Security Audit Report

Abstract Account Contract 0.26.0

v1.0

December 31, 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security GmbH has been engaged by Abstract Money Pte. Ltd. to perform a security audit of Abstract Account Contract.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/AbstractSDK/abstract/tree/main/framework/contracts/account
Commit	`a6472295406b21f06f67a8026df4bf2348f07f84`
Scope	In scope only `framework/contracts/account` directory. The remaining codebase, including critical dependencies like `abstract_std` and `abstract_sdk`, was not reviewed.
Fixes verified at commit	`1871ad0aa16fb87fd0cb630862a8a5ef09436ac9` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Abstract Accounts are programmable smart-contract wallets that hold funds for users and/or applications while exposing a set of programmable endpoints that can be used to configure and interact with the account.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending**, **Acknowledged**, **Partially Resolved**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium	While generally readable, the code contains some dead code and could benefit from more in-line documentation to clarify complex logic and design choices.
Level of documentation	Medium-High	The existing documentation is extensive and covers most of the codebase. However, some features like governance types, lack comprehensive documentation.
Test coverage	Medium-High	`85.41%` line coverage

Summary of Findings

No	Description	Severity	Status
1	Missing sender validation allows anyone to add and remove authenticators	Critical	Resolved
2	Concurrent `admin_execute` calls may lead to unexpected behavior	Minor	Resolved
3	Potential panic during module installation	Minor	Resolved
4	Old adapter contract is a single point of failure during adapter migration	Minor	Acknowledged
5	Registry contract is a single point of failure in the module installation process	Minor	Acknowledged
6	Incorrect interface generated for `AdminExecute` and `AdminExecuteOnModule`	Informational	Resolved
7	Silent failures when unregistering non-existent sub-accounts	Informational	Resolved
8	Missing documentation about suspension status initialization in XION migration	Informational	Resolved
9	Misleading success response for non-xion auth operations	Informational	Resolved
10	Insufficient validation of input URLs	Informational	Acknowledged
11	Dead code in the `migration` function	Informational	Resolved
12	Miscellaneous comments	Informational	Resolved

Detailed Findings

1. Missing sender validation allows anyone to add and remove authenticators

Severity: Critical

In `framework/contracts/account/src/contract.rs:356-360`, the `AddAuthMethod` and `RemoveAuthMethod` messages do not implement any validation to ensure that the caller is authorized. This allows anyone to add and remove authenticators when the contract is deployed on the XION chain, as seen in `framework/contracts/account/src/contract.rs:158-172`.

Consequently, attackers can add malicious authenticators to an abstract account to gain control of it, including access to the funds in the contract.

Recommendation

We recommend adding validation to ensure that the caller is authorized to update authenticators.

Status: Resolved

2. Concurrent `admin_execute` calls may lead to unexpected behavior

Severity: Minor

In `framework/contracts/account/src/execution.rs:87-107`, the `admin_execute` function saves the target address in `CALLING_TO_AS_ADMIN` and creates a submessage with `reply_on_success`. If the executed message triggers another `admin_execute` before the first reply is processed, it will overwrite the `CALLING_TO_AS_ADMIN` state. This could cause the reply handler to process the first reply with the wrong address context, potentially leading to incorrect state management or security issues. We classify this issue as minor because the function is restricted to admin calls, which mitigates the potential for abuse.

Recommendation

We recommend returning an error if the `CALLING_TO_AS_ADMIN` state is already set to prevent concurrent executions and ensure proper reply handling.

Status: Resolved

3. Potential panic during module installation

Severity: Minor

In `framework/contracts/account/src/modules.rs:133`, the `_install_modules` function, when handling `ModuleReference::App` or `ModuleReference::Standalone` cases, directly unwraps `init_msg` without validating its presence. This could lead to a panic during module installation if `init_msg` is `None`.

Recommendation

We recommend adding validation at the entry point of `_install_modules` to ensure that `init_msg` is `Some` before proceeding with the installation.

Status: Resolved

4. Old adapter contract is a single point of failure during adapter migration

Severity: Minor

In `framework/contracts/account/src/modules/migration.rs:250-253`, the `replace_adapter` function relies on querying the old adapter contract for authorized addresses. This introduces a dependency on an external contract. If the old adapter contract is compromised or malfunctioning, it could return manipulated data, leading to unauthorized addresses being transferred to the new adapter or legitimate addresses being omitted from the migration.

Recommendation

We recommend maintaining an independent storage of authorized addresses within the Account contract to reduce reliance on external contract state during migrations. This improves the security and resilience of the adapter replacement process.

Status: Acknowledged

5. Registry contract is a single point of failure in the module installation process

Severity: Minor

In `framework/contracts/account/src/modules.rs:74-77`, the `_install_modules` function relies entirely on the Registry contract to validate and provide module configurations through `query_modules_configs`. If the Registry contract is compromised, it could return malicious module configurations, leading to the installation of

unauthorized or malicious modules on user accounts. This centralization of trust creates a single point of failure in the module installation process.

Recommendation

We recommend implementing additional validation layers such as allowing accounts to maintain their own whitelist of trusted modules or implementing additional safeguards like a multi-signature approval process for module installations from the Registry.

Status: Acknowledged

6. Incorrect interface generated for `AdminExecute` and `AdminExecuteOnModule`

Severity: Informational

In `framework/contracts/account/src/execution.rs:101`, the `admin_execute` function accepts `info.funds` but the `#[cw_orch(payable)]` macro is missing for the `AdminExecute` and `AdminExecuteOnModule` messages. This will cause the interface generated by `cw-orchestrator` to incorrectly disallow sending funds with these messages, decreasing user experience when integrating with other applications. In contrast, the `ExecuteWithData` message correctly includes the macro in `framework/packages/abstract-std/src/account.rs:140`.

Recommendation

We recommend adding the `#[cw_orch(payable)]` macro to the `AdminExecute` and `AdminExecuteOnModule` messages to ensure the generated interface correctly allows funds to be sent.

Status: Resolved

7. Silent failures when unregistering non-existent sub-accounts

Severity: Informational

In `framework/contracts/account/src/sub_account.rs:116-117`, the `unregister_sub_account` function silently fails when attempting to unregister a non-existent account. The `account.is_some_and()` check returns `AccountError::SubAccountRemovalFailed` without distinguishing between an unauthorized removal attempt and a non-existent account. This makes it difficult for callers to understand whether the failure was due to a non-existent account or an unauthorized removal attempt.

Recommendation

We recommend adding explicit validation for account existence and returning a specific error variant when attempting to unregister a non-existent account.

Status: Resolved

8. Missing documentation about suspension status initialization in XION migration

Severity: Informational

In `framework/contracts/account/src/migrate.rs:86`, during migration from a XION account, the function initializes `SUSPENSION_STATUS` as `false` without explicit documentation of this behavior. While this is correct, since XION accounts do not have this flag, it could benefit from clearer documentation.

Recommendation

We recommend adding documentation to explicitly state that `SUSPENSION_STATUS` is initialized as `false` when migrating from XION accounts and consider wrapping this initialization in a `#[cfg(feature = "xion")]` block to make the XION-specific nature of this part more apparent.

Status: Resolved

9. Misleading success response for non-xion auth operations

Severity: Informational

In `framework/contracts/account/src/execution.rs:119-132` and `134-143`, the `add_auth_method` and `remove_auth_method` functions return success responses, with actions "add_auth" and "remove_auth" respectively, even when the "xion" feature is not enabled, and no authentication method is added. This could mislead users or applications into believing the operation was successful.

Recommendation

We recommend either returning an empty response or explicitly indicating that no authentication method was added or removed because the "xion" feature is disabled.

Status: Resolved

10. Insufficient validation of input URLs

Severity: Informational

In `framework/contracts/account/src/contract.rs:110`, the `validate_link` function uses the primitives defined in `framework/packages/abstract-std/src/objects/validation/verifiers.rs:17-37` for input validation of the `link` parameter. However, these primitives do not sufficiently validate URLs, allowing malformed or potentially malicious URLs to pass, such as:

- `http://http://evil.com/`
- `http://////////////`

Recommendation

We recommend enhancing the validation logic to be more robust against malformed inputs and aligning with URL validation best practices.

Status: Acknowledged

11. Dead code in the `migration` function

Severity: Informational

In `framework/contracts/account/src/migrate.rs`, there are two instances of dead code that will never be executed:

- The `has_info()` check in line `74` will always return `false` since all `account_info` fields are explicitly set to `None` in lines `68-72`. As a result, the subsequent `INFO.save()` call will never execute.
- The `GovernanceDetails::SubAccount` match in line `100` will never be reached because the `initialize_owner` function forces the `AbstractAccount` governance type, making `cw_gov_owner.owner` always be `AbstractAccount`.

Recommendation

We recommend removing both blocks of dead code to improve code clarity and maintainability. If these code paths are intended for future use, they should be documented as such.

Status: Resolved

12. Miscellaneous comments

Severity: Informational

Miscellaneous recommendations can be found below.

Recommendation

The following are some recommendations to improve the overall code quality and readability:

- **Incomplete documentation:** The current documentation on governance types is incomplete and potentially misleading. It only mentions Monarchy and Multi-signature, but should be updated to include NFT-based ownership, and AbstractAccount ownership (for the “xion” feature). We recommend updating the documentation to include all governance types to avoid confusion.
- **Inconsistent spelling:** In `framework/contracts/account/src/error.rs:55`, "Upgradable" is spelled inconsistently. Use either "Upgradable" or "Upgradeable" throughout. We recommend correcting the spelling for consistency.
- **Uninformative error message:** In `framework/contracts/account/src/error.rs:109-113`, the error message `AbsAccInvalidAddr` can be improved:
    - Error fields (`abstract_account`, `contract`) are not used in the message. We recommend including the `abstract_account` and `contract` variables in the error message to display the actual addresses that did not match.
    - We recommend correcting "don't" to "doesn't".
    - Inconsistent abbreviation: `AbsAcc` is used for Abstract Account. Use `AbstractAccount` consistently. We recommend renaming `AbsAccInvalidAddr` to `AbstractAccountInvalidAddress` to improve readability.
- **Redundant `abstract_code_id` query**: In `framework/contracts/account/src/migrate.rs:53-54` and `116-117`, the `migrate` function queries the `abstract_code_id` variable twice: first for creating the registry contract and then again for installing modules, even though the value has not changed. This redundant query introduces unnecessary gas costs and complexity. We recommend storing the initially queried `abstract_code_id` in a variable and reusing it throughout the function instead of querying it multiple times.

Status: Resolved