies, but there are some key actionable items that, if addressed, would make this area minimally meet best practices.
Sufficient	The scored area matches best practices, even though some key, concrete items could be improved.
Strong	The scored area surpasses best practices, and only minor issues, if any, were found.

Table of contents

Introduction

This document is the official code security audit report by Solidstate for the NFID Wallet client and canister code. It represents a professional security review by a team of industry-native Internet Computer Protocol (”ICP”) experts.
The initial audit review was started on September 16, 2024, and completed October 25. Here are the team's latest code updates since the audit began, along with their resolutions and responses to the detailed list of initial audit findings.

Purpose

This security audit was mainly focused on NFID's delegation flow but also extended to the non-deprecated parts of the Identity Manager canister and some other immediately adjacent areas. Our assessment includes a detailed review of the smart contracts, delegation processes, and associated code to ensure robustness against potential vulnerabilities, with particular attention to the integrity of the delegation logic and potential attack vectors, as well as a general review of the canisters for security, upgradeability, and SNS readiness. The review of the client code was restricted to the delegation flows.

Auditing Agency

Solidstate is the first ICP-native code security auditing agency. It’s a venture by Code & State, which is an independent venture studio devoted to making it easier to build and earn on the Internet Computer. Solidstate contracts independent ICP protocol experts, with extensive experience building secure ICP decentralized applications to execute professional code security audits for ICP projects. Our audits follow accepted industry standards while expanding them to cover the unique security considerations of ICP decentralized applications.

Integrity Directives

Solidstate conducts audits with the highest standards of professional integrity and deep respect for the ICP ecosystem. We work with our clients to allow them some control over the scope and duration of the audits, but clients do not have any control over the content of the final audit report. Our audit reports must be an accurate representation of all findings during the audit.

Compensation for Solidstate audits is determined by a fixed USD-based daily rate contracted with the client upfront. To avoid any potential conflicts of interest, Solidstate does not accept native tokens or equity from a project as compensation for executing its audit. Neither Solidstate nor Code & State hold any equity stake in Identity Labs.

Solidstate auditors are instructed to follow these directives:

- The scope of the audit must be made clear in the audit report.
- The audit report must list any important security considerations that were not covered during the audit as out of scope.
- All known identified and verified issues must be included in the detailed list of findings on the audit report.
- Issues will remain listed as open or “not fixed” on the audit report until an appropriate resolution is verified during a post-audit review.
- In general, any significant considerations that might impact prospective investors and users of the decentralized application should be included in the final audit report.

Audit Team

Lead Auditor: David Alves

David is a leading mind in the development and security of decentralized infrastructure. While at DFINITY Foundation, he served as the Engineering Lead in charge of creating the NNS and the SNS. This experience gives him a rare level of protocol expertise that uniquely qualifies him as a leader in the emerging field of ICP security.

Security Researcher: Nima Rasooli

With a history of leading the development of multiple notable ICP projects, Nima now works full time as a Security Researcher at Solidstate, bringing a strong blend of technical knowledge and hands-on experience in the Web3 space.

Senior Auditor: Anonymous by request

Our senior auditor has co-founded notable projects on ICP since Genesis and brings over a decade of development experience. They also have experience creating security tools for ICP dApps and currently serve as a CTO at a Web2 startup.

Project Manager: Esteban Suarez

As Project Lead at Solidstate, Esteban oversees the audit’s progress, ensuring clear communication between the audit team and the client, and assists in the organization and delivery of the final report.

Technical Writer: Robin Moore

Robin edited the raw input of the auditors into this final report. Leveraging her expertise in editing, technical content creation, SEO, social media, and deep research, she strives to make technology accessible and engaging from beginners to enthusiasts.

Methodology & Assessment Priorities

State of Audits on ICP

ICP is unlike any other cloud network or L1 blockchain:

- It is asynchronous
- It hosts front ends
- It hosts complex data models
- It introduces new mutability types
- It decentralizes large amounts of data storage and significant computing capacity
- It can interact directly with both Web2 and other Web3 protocols
- It allows for multi-canister decentralized application architecture, which is a completely new technological frontier

As with any technology that pushes new boundaries, the ICP's advanced capabilities introduce novel security implications. Many established Web3 industry standards don't translate to ICP. As such, security audits on the Internet Computer Protocol is an emerging field in its early stages, with few standards, resources, and examples to pull from.

ICP auditors face the challenge of exploring and developing new industry methodologies so that decentralized applications on ICP can have their security properly evaluated. This means strong protocol expertise and extensive experience building on ICP are critical requirements for any ICP auditor.

Phase 1: Scoping

- NFID Labs provided access to their code repository, documentation, canister IDs, and all other required assets.
- NFID Labs and the Lead Auditor collaborated to scope the project and set risk assessment priorities.
- The lead auditor defined the audit assignments and security practices necessary for this audit.

Phase 2: Execution

- The auditors conducted a detailed code review of the provided NFID repository, in line with the scope defined during Phase 1.
- Each auditor deployed the code locally and evaluated it according to their assignments.
- The audit team collaborated heavily both internally and with NFID team throughout the audit.
- The audit team delivers the findings to the client in an initial audit review.

Phase 3: Finalization

- The auditors conducted a post-audit review to:
  i. Incorporate the NFID Labs team latest code updates since work on the initial audit review began.
  ii. Evaluate any attempted resolutions for issues identified during the initial audit review.
- The audit team collected the results into this final audit report.

Risk Assessment Priorities

These are the main risk assessment priorities that NFID was evaluated for:

Unauthorized Creation of Delegations

- **Description:** NFID relies on delegation mechanisms to authenticate users and grant access to canisters. Any vulnerability that allows unauthorized parties to create delegations could enable attackers to impersonate users, access sensitive data, or perform unauthorized actions within the network.
- **Impact:** Unauthorized delegations can lead to compromised user accounts, unauthorized transactions, and potential loss of assets. This undermines the trust in NFID's authentication system and can result in significant security breaches affecting both users and integrated services.

Weaknesses in 2FA/Self-Sovereign Mode

- **Description:** The self-sovereign mode, also known as passkey authentication or 2FA, is critical for securing user accounts by ensuring that only the passkey or recovery phrase owner can create delegations and perform authenticated actions. Weaknesses in the implementation of this mode could allow attackers to bypass authentication measures, especially if they can spoof devices or exploit flaws in device registration and verification processes.
- **Impact:** If attackers can circumvent the self-sovereign mode, they could gain unauthorized access to user accounts, manipulate account settings, or perform transactions, leading to loss of user assets and compromising the integrity of the NFID platform.

Vulnerabilities in Delegation Creation Process

- **Description:** The delegation creation process involves sensitive operations that require strict verification of user identities and devices. Flaws such as insufficient caller verification, inadequate device validation, or improper handling of delegation requests can be exploited to create unauthorized delegations.
- **Impact:** Exploitation of these vulnerabilities could lead to unauthorized access, privilege escalation, and compromise of user accounts, affecting both NFID and the security of integrated services and applications.

Insufficient Input Validation and Resource Management

- **Description:** Lack of proper input validation, such as unvalidated e-mail field sizes or delayed input checks, can make the system susceptible to resource exhaustion attacks. Attackers may submit oversized inputs or malformed requests to overload the canisters, leading to cycles draining or denial of service.
- **Impact:** Resource exhaustion can render NFID canisters unresponsive, affecting service availability for all users. This can result in significant operational disruptions and damage the platform's reputation.

Governance and Access Control Post-SNS Integration

- **Description:** Integration with the Service Nervous System (SNS) introduces changes to the governance model of NFID. If roles like 'admin' become obsolete or inaccessible without proper adjustments, critical functions may become unmanageable. Insufficient separation of duties and unclear role definitions can lead to improper access controls.
- **Impact:** Misaligned governance and inadequate role-based access control can result in unauthorized access to sensitive functions or the inability to perform critical operations. This affects the system's security and functionality, potentially leading to misuse or service disruptions.

Potential for Cycles Draining Attacks

- **Description:** Without adequate protections against cycles draining attacks, methods that require existing accounts but lack proper verification can be exploited. Attackers might create numerous accounts or send excessive requests to exhaust the canister's computational resources.
- **Impact:** Cycles exhaustion can lead to a denial of service, making the NFID canisters unresponsive and affecting all users. This poses a significant threat to service availability and can incur additional operational costs.

Disclaimers & Scope

Review watermarks

The initial audit review performed in the code was based on commit b32a9232b17c4abec6ba43b47f5a0147c34abfb6 of the NFID Wallet Server GitHub repository and on commit 3a5389f2af1aae4b93b6ba0c61b131b71f337f0e of the NFID Wallet Client Github repository. It was from this commit hash that the auditors reviewed all the code/documentation/scripts.
A post-audit review of fixes was done at commit: 5f7f19874999c1f29afa2fec9bf53a85b262bf39 for the server and 3f35d16e3142a121d17d2ee6244f656c5a49d905 for the client, and the reviews of the fixes can be found in Post-Audit Review #1

General Disclaimers

- **Experimental Nature of ICP dApps** - The security of decentralized applications on the Internet Computer (ICP) is still in its early stages, with many unknowns and evolving risks. Users should be aware of this experimental nature and are encouraged to conduct thorough research before using any ICP applications.

- **Use at your own risk** - Neither Solidstate, Code & State nor our auditors are liable for any potential losses or damages experienced by users of the NFID Wallet.

- **Not Financial or Legal Advice** - The information contained in this report is for technical and security purposes only and should not be interpreted as financial or legal advice.

- **Not comprehensive** - There is no such thing as a comprehensive audit, but there is always a limited scope. This report captures the work performed for this audit. Readers of this report should not assume that anything not explicitly covered here was evaluated.

- **No guarantees** - Due to the experimental nature of audits for ICP applications, neither Solidstate, Code & State nor the auditors make any guarantees regarding the security of the NFID Wallet. We conducted a best-effort audit with talented protocol experts in good faith, but that is not enough to guarantee that new exploits won't ever be discovered on a technology as novel as ICP.

- **Limited applicability** - ICP canisters have mutable code. This audit report was conducted on a specific commit hash of NFID Wallet, and the issues were verified to be fixed on another specific hash (see section above). Once the code is updated again, the applicability of this audit is voided, and incremental reviews are recommended, as future versions of the NFID Wallet source code may introduce new vulnerabilities.

Within Scope

NFID code for these canisters and components:

- **Identity Manager Canister** (referred to as "IM")
  - Methods marked as deprecated are out of scope

- **Delegation Factory Canister** (referred to as "DF")
- **Frontend Code** (referred to as "FE"), but at request of the developers strictly limited to the following packages/folders:
  - nfid-wallet-client/a/nfid-frontend/s/f/id/service/method/interactive
  - nfid-wallet-client/packages/integration/src/lib/delegation-factory

In general we looked at the following aspects of the code within scope:

- **Governance**
- **Upgradeability**
- **Scaling**
- **State/Variable Management**
- **Vulnerability to DOS and Cycles drain attacks**
- **Best Practices with the Code Repository** (documentation, build-ability, comments, etc.)

Outside of Scope

- **Account creation flow outside the IM canister:** The auditing process did not include the account creation flow that occurs outside of the Identity Manager (IM) canister.

- **Internet Computer Protocol-Level Exploits:** Auditing the entire Internet Computer Protocol, including tooling and packages provided by the DFINITY Foundation, was outside the scope of this audit.

- **Third-Party Integrations:** Security assessments of third-party services, libraries, or dependencies used by NFID, unless they directly impact the audited components, were not independently conducted.

- **Other NFID Applications and Canisters:** Any other NFID applications, canisters, or repositories not directly related to the Identity Manager, Delegation Factory, or Frontend, including the large majority of the Frontend, were outside the scope of this audit.

- **Client-Side Device Security:** Security of client devices, including browser security and local storage, was not assessed.

- **Testing:** While existing tests within the repository were reviewed, auditors did not develop additional tests for NFID or expand testing coverage in the repository.

Summary of Findings

In this section, we present a summary of the findings from the initial audit review. The full list and details are presented in the Detailed List of Findings section, and the findings for the latest code commit can be found in Post-Audit Review #1.

We audited the code across 3 main verticals.

1. **Security:** This section focuses on code correctness and potential DoS or other attack vectors.
2. **Quality & Documentation:** This section evaluates testing coverage, documentation quality, build reproducibility, and other essential quality indicators.
3. **SNS-Readiness:** This section assesses the architecture and code readiness for transfer to SNS DAO governance.

We classify our findings into the following categories:

Low	Stylistic or grammatical error that doesn't influence how the application runs.
Medium	Erroneous return or calculation that doesn't affect the integrity of the overall application.
High	A problem that affects the overall integrity of the application, but doesn't allow the extraction of funds.
Very High	A problem that affects the overall integrity of the application and allows a sophisticated caller to directly extract value or take control of critical application functions.

Tally of issues by severity

Category	Security Issues	Non-Security Issues	SNS-Readiness Issues
Very High	1	0	0
High	6	0	2
Medium	9	5	1
Low	6	0	0
Total	22	5	3

Security Concerns

Several high-severity vulnerabilities were identified, which were addressed and re-reviewed as part of the Post-Audit Review #1:

- **Attacker can make IM canister non-upgradeable and unrecoverable (SS-NFID-001, Very High):** By spamming the canister programmatically, an attacker can spam the IM canister with enough accounts such that the canister is out of memory and the pre-upgrade hook fails for lack of cycles, making the IM canister unrecoverable.
- **Configuration Loss After Upgrade (SS-NFID-002, High):** The Identity Manager (IM) canister does not store configuration in stable storage. After an upgrade, essential configurations are lost, potentially causing service disruptions and security vulnerabilities until the SNS governance canister resets them.
- **Lack of Email Field Size Validation (SS-NFID-003, High):** The IM canister does not validate the size of the e-mail field. Attackers could submit excessively large e-mails, causing out-of-memory errors and crashing the canister, leading to denial of service.
- **Absence of Cycles Drain Protection (SS-NFID-004, High):** The system lacks protections against cycles draining attacks. Methods do not verify account existence before processing, and account creation is not throttled, making the system vulnerable to denial-of-service attacks.
- **Unsafe Use of** storage::get_mut **(SS-NFID-005, High):** Using storage::get_mut allows multiple mutable references, violating Rust's safety guarantees and potentially leading to data races or undefined behavior, compromising data integrity.
- **Deprecated** ic-cdk **Version (SS-NFID-006, High):** The project uses an outdated version of the ic-cdk library, missing important security patches and features, increasing the risk of vulnerabilities.
- **Missing Caller Verification in** prepare_delegation **(SS-NFID-007, High):** The prepare_delegation method in the Delegation Factory lacks proper caller verification, allowing unauthorized creation of delegations and potential resource exhaustion attacks.

Quality & Documentation

Key issues affecting code quality and maintainability, which were partially addressed and re-reviewed as part of the Post-Audit Review #1:

- **Lack of Comprehensive Documentation (SS-NFID-015, Medium):** The codebase lacks sufficient comments and documentation, especially for public methods and critical logic, making it difficult to understand and maintain.
- **Inconsistent Error Handling and Logging (SS-NFID-012 & SS-NFID-014, Medium)**: Error handling is inconsistent, and the logging mechanism is unused, hindering debugging efforts and obscuring issue detection.
- **Insufficient Testing and Verification (SS-NFID-020, Medium)**: Critical functions lack proper testing, and there are no clear instructions on how to run or interpret tests, increasing the risk of undetected bugs.
- **Missing Build and Verification Instructions (SS-NFID-013, Medium)**: The project lacks clear instructions on how to build and verify the code, making it challenging for developers to set up the environment and reproduce builds.

SNS Readiness

Critical issues affecting the system's readiness for SNS integration, which were addressed and re-reviewed as part of the Post-Audit Review #1:

- **Need for Separation of Operator and Admin Responsibilities (SS-NFID-028, High)**: The current role management conflates admin and operator roles. Without proper separation, critical operations may be delayed due to governance processes, or security may be compromised if too much power is centralized.

- **Ineffective Backup System Post-SNS Integration (SS-NFID-029, High)**: The existing backup mechanism relies on admin-only methods, which will become ineffective after SNS integration. This could lead to data loss and hinder disaster recovery efforts.

- get_configuration **Method Becomes Inaccessible Post-SNS (SS-NFID-030, Medium)**:
  The get_configuration method is currently accessible only by the 'admin' role. After SNS integration, this role may not have direct access, rendering critical configuration data inaccessible when needed.

Application Architecture

System Overview

The NFID Wallet system is designed to securely manage user assets, especially when passkey authentication is activated. Below is an overview of the main components involved, their responsibilities, and how they communicate to ensure the integrity of the system.

Self-Sovereign Mode and Security Validation

- **Self-sovereign mode** is activated via the is2fa_enabled field and enforced by the secure_2fa method. When enabled, only registered devices (WebAuthn, LedgerKeys, or RecoveryPhrase) can modify a user's account state.
- **Device Authentication:** All user devices are registered as access_points under the Identity Manager canister, with unique public keys that are validated during delegation requests.

User Delegation Flow

- **The delegation flow** begins when a user requests delegation through one of the integration layer methods. This involves validating the user identity, generating the delegation chain, and verifying the origin of the request.
- **The delegation identity** is unique to each device, serving as the access point for the user's account. Delegations are stored in IndexedDB and managed by authState for static access during user sessions.

Inter-Canister Communication

- **The Identity Manager and Delegation Factory** canisters communicate using **Inter-Canister Query Calls (ICQC)** to validate requests and ensure that only authenticated users can create delegations.
- This communication ensures that only authorized users or devices can interact with the backend to initiate actions like delegation creation or state modification.

Standards and RPC Services

- The system uses implementations of standards such as **ICRC-34, ICRC-49**, and **ICRC-27** to facilitate various user interactions, including delegation, canister calls, and account-related services.
- Each service (icrc34-delegation-method.service.ts, icrc49-call-canister-method.service.ts, icrc27-accounts-method.service.ts) receives requests through an rpcReceiver and either processes the request internally or forwards it to the backend to obtain necessary delegations or perform user-related actions.

Security Considerations

- The audit scope is primarily concerned with ensuring that **only the owner of a passkey or recovery phrase** can create a delegation or make authenticated calls to other canisters. Unauthorized access is mitigated through strict validation of public keys and ensuring only registered devices can initiate modifications.
- **Delegation Creation Security**: The delegation process requires an additional comparison between user anchors to verify ownership before issuing a delegation.

Communication Flow Summary

1. **User Requests Delegation:** Through the frontend, the user requests delegation for account interactions (e.g., canister calls).
2. **Integration Layer:** The integration layer manages the delegation request and forwards it to the backend through rpcReceiver.
3. **Delegation Factory:** The Delegation Factory interacts with the Identity Manager to validate user anchors and generate the delegation chain.
4. **Identity Manager:** The Identity Manager canister verifies if the user anchor exists, and the device is registered for access.
5. **Delegation Issuance:** After successful validation, a delegation chain is signed and provided to the user, which is stored in IndexedDB and used for secure interactions.

Risks & Considerations

Considerations:

The NFID project architecture is divided in to several components: wallet frontend, several backend canisters, AWS Lambda functions, and npm library for integration between dApps and the NFID Wallet. This multi-tiered architecture spans both on-chain and off-chain components, providing authentication services through various integration points.

Documentation for the frontend contains instructions on how to prepare the local environment for development and how to run E2E tests. There is also mention about general architecture and coding standards used during development.

Documentation for the backend contains several commands for the local environment setup and running tests. There is no mention about architecture of the solution or links to used standards. Provided prerequisites are missing the required Rust version, which might cause issues with building and testing. It lacks detailed usage instructions, insights into architecture, and used standards (like Internet Identity or different ICRC standards).

Security Measures:

The delegation_factory canister operates without explicit role definitions, as it contains no configuration or administrative functions requiring access control.

The identity_manager canister contains several admin roles: admin, operator, lambda. The admin role protects the configuration of the canister. operator is used for several maintenance functions necessary for oversight of the canister. lambda is reserved for usage by the AWS Lambda component.

Governance:

The delegation_factory is mostly a stateless canister, so there is no need to backup and restore canister's state.

The identity_manager canister stores user account data and provides public functions accessible to the operator role for state management and backup operations.

Scalability:

While the whole product is divided in to several parts, the main endpoint to retrieve authorization is identity_manager. In the event of high demand or subnet congestion, there is no option to scale identity_manager by creating new ones on different subnets. Scaling beyond a single identity_manager canister will require a smart approach to sharding identities across them, with added redundancy.

Auditability and Transparency:

The project implements a complex identification system composed of multiple interconnected components. The current documentation requires a more comprehensive approach and provides information about:

1. Detailed documentation of individual components, their architecture, and their role in the solution
2. An architecture overview explaining component interactions
3. A comprehensive explanation of the identification workflow across the system

These improvements would facilitate better understanding of the system's identification functionality and its implementation.

Risks:

There is no mention of external monitoring tools. Given the interdependent nature of the system's components, a failure or unavailability of any single component could render the entire solution inoperable. The absence of monitoring infrastructure may extend incident response times and complicate root cause analysis during system failures.

Recommendations

Governance

The delegation_factory canister is well suited to be managed by an SNS. It can cleanup delegations to prevent OOM by upgrading; it's easily upgradeable and doesn't require any major changes to have the process managed by an SNS governance canister. On the other hand, the identity_manager canister presents some deficiencies that must be addressed before handing over control to an SNS governance canister, the list of which can be found in the SNS Readiness Related Findings section.

Scalability

Consider preparing a sharding strategy of identity_manager to enable horizontal scaling. Strategy should include topics like: subnet deployment, redundancy mechanism, and load balancing approach.

One of such approaches could leverage frontend-based load balancing. This strategy is optimal because:

1. The fronted is stateless; it is easier to scale than canisters
2. Client-side load balancing eliminates the need for additional on-chain coordination
3. Identity distribution logic can be implemented without modifying canister architecture

This approach could minimize required changes while enabling horizontal scaling of the identity_manager functionality.

Readability

The code is well structured; coding standards are set and followed.

Implement thorough inline documentation in frontend code to enhance maintainability, streamline developer onboarding, and facilitate future development and contributions.

This documentation should detail the reasoning behind implementation decisions and clarify non-obvious code paths throughout the frontend codebase.

The frontend codebase implements standards from the DFINITY Identity and Authentication Working Group, but it lacks explanatory context for their usage.

Maintainability/Upgradability

The following points have already been mentioned but are re-mentioned here since they affect the maintainability and upgradeability of the system.

1. Reproducible builds: there should be a means and instructions for anyone to build the canister artifacts to verify that the upgraded wasms match the code.

Tests

Tests are an essential part of the overall solution, and each audited component includes a comprehensive set of tests. However, some components may require extra steps to run these tests in a local environment, which should subsequently be added to the documentation.

Risk Mitigation

1. Develop an external monitoring solution: While the Internet Computer (IC) is a strong blockchain platform, it's important to set up an additional external monitoring system for every part of the solution. This adds an extra layer of oversight to help catch potential issues that might not be handled by IC alone.

2. Add the possibility to freeze/lock accounts. Since accounts can be accessed via various means, it increases the chances of compromise due to phishing or other methods. If a user or operator suspects that given account is breached, there should be a way to lock it. There should also be a clear path to recover access once the breach is resolved.

Components

1. Identity Manager (IM) Canister

- **Purpose:** Acts as a registry for public keys associated with users, maintaining their identities and managing device-based access.
- **Key Roles:** The Identity Manager handles account creation, anchor assignment, and registers device keys as user access points. It verifies user ownership and determines if the self-sovereign mode (2FA) is enabled.
- **Communication:** Receives requests from the Delegation Factory canister for anchor validation. Uses inter-canister query calls to interact with other canisters.

2. Delegation Factory (DF) Canister

- **Purpose:** Responsible for creating delegations that allow secure interactions between user accounts and various canisters.
- **Key Roles:** The delegation process involves authenticating the user's identity, validating self-sovereign mode, and issuing delegations only to valid devices (e.g., WebAuthn devices).
- **Communication:** The DF canister interacts with the IM canister to verify that the user requesting delegation is valid and has self-sovereign mode enabled, using inter-canister query calls. Delegations are only created after verifying user anchors and devices.

3. Frontend Integration Layer

- **Purpose:** The integration layer is responsible for enabling the client-side flow for creating delegations and managing user identities.
- **Key Roles:** Manages user identity using the methods – getGlobalDelegation, getGlobalDelegationChain, and getAnonymousDelegation. These methods handle requests for delegation either for internal usage (e.g., within nfid.one) or for third-party apps.

- **Communication:** Requests for creating or modifying delegations are forwarded to the backend services through RPC calls, with messages passed between the client, rpcReceiver, and delegation services.

Repository Assessment

Documentation

The existing documentation provides basic guidance for build procedures and test execution. However, it contains gaps that prevent efficient local development and testing processes. These omissions are related to the required setup of a local environment necessary for build and test reproducibility.

Instructions regarding build and test should contain complete guidelines on clean systems, from cloning to successful test and build runs.

Build Errors

All projects build successfully. However, canisters compile with warnings. The identity_manager canister has 95 warnings, primarily related to deprecated code and unused functions and variables. The delegation_factory canister has 11 warnings only about unused imports or functions.

Compiler warnings for rust should be addressed by either removing unused code or marking with appropriate ignores.

Tests

The project includes a comprehensive set of tests. The TypeScript-based nfid-frontend and Rust-based delegation_factory and identity_manager projects include comprehensive test suites that verify core functionality. The testing approach aligns appropriately with each project's technology stack.

The nfid-frontend incorporates unit tests, integration tests, and end-to-end tests which ensure functionality across different levels of the application stack.

This extensive test coverage provides confidence in system reliability and ensures safe future modifications.

Github Actions

All projects implement automated GitHub Actions workflows for continuous integration, enforcing consistent build and test execution across all components. This automation is particularly valuable for an identity management system handling user funds, as it ensures reliable validation of all code changes. The reproducible build process adds an essential layer of security verification to the development lifecycle.

This automation enhances system reliability by enforcing quality checks and reducing the possibility of human error during the development process.

Detailed List of Findings

Summary

To recap, we classified our findings into the following categories:

Low	Stylistic or grammatical error that doesn't influence how the application runs.
Medium	Erroneous return or calculation that doesn't affect the integrity of the overall application.
High	A problem that affects the overall integrity of the application but doesn't allow the extraction of funds.
Very High	A problem that affects the overall integrity of the application and allows a sophisticated caller to directly extract value or take control of critical application functions.

Security Related Findings

SS-NFID-001: Attacker make IM canister un-upgradeable

Component: Identity Manager (IM) Canister

Severity: Very High

Details: The Identity Manager (IM) canister doesn't have a hard cap on the total number of accounts. A attacker could spam the canister with thousands or even millions of accounts.

Implication: Suppose the number of accounts gets too big. Eventually, the total number of accounts can't persist in stable memory on pre-upgrade (namely, because the canister runs out of cycles to run the pre-upgrade hook). If this happens the canister becomes un-upgradeable and unrecoverable.

Recommendations:

i. **Implement a hard cap:** There should be a hard cap ****on the maximum number of accounts so that it's always possible to upgrade the canister.

ii. **Follow best practices for canister upgrades:** For example, accounts could be stored directly in stable memory using the ic-stable-structures library. For more information see: https://internetcomputer.org/docs/current/developer-docs/backend/rust/upgrading

iii. **Monitor and control:** Monitor the number of accounts and other metrics to detect malicious activity. Add pause controls to the canister(s) so they may react in time

SS-NFID-002: Configuration Not Stored in Stable Storage Causes Loss After Upgrade

Component: Identity Manager (IM) Canister

Severity: High

Details: The Identity Manager (IM) canister does not persist in its configuration in stable storage. This means that, after an upgrade, all configuration settings are lost, leaving the IM canister without essential configurations. Since only the SNS governance canister can set these configurations, there might be a period after an upgrade where the IM canister operates without necessary configurations, leading to potential service disruptions.

Implication: Loss of configuration after an upgrade can result in service downtime, unexpected behavior, or security vulnerabilities until the configuration is restored. This poses a significant risk to the system's reliability and security.

Recommendations:

i. **Persist Configuration:** Store all configuration data in stable storage to ensure it survives canister upgrades.

ii. **Backward Compatibility:** Ensure that the configuration storage format remains backward compatible to prevent issues during updates.

SS-NFID-003: Lack of Email Field Size Validation Leading to Potential OOM Errors

Component: Identity Manager (IM) Canister

Severity: High

Details: The IM canister does not validate the size of the e-mail field provided by users. An attacker could exploit this by submitting an excessively large email string, causing the canister to run out of memory (OOM) and potentially crash.

Implication: An OOM error can lead to a denial of service, making the canister unresponsive and affecting all users. This vulnerability can be exploited to disrupt services and impact system availability.

Recommendations:

i. **Input Validation:** Implement strict validation on the e-mail field to enforce a reasonable maximum length.

ii. **Reject Oversized Inputs:** Immediately reject any input that exceeds the defined limit with an appropriate error message.

SS-NFID-004: Absence of Cycles Drain Protection Mechanisms

Component: All Canisters

Severity: High

Details: The system lacks adequate protections against cycles draining attacks. Methods that require existing accounts do not verify the account's existence before processing, and account creation methods are not throttled. Additionally, there is no mechanism for operators to pause account creation during suspected abuse.

Implication: Attackers could exploit these vulnerabilities to exhaust the canister's cycles, leading to denial of service. Unchecked account creation could be used to overwhelm the system.

Recommendations:

i. **Implement inspect_message Methods:** Verify account existence before processing requests that require an existing account.

ii. **Throttle Account Creation:** Introduce rate limiting on account creation to prevent abuse.

iii. **(Optional) Pause Mechanism:** Provide operators with the ability to pause account creation during attacks or high-load periods..

SS-NFID-005: Unsafe Use of storage::get_mut Allowing Multiple Mutable References

Component: Identity Manager (IM) Canister

Severity: High

Details: he use of storage::get_mut in the code allows access to multiple non-exclusive mutable references, violating Rust's safety guarantees and potentially leading to data races or undefined behavior. Here is a concrete example where this is a problem: For instance, in AccountService::update_account, if an e-mail is part of the update, then there's a call to a service to validate the e-mail and the principal. However, because get_mut was used, AccountService is obtained in non-exclusive mode, meaning that another call might come in for the same account, the AccountService::remove_account, while the mail service is being queried and completed successfully. After remove account completes, the AccountService::update_account might still complete as well, having the account updated with AccountService::store_account . But, because this is different from the AccountService::create_account , which does more than just update the account in the map, the whole account might be left in an inconsistent state.

Implication: Breaking Rust's ownership and borrowing rules can introduce hard-to-detect bugs, compromising data integrity and system stability. This is especially true as implementations evolve and may have inter-canister calls added.

Recommendation:

- **Safe Reference Handling:** Refactor the code to ensure that only one mutable reference exists at any time. For instance, by using the same method/strategy as in the DelegationFactory canister.

SS-NFID-006: Use of Deprecated ic-cdk Version with Potential Security Issues

Component: All Canisters

Severity: High

Details: TThe project uses an outdated version of the Internet Computer's SDK (ic-cdk), specifically v0.3.2, while the current version is v0.16. This outdated version may lack important security patches, optimizations, and features present in newer releases.

Implication: Using deprecated dependencies increases the risk of security vulnerabilities and may lead to compatibility issues.

Recommendation:

- **Upgrade Dependencies:** Update ic-cdk to the latest stable version.

SS-NFID-007: Missing Caller Verification in prepare_delegation Method

Component: Delegation Factory (DF) Canister

Severity: High

Details: The prepare_delegation method lacks verification of the caller's identity. While it does not introduce a direct vulnerability in isolation, this method creates delegations—a sensitive operation that should be restricted to authorized entities.
Moreover, this allows an authenticated party to create arbitrary numbers of delegations, thus possibly making the canister run out of cycles, memory, or both.

Implication: Without caller verification, unauthorized parties might exploit this method to create delegations, potentially leading to security breaches or unauthorized access.

Recommendations:

i. **Add Caller Verification:** Implement checks to ensure that only authorized principals can invoke prepare_delegation. Potentially use the same method as in get_delegation.

ii. **Align with Best Practices:** Follow the approach used in the Internet Identity (II) implementation, which includes additional safeguards.

SS-NFID-008: No cleanup of delegations might lead to out of memory

Component: Delegation Factory (DF) Canister

Severity: Medium

Details: Despite the fact that delegation has a time-to-live, they are never cleaned up in the delegation_factory.

Implication: Since delegations are never cleaned up, eventually, the canister will run out of memory, despite the fact delegations could easily be cleaned up on a timer loop. Note that on upgrades, delegations are cleaned up, which is why this is not a High-severity issue.

Recommendation:

- Clean up expired delegations on a timer.

SS-NFID-009: Exposure of Debugging Methods in Production Frontend Code

Component: Frontend (fe)

Severity: Medium

Details: The frontend code includes window.resetAuthState and window.setAuthState methods, which appear to be intended for debugging purposes. If included in production builds, these methods could be exploited to manipulate authentication states.

Implication: Exposing debugging functions in production can lead to security vulnerabilities, allowing attackers to alter authentication flows or bypass security mechanisms.

Recommendations:

i. **Remove Debugging Code:** Exclude debugging methods from production builds using appropriate build configurations.

ii. **Secure Global Objects:** Avoid attaching sensitive methods to the window object or any global scope in production code.

iii. **Review Build Process:** Ensure the build process strips out or guards any development-only code.

SS-NFID-010: Delayed User Input Validation Leading to Inefficient Computations

Component: Identity Manager (IM) Canister

Severity: Medium

Details: In the update_account method and similar functions, user input validations are performed after fetching data from storage and performing computations. Validating user inputs later in the process wastes resources if the inputs are invalid.

Implication: This practice leads to unnecessary cycles consumption and increases the risk of denial-of-service attacks, as the canister performs expensive operations before rejecting invalid requests.

Recommendations:

i. **Early Validation:** Perform all user input validations at the beginning of the method before any data fetching or heavy computations.

ii. **Optimize Control Flow:** Refactor code to short-circuit and return errors immediately upon detecting invalid inputs.

SS-NFID-011: Unhandled Errors in Delegation Factory Methods

Component: Delegation Factory (DF) Canister

Severity: Medium

Details: In the get_delegation and get_principal methods, if calls to the IM canister fail (e.g., if the IM canister is unavailable), the code uses .unwrap(), causing the method to trap and provide no meaningful error messages.

Implication: Trapping without proper error handling leads to poor user experience and makes debugging difficult, as clients receive no information about the failure's cause.

Recommendations:

i. **Implement Error Handling:** Replace .unwrap() with proper error handling using match or if let constructs to handle None or Err cases gracefully.

ii. **Return Informative Errors:** Provide meaningful error responses to the client, indicating the nature of the failure.

SS-NFID-012: Inconsistent Error Handling and Logging in Frontend

Component: Frontend (fe)

Severity: Medium

Details: Error handling and logging are inconsistently implemented in the frontend code. While some functions handle errors and log them, many others, especially canister calls, do not catch exceptions or log errors, potentially leading to silent failures.

Implication: Inconsistent error handling can result in a poor user experience, make debugging difficult, and allow issues to go unnoticed, affecting application reliability.

Recommendations:

i. **Standardize Error Handling:** Implement a consistent error handling strategy across all frontend code.

ii. **Proper Logging:** Ensure that errors are logged with sufficient detail to aid in debugging.

SS-NFID-013: Lack of Clear Build and Verification Instructions

Component: All Components

Severity: Medium

Details: The project lacks comprehensive instructions on how to build and verify the code. Key details like required versions of Rust, npm, DFX, and other tools are missing, making it difficult to set up the development environment.

Implication: Without clear instructions, developers may face challenges in reproducing builds, leading to inconsistencies, build failures, or security issues due to incorrect tooling versions.

Recommendation:

- **Update Documentation:** Provide detailed setup instructions, including tool versions and installation steps.

SS-NFID-014: Unused Logging Mechanism and Inconsistent Error Reporting

Component: All Components

Severity: Medium

Details: A logging module is defined and stored in stable storage, but it appears unused throughout the code. Additionally, error reporting is inconsistent, with frequent use of panic or trap without meaningful messages.

Implication: Lack of logging and inconsistent error messages make it difficult to diagnose issues, monitor system health, and improve the application's reliability.

Recommendations:

i. **Utilize Logging Module:** Actively use the logging mechanism to record important events and errors.

ii. **Improve Error Messages:** Provide clear and informative error messages instead of generic panics.

iii. **Error Handling Standards:** Establish and follow error handling and logging standards across the codebase.

SS-NFID-015: Lack of Comments and Documentation in Codebase

Component: All Components

Severity: Medium

Details: The codebase lacks comments and documentation, especially around public methods and critical logic. This absence makes it difficult to understand the system's functionality and hinders collaboration and maintenance.

Implication: Without proper documentation, new developers may struggle to contribute, and code verifiers will struggle to validate the code, increasing the risk of bugs and slowing development.

Recommendations:

i. **Document Public Methods:** Add comments explaining the purpose, parameters, and expected behavior of public functions.

ii. **(Optional) Explain Complex Logic:** Provide inline comments for complex or non-obvious code sections.

SS-NFID-016: Account Recovery Flow May Unexpectedly Create Accounts

Component: Identity Manager (IM) Canister

Severity: Medium

Details: The account recovery flow attempts to create a new account if one does not exist, a behavior that may be unexpected and is not documented. Users invoking account recovery may not anticipate that a new account will be created.

Implication: This could lead to confusion, unintended account creation, or security concerns if users are unaware of this functionality.

Recommendations:

i. **Clarify Functionality:** Document this behavior clearly in the method's documentation.

ii. **Separate Concerns:** Consider separating account creation from recovery to align with the principle of least surprise.

iii. **User Confirmation:** Prompt users before creating a new account during recovery to ensure they intend this action.

SS-NFID-017: Extensive Use of unwrap() Without Informative Error Messages

Component: All Components

Severity: Low

Details: The code extensively uses unwrap() on Option and Result types without providing custom error messages. This practice can cause the application to panic with generic messages when encountering None or Err values.

Implication: Panicking without informative messages hinders debugging and can lead to unexpected crashes, affecting system reliability.

Recommendations:

i. **Use expect() with Messages:** Replace unwrap() with expect() and provide meaningful error messages.

ii. **Proper Error Handling:** Where appropriate, handle errors gracefully using pattern matching or the ? operator and returning Result.

SS-NFID-018: Unnecessary Cloning in Account Recovery Flow

Component: Identity Manager (IM) Canister

Severity: Low

Details: In the recover_account method, the code unnecessarily clones the wallet parameter during pattern matching. This cloning is redundant and can be avoided.

Implication: Unnecessary cloning can lead to minor performance overhead, especially if done frequently or with large data structures.

Recommendation:

- **Avoid Redundant Clones:** Refactor the code to use references or pattern matching without cloning.

SS-NFID-019: prepare_delegation Method Unnecessarily Asynchronous

Component: Delegation Factory (DF) Canister

Severity: Low

Details: The prepare_delegation method is declared as async but does not perform any asynchronous operations. This unnecessary use of async can introduce confusion as the semantics of calls with async differ (because the reader will assume there's an inter-canister call somewhere).

Implication: Using async without need can mislead developers about the method's behavior.

Recommendation:

- **Remove async Keyword:** Modify the method to be synchronous if it does not perform asynchronous tasks.

SS-NFID-020: Lack of Clear Instructions on Running and Interpreting Tests

Component: All Components

Severity: Low

Details: The project does not provide clear instructions on how to run tests or interpret their results. Without guidance, developers may overlook testing or misinterpret test outcomes.

Implication: Inadequate testing practices can lead to undetected bugs and reduced code quality, as developers may not effectively verify their changes.

Recommendations:

i. **Testing Documentation:** Include detailed instructions on running tests in the project documentation.

ii. **Explain Test Suites:** Provide information on different test suites, what they cover, and how to interpret failures.

SS-NFID-021: Empty wasm_get_random file

Component: Delegation Factory (DF) Canister

Severity: Low

Details: The wasm_get_random file appears to be empty or incomplete.

Recommendation:

- **Remove:** If the file is not necessary, remove it to avoid confusion.

SS-NFID-022: Critical Line in account_service.rs Lacks Documentation and Testing

Component: Identity Manager (IM) Canister

Severity: Low

Details: Line 275 in account_service.rs is crucial for preventing phishing attacks on AccountResponses, but it lacks comments and is not covered by tests.

Implication: Without documentation, developers may modify or remove this line unintentionally, reintroducing vulnerabilities. Lack of tests means its functionality is not verified during changes.

Recommendations:

i. **Add Comments:** Document the importance of this line and how it prevents phishing.

ii. **Include in Tests:** Write tests that verify the security measures provided by this line.

Non-Security Issues Identified

SS-NFID-023: README Lacks Context and Complete Setup Instructions

Component: Frontend (fe)

Severity: Medium

Details: The README file provides minimal context and lacks comprehensive setup instructions. Users attempting to run the frontend may encounter errors during wallet creation or login due to missing or unclear steps.

Implication: Incomplete documentation can hinder new developers or users from engaging with the project, slowing down adoption and contributions.

Recommendations:

i. **Enhance README:** Provide step-by-step setup instructions, including prerequisites and environment configuration.

ii. **Troubleshooting Guide:** Include common issues and their resolutions.

iii. **User Guidance:** Offer clear guidance on how to create wallets and log in.

SS-NFID-024: Lack of Comments in Frontend Code

Component: Frontend (fe)

Severity: Medium

Details: The frontend code is largely uncommented, making it difficult to understand the logic and flow of the application. This lack of documentation impedes maintenance and onboarding of new developers.

Implication: Poorly documented code increases the risk of bugs, slows down development, and makes collaboration challenging.

Recommendations:

i. **Document Code:** Add comments explaining the purpose and functionality of components, functions, and complex logic.

ii. **Maintain Documentation Standards:** Encourage consistent documentation practices among developers.

SS-NFID-025: Inconsistent Node.js Version Recommendations

Component: Frontend (fe)

Severity: Medium

Details: The README recommends using Node.js v20, while GitHub Actions use Node.js v18. This inconsistency can lead to compatibility issues and confusion among developers.

Implication: Developers may encounter build errors or unexpected behavior if they use a Node.js version different from what the CI/CD pipeline uses.

Recommendations:

i. **Standardize Node.js Version:** Agree on a specific Node.js version for the project.

ii. **Update Documentation and CI/CD:** Ensure the README and CI configurations use the same Node.js version.

iii. **Specify Engines:** Use the engines field in package.json to enforce the Node.js version.

SS-NFID-026: Lack of Documentation on InteractiveMethodService Usage

Component: Frontend (fe)

Severity: Medium

Details: Implementations of ICRC features rely on InteractiveMethodService, but there is no documentation explaining its usage or interaction with the project. Understanding how it integrates with IdentityKitRPCCoordinator is also unclear.

Implication: Developers may struggle to understand or modify these features, leading to potential errors or inefficiencies.

Recommendations:

i. **Add Documentation:** Explain the purpose and usage of InteractiveMethodService and how it relates to ICRC features.

ii. **Provide Examples:** Include code examples or usage scenarios.

iii. **Clarify Integrations:** Document how IdentityKitRPCCoordinator fits into the overall architecture.

SS-NFID-027: Development and Testing Depend on Unspecified DFX Version

Component: Frontend (fe)

Severity: Medium

Details: Development and testing are performed using DFX version 0.17.0, but this requirement is not mentioned in the documentation. Developers using different DFX versions may encounter issues.

Implication: Lack of version specification can lead to incompatibilities, build failures, and wasted time troubleshooting environment issues.

Recommendations:

i. **Specify DFX Version:** Clearly state the required DFX version in the documentation.

ii. **Provide Installation Instructions:** Guide developers on how to install or manage the correct DFX version.

iii. **Consider Upgrading:** Evaluate the feasibility of updating to a newer DFX version to stay current with tooling improvements.

SNS Readiness Related Findings

SS-NFID-028: Need for Separation of Operator and Admin Responsibilities

Component: Identity Manager (IM) Canister

Severity: High

Details: The current role management within the IM canister conflates administrative and operational responsibilities under a single 'admin' role. With the upcoming SNS (Service Nervous System) integration, this setup becomes problematic. It would be beneficial to separate these duties into at least two distinct roles:

- **Admin:** Assigned to the SNS governance canister, with actions mediated through governance proposals.
- **Operator:** Designated principals who can execute queries, monitor system status, and perform account recovery without needing governance proposals.

Implication: Without proper role separation, critical operations may be delayed due to governance processes, or security may be compromised if too much power is vested in a single role. This can hinder system responsiveness and increase vulnerability to misuse.

Recommendations:

i. **Implement Role-Based Access Control:** Separate roles into 'Admin' and 'Operator' with clearly defined permissions.

ii. **Assign Appropriate Privileges:** Ensure the 'Admin' role is taken by the SNS governance canister, while 'Operators' have the necessary access for operational tasks.

iii. **Update Access Controls:** Modify canister methods to check for the appropriate role before execution.

SS-NFID-029: Ineffective Backup System Post-SNS Integration

Component: Identity Manager (IM) Canister

Severity: High

Details: The current backup system relies on methods (get_all_accounts_json and add_all_accounts_json) accessible only by the 'admin' role. After SNS integration, the 'admin' role becomes obsolete or inaccessible, rendering these methods ineffective. Additionally, these methods lack paging, leading to potential size limitations and incomplete backups.

Implication: The inability to perform backups can lead to data loss and hinder disaster recovery efforts. Without effective backups, account data may be irretrievable in case of system failures.

Recommendations:

i. **Implement Paging:** Modify backup methods to support paging, allowing large datasets to be handled efficiently.

ii. **Adjust Access Controls:** Make backup methods accessible to the 'operator' role or another appropriate role post-SNS.

SS-NFID-030: get_configuration Method Becomes Inaccessible Post-SNS Integration

Component: Identity Manager (IM) Canister

Severity: Medium

Details: The get_configuration method is currently accessible only by the 'admin' role. After integrating with SNS, this role may not have direct access, rendering the method unreachable and the configuration data inaccessible when needed.

Implication: Critical configuration information may become inaccessible, hindering system maintenance, troubleshooting, and the ability to make informed decisions based on current settings.

Recommendations:

i. **Update Access Controls:** Modify the method to be accessible by the 'operator' role or another appropriate role post-SNS.

Post-Audit Review #1

A “post-audit review” is essentially another audit focused on reviewing and incorporating all changes in the code since the previously audited commit. The majority of this report was based on the “initial audit review”, and this section covers the findings of the first and last “post-audit review” required to incorporate code changes leading up to the publication of this final audit report.

Status of found issues

Summary

text	Found	Fixed/Partially Fixed	Not an Issue	Not fixed (see Notes)
Very High	1	1	0	0
High	8	6	0	1
Medium	16	13	0	3
Low	5	5	0	0

Detailed statuses

Finding	Severity	Status	Commit	Notes
SS-NFID-001	Very High	Fixed	9273ff085157f3109f64d649d4e0324eaf23dc3c	Developer notes: “Attackers can no longer make the IM canister un-upgradable because the Operator role can pause account creation.” Auditor notes: This helps but needs to be complemented with monitoring alerting. There should be an empirically calculated max number close to which an alert is triggered and account creation is paused.
SS-NFID-002	High	Fixed	1eec0dcf3328a935ebbde00890a1cd1efaf86bc9	Auditor notes: Configuration is now saved to stable storage.
SS-NFID-003	High	Fixed	a1646160870709d61d97ac7e1952f229bea199f7	Auditor notes: email size is now validated to 320 chars

Detailed statuses

Finding	Severity	Status	Commit	Notes
SS-NFID-004	High	Partially Fixed	9273ff085157f3109f64d649d4e0324eaf23dc3c	Developer Notes: “Attackers will be unsuccessful in their attempts to drain cycles because the Operator role can pause account creation. In the event of an attack spamming delegation creation, canisters can always be restarted to clean memory. inspect_message will not be useful in this case because it doesn’t support inter-canister calls, on which the delegation creation mechanism relies. In the future, we might use inspect_message in composite query calls to add cycle drain protection (https://forum.dfinity.org/t/composite-query-in-inspect-message-mode/36523).” Auditor notes: “inspect_message” could still be used in update methods in the IM canister, even just to reduce general cycles consumption. For instance all methods that are tagged with the “two_f_a” tag could check the 2fa on the ispect message call.
SS-NFID-005	High	Fixed	a420baa1939d6f3dbd05c46a397f473c50a0325c	Auditor notes: Not only was storage::get_mut removed but the ic-cdk lib was upgraded and usage was upgraded to recommended best practices.
SS-NFID-006	High	Fixed	a420baa1939d6f3dbd05c46a397f473c50a0325c	Auditor notes: SS-NFID-005 notes
SS-NFID-007	High	Not Fixed	-	Developer notes: “Given that this doesn’t introduce a direct vulnerability, we decided to improve user experience instead and handle potential cycle drain attacks in the same way as 001 and 004. Internet Identity doesn’t have this issue because both methods are in one canister, whereas we use a microservices canister architecture. In the future, we might use inspect_message in composite query calls to add cycle drain protection (https://forum.dfinity.org/t/composite-query-in-inspect-message-mode/36523).”

Detailed statuses

Finding	Severity	Status	Commit	Notes
SS-NFID-008	Medium	Fixed	f2ee47e309d64e49713e5455b52a4ec5d364ff6f	Auditor notes: There is now an operator that can clean the state.
SS-NFID-009	Medium	Fixed	f2632bcb8bb5b6829e8317356e8ee3c46e2b75c0	Auditor notes: There is not a global var that detects the environment (local, test, dev, etc) and doesn't allow to resetAuthState unless it's non sensitive env.
SS-NFID-010	Medium	Fixed	a420baa1939d6f3dbd05c46a397f473c50a0325c	Auditor notes: update_account was removed
SS-NFID-011	Medium	Fixed	99b712bde5b1c9182d2a7974608b88d93f8e6652 88f1fedc9eb1fb025bad4ad1a693d8f0215a8435	Auditor notes: The developers opted to retain the panics but change the unwrap to expect with meaningful errors messages.
SS-NFID-012	Medium	Not Fixed	-	Developer notes: "Although there is room for improving the user experience by better handling and logging errors, we believe that the existence of e2e, unit, integration, and manual tests for every important user flow on every release will catch the vast majority of potential frontend issues."
SS-NFID-013	Medium	Partially Fixed	215db061cea5860decbe2f3da91a21bfa1b71a46 21950b365a511342a3913987c66ee4c9fdbd9839 73ebb12c74ed300c376a744a0a24c311980f5af9	Auditor notes: There are much better instructions on how to build and develop, however these instructions are still missing how to make sure that the code matches the proposed canister upgrades through reproducible builds. See https://internetcomputer.org/docs/current/developer-docs/smart-contracts/best-practices/reproducible-builds

Detailed statuses

Finding	Severity	Status	Commit	Notes
SS-NFID-014	Medium	Partially Fixed	99b712bde5b1c9182d2a7974608b88d93f8e6652 88f1fedc9eb1fb025bad4ad1a693d8f0215a8435	Developer notes: “Logging data can’t be persisted with query calls, and for update calls we’ve chosen instead to trap so that state can be safely rolled back. The frontend will handle the trap more reliably and usefully for users. We tried using CanisterGeek but we decided against it after learning of the reasons listed above.” Auditor notes: While it’s true that logs can’t be persisted on query calls, they could be persisted on update calls, however that is incompatible with the strategy of trapping/panicking since that would mean the logs would be rolled back along with the rest of the state. However logs would persist if a strategy of returning errors instead of trapping/ panicking would be adopted.
SS-NFID-015	Medium	Fixed	bc3dd8b71c296b2958f9d81f0c379cdc4997c8aa	Auditor notes: Public canister methods now have meaningful comments.
SS-NFID-016	Medium	Fixed	2b42efe23aea02ec19217e429cd1bc8a13f2a89b	Auditor notes: recover_account was removed
SS-NFID-017	Medium	Fixed	99b712bde5b1c9182d2a7974608b88d93f8e6652 88f1fedc9eb1fb025bad4ad1a693d8f0215a8435	Auditor notes: unwrap s were changed to expect s with meaningful error messages.
SS-NFID-018	Low	Fixed	2b42efe23aea02ec19217e429cd1bc8a13f2a89b	Auditor notes: recover_account was removed
SS-NFID-019	Low	Fixed	f2ee47e309d64e49713e5455b52a4ec5d364ff6f	Auditor notes: spurious async was removed

Detailed statuses

Finding	Severity	Status	Commit	Notes
SS-NFID-020	Low	Fixed	215db061cea5860decbe2f3da91a21bfa1b71a46 21950b365a511342a3913987c66ee4c9fdbd9839 73ebb12c74ed300c376a744a0a24c311980f5af9	Auditor notes: There are now clear instructions on how to run the tests.
SS-NFID-021	Low	Fixed	1b1779c8047419faad90a1e01c5f891299968970	Auditor notes: spurious file was removed
SS-NFID-022	Low	Fixed	2b42efe23aea02ec19217e429cd1bc8a13f2a89b	Auditor notes: all of recover_account was removed
SS-NFID-023	Medium	Fixed	215db061cea5860decbe2f3da91a21bfa1b71a46 21950b365a511342a3913987c66ee4c9fdbd9839 73ebb12c74ed300c376a744a0a24c311980f5af9	Auditor notes: There are now build instructions.
SS-NFID-024	Medium	Not Fixed	-	Developer notes: “While we agree it is best practice to add comments and appreciate the focus on documentation, we don’t consider a lack of comments a security issue as it can’t break anything and would like to add them at a time when we have weeks to dedicate to the task.”
SS-NFID-025	Medium	Fixed	af5946b892781a03520242dba627a91915753557 63152a06bebf6d48ce10d6446e2606fea6cb65e3	Auditor notes: The NodeJS version has been made consistent both in the server and the client
SS-NFID-026	Medium	Not Fixed	-	Developer notes: “While we agree it is best practice to add comments and appreciate the focus on documentation, we don’t consider a lack of comments a security issue as it can’t break anything and would like to add them at a time when we have weeks to dedicate to the task.”
SS-NFID-027	Medium	Fixed	d84af8ba27e8669ca9c444f551446eb6731608a5	Auditor notes: Dfx version was upgraded to 0.24.1 and a note was made in the README
SS-NFID-028	High	Fixed	6b0d83dceb8bad1e1fa116bd0ccab1e433cb6a1e	Auditor notes: The IM canister now has client separation between admin duties (to be given to an SNS governance canister) that, currently, are to control configuration change; and operator duties, which include backups.

Detailed statuses

Finding	Severity	Status	Commit	Notes
SS-NFID-029	High	Fixed	6b0d83dceb8bad1e1fa116bd0ccab1e433cb6a1e	Auditor notes: The method get_all_accounts_json was made accessible to the operator role only and add_all_accounts_json was removed. This is a great practice sense as accounts can be continuously backed up by operator but a canister upgrade (mediated by governance) would be necessary to actually replace all accounts, in case something went wrong.
SS-NFID-030	Medium	Fixed	6b0d83dceb8bad1e1fa116bd0ccab1e433cb6a1e	Auditor notes: The get_config method is now open to everyone.


Solidstate