Audit Report

XION and Burnt Contract Updates and Account Abstraction

v1.1

September 22, 2023

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 IS ADDRESSED EXCLUSIVELY TO THE CLIENT. THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THE CLIENT OR THIRD PARTIES.

This audit has been performed by

Oak Security

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

Introduction

Purpose of This Report

Oak Security has been engaged by Thames Brook Associates, LLC to perform a security audit of XION and Burnt Updates and Account Abstraction.

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 targets:

Repository	https://github.com/burnt-labs/xion
Reference	This repository is referred to with the prefix `xion:` in all paths below.
Commit	`3120374e88dbed49061c4794fe32cbfdb4252ab5`
Scope	Only changes since our last audit, which was performed on commit `432e5d73e8bfc0e88de22dda206f405542ee7c91`, were in scope.

Repository	https://github.com/burnt-labs/burnt-cw-std
Reference	This repository is referred to with the prefix `cw-std:` in all paths below.

Commit	`0fe411e762d22384bb00f9304cac31585239c805`
Scope	Only changes since our last audit, which was performed on commit `432e5d73e8bfc0e88de22dda206f405542ee7c91`, were in scope.

Repository	https://github.com/burnt-labs/burnt-cw-hubs
Reference	This repository is referred to with the prefix `cw-hubs:` in all paths below.
Commit	`3b75302aead752e86123fd9f792b5f9a22c8d8a6`
Scope	Only changes since our last audit, which has been performed on commit `f840b6b09fe6227e11cf06007d60546cf641c57f`, were in scope.

Repository	https://github.com/larry0x/abstract-account
Reference	This repository is referred to with the prefix `abstract-account:` in all paths below.
Commit	`2ea462db913fff198900ad94ad27d533ccdd681b`
Scope	Only the Cosmos SDK module in the `x/abstractaccount` directory and its imports from this repository were in scope of this audit.

Repository	https://github.com/burnt-labs/contracts
Reference	This repository is referred to with the prefix `account-contract:` in all paths below.
Commit	`574395b76cac43ed13386eb180250a8de50a9f2f`
Scope	All code was in scope.

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

The audit focuses on functionality tied to the CosmosSDK app chain XION, including its associated hub and seat contracts. Additionally, it covers a suite of XION-specific CosmWasm standard libraries that are tailored to support smart contract developers in building contracts on the XION app chain. These items have previously been audited by Oak Security so the scope of this audit covers the changes implemented since the commit hashes mentioned above.

Additionally, this audit covers functionality to facilitate account abstraction. This includes the integration of both a Cosmos SDK module and a corresponding CosmWasm smart contract. Specifically, the abstractaccount module empowers smart contract accounts with the ability to initiate transactions.

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**, 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	The code in scope of this audit showed a relatively high complexity due to components that involved modified ante decorators, and signature validation.
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	Most of the components were well documented with provided readme documents. The abstractaccount module was very well documented.
Test coverage	Medium	cw-std: `43.50%` coverage, `231/531` lines covered cw-hubs: Tests failing due to outdated dependency account-contract: `16.44%` coverage, `24/146` lines covered abstract-account: • `types/account.go: 11.5%` • `types/genesis.go: 0%` • `types/tx.go: 0%` • `module.go: 47.4%` xion: • `burnt-labs/xion/app: 64.1%` • `burnt-labs/xion/x/globalfee/ante: 29.9%` • `burnt-labs/xion/x/globalfee/ante/antetest: 86.1%` • `burnt-labs/xion/x/globalfee/types: 6.0%` • `burnt-labs/xion/x/mint: 38.6%` • `burnt-labs/xion/x/mint/client/cli: 76.9%`

		• burnt-labs/xion/x/mint/keeper: 89.7% • burnt-labs/xion/x/mint/simulation: 97.8% • burnt-labs/xion/x/mint/types: 1.8% • burnt-labs/xion/x/xion/client/cli: 82.0%

Summary of Findings

No	Description	Severity	Status
1	`try_list` poses centralization threat	Major	Resolved
2	`MaxCoins` function could potentially return an unexpected set of fees	Minor	Resolved
3	`buy_item` does not properly validate sent funds	Minor	Resolved
4	Incorrect gas limit used in `PostHandle` function	Minor	Resolved
5	Missing Genesis validation in the `InitGenesis` function	Minor	Resolved
6	Unused code relating to the `Redeemable` functionality	Informational	Resolved
7	Missing check for `sig_bytes` length in `verify` function	Informational	Resolved
8	Improve credentials processing in `before_tx`	Informational	Resolved
9	Duplicate validations are inefficient	Informational	Resolved
10	Usage of the deprecated function `EmitEvents`	Informational	Resolved
11	Missing validation of parameters	Informational	Resolved
12	Missing `params` validation in `SetParams`	Informational	Resolved
13	Outdated dependencies in `abstractaccount` module	Informational	Partially resolved
14	Contracts should implement a two-step ownership transfer	Informational	Acknowledged
15	Misspelled error	Informational	Resolved
16	Unused errors	Informational	Resolved
17	Unused functions	Informational	Resolved
18	Execute messages of allowable contract do not emit any attributes	Informational	Resolved
19	Outstanding TODO comments in codebase	Informational	Resolved

Detailed Findings

1. `try_list` poses centralization risk

Severity: Major

The `try_list` function in `sellable/src/execute.rs:212` allows the `ownable.owner` to list any token. While this is a privileged account, it allows anyone controlling this account to list tokens that belong to other accounts. If the `ownable.owner` account is compromised, it can create a situation where a user's tokens may be listed without the owner's consent at lower than market value prices, effectively allowing them to be stolen. This also introduces a discrepancy where the `ownable.owner` can list tokens, but the `listed_token.owner` is the only address that can delist tokens.

Recommendation

We recommend only allowing the token owner to be able to list and delist their own token.

Status: Resolved

2. `MaxCoins` function could potentially return an unexpected set of fees

Severity: Minor

The `globalfee` module implements a combination of global and local minimum gas prices, similar to Gaia's implementation. The list of `localFees` is expected to initiate and enforce a higher transaction fee requirement, if the amounts are all greater than the `globalFees` amounts, as a spam prevention measure. The function `GetTxFeeRequired` in `xion:x/globalfee/ante/fee.go:70` retrieves the global and local fees and returns the combined fee requirement using the `MaxCoins` function in line `88`. However, the function implementation can return unexpected results in the following scenarios:

- If the local fees include denoms not present in the global fees, the transaction fee requirement will be set to the local fees, potentially including unwanted fee denominations.
- If the local fees include denoms not present in the global fees, the transaction fee requirement will be set to the local fees, potentially including lower fee amounts for denominations present in the global fees.
- If the local fees include at least one fee amount that is equal to or greater than the equivalent global fee, the transaction fee requirement will be set to the local fees, potentially including lower fee amounts for other denominations present in the global fees.

Recommendation

We recommend using the `CombinedFeeRequirement` function in `xion:x/globalfee/ante/fee_utils.go:27`, instead of the `MaxCoins` function.

Status: Resolved

3. `buy_item` does not properly validate sent funds

Severity: Minor

The `buy_item` function in `cw-std:sales/src/execute.rs:136` contains a logic error that does not properly validate that the funds provided in the message are greater than or equal to the price of the item being bought. While this issue has the potential to have a severe impact on the balance of the contract, it is not directly exploitable under normal conditions due to the fact that the contract is not intended to hold a balance.

Since the contract does not hold a balance, the bank message that performs the funds transfer to the item owner will error due to insufficient funds. If the contract was extended in the future though to hold funds, or if funds are accidentally sent to the contract, an attacker could use the `buy_item` function to steal these funds by listing their own item for a large sum and supplying a very small amount of funds so that they will receive the excess balance.

Recommendation

We recommend updating the value in line `136` to compare the `info.funds` amount to the sales price of the item.

Status: Resolved

4. Incorrect gas limit used in `PostHandle` function

Severity: Minor

In the `PostHandle` function in `abstract-account:x/abstractaccount/ante.go:180`, the `sudoWithGasLimit` function is called with `params.MaxGasBefore` as the maximum gas parameter. Instead, the maximum amount of gas that can be consumed by the contract call in the `after_tx` decorator should be set to `MaxGasAfter`.

Recommendation

We recommend updating `sudoWithGasLimit`'s `maxGas` parameter from `MaxGasBefore` to `MaxGasAfter` when it is called in the `PostHandle` function.

Status: Resolved

5. Missing Genesis validation in the `InitGenesis` function

Severity: Minor

In `abstract-account:x/abstractaccount/module.go:98-103`, the genesis state is created from the supplied input data during genesis initialization. However, the `Validate` function is not called for the created type. Consequently, the message is not validated, potentially leading to an invalid genesis state.

Recommendation

We recommend validating the input genesis using the `Validate` function.

Status: Resolved

6. Unused code relating to the `Redeemable` functionality

Severity: Informational

While the `Redeemable` functionality has been removed, messages relating to the instantiation and querying of redemption of tokens still exist in the codebase in `redeemable::InstantiateMsg` in `cw-hubs:seat/src/msg.rs:14` and `redeemable::QueryMsg` in `cw-hubs:seat/src/msg.rs:41`.

Recommendation

We recommend removing the messages and related unused code, as well as updating the documentation to reflect the removal of the functionality.

Status: Resolved

7. Missing check for `sig_bytes` length in `verify` function

Severity: Informational

In `account-contract:account/src/eth_crypto.rs:37`, there is no validation that the length of `sig_bytes` equals 65 bytes. If the `verify` function is called with `sig_bytes` having a length of less than 65 bytes, it will cause the function to panic. Panics go against best practices since they do not provide error messages to the caller which can help to understand and resolve the cause of the error.

Recommendation

We recommend validating the length of the `sig_byte` and returning a descriptive error if the validation fails.

Status: Resolved

8. Improve credentials processing in `before_tx`

Severity: Informational

In `account-contract:account/src/execute.rs:38-54`, the credential processing is implemented as follows: Check the length of the credential bytes, if the length is less than 1, return an error. The implementation then checks the `cred_byte` (the first byte) and derives the signature algorithm. All other bytes form `sig_bytes` and are interpreted as signature bytes. As a result, the current implementation accepts `cred_bytes` of arbitrary length. It is a widespread implementation practice to read the exact number of signature bytes because the length of the signature is known prior.

Recommendation

We recommend improving the implementation in the following way:

1. In `account-contract:account/src/execute.rs:39`, ensure that the length of `cred_bytes` is at least 65 bytes: 1 byte to encode the signature algorithm and at least 64 bytes to carry the signature bytes.
2. In `account-contract:account/src/execute.rs:48`, validate `cred_index` value equal to one of the known `Authenticator` values.
3. In `account-contract:account/src/execute.rs:49`, if `cred_index` corresponds to `Ed25519` or `Sec256K1`, ensure that the length of `sig_bytes` equals 64 bytes. Otherwise, return an error. If `cred_index` corresponds to `EthWallet`, ensure that the length of `sig_bytes` equals 65 bytes. Otherwise, return an error.

Status: Resolved

9. Duplicate validations are inefficient

Severity: Informational

Across the codebase, instances of duplicate validation have been found, which decrease the efficiency of the code.

Recommendation

We recommend increasing the efficiency of the code by applying the following suggestions:

- Remove the validation of `salt` in `abstract-account:x/abstractaccount/client/tx.go:65` as it is validated already in `abstract-account:x/abstractaccount/types/tx.go:52`.

- Remove the validation of `params` in `abstract-account:x/abstractaccount/keeper/msg_server.go:32` as it is validated already during a stateless check of the `MsgUpdateParams` message.

Status: Resolved

10. Usage of the deprecated function `EmitEvents`

Severity: Informational

The deprecated function `EmitEvents` is used in `abstract-account:x/abstractaccount/keeper/msg_server.go:102`. Using deprecated functions is generally discouraged as they may be removed in future versions of the library, leading to potential compatibility issues.

Recommendation

We recommend using `EmitTypedEvents` instead.

Status: Resolved

11. Missing validation of parameters

Severity: Informational

The `Validate` function in `abstract-account:x/abstractaccount/types/params.go:22` validates only the `maxGasBefore` and `maxGasAfter` parameters, but no validation is performed for `allowAllCodeIDs` and `allowedCodeIDs` parameters. This can allow for a situation where these parameters introduce potential misconfigurations.

Recommendation

We recommend ensuring that `allowedCodeIDs` is not empty when `AllowAllCodeIDs` is `false` and does not include the 0 code ID.

Status: Resolved

12. Missing `params` validation in `SetParams`

Severity: Informational

The `SetParams` function in `abstract-account:x/abstractaccount/keeper/keeper.go:67-78` does not implement validation of the `params` input. `SetParams` is used in the `UpdateParams` function in `abstract-account:x/abstractaccount/keeper/msg_server.go:36`

and in the `InitGenesis` function in `abstract-account:x/abstractaccount/keeper/genesis.go:12`.

While the `UpdateParams` function validates input `params` before passing it to the `SetParams` function, the `InitGenesis` function does not do that. By validating `params` in the `SetParams` function it will improve the overall consistency of input validation and help with future maintainability.

Recommendation

We recommend implementing validation of the `params` input in the `SetParams` function directly.

Status: Resolved

13. Outdated dependencies in `abstractaccount` module

Severity: Informational

Many packages (e.g., `cometbft`, `cosmos-sdk`, `net`, `runtime`) are outdated and have known vulnerabilities fixed in the newest versions.

Examples of known vulnerabilities:

1. GO-2023-1861. Cosmos "Barberry" vulnerability in `github.com/cosmos/cosmos-sdk`.
   Found in: `github.com/cosmos/cosmos-sdk@v0.47.2`.
   Fixed in: `github.com/cosmos/cosmos-sdk@v0.47.3`.
2. GO-2023-1883. Denial of service via OOM in `github.com/cometbft/cometbft`.
   Found in: `github.com/cometbft/cometbft@v0.37.1`.
   Fixed in: `github.com/cometbft/cometbft@v0.37.2`.
3. GO-2023-1878. Insufficient sanitization of Host header in `net/http`.
   Found in: `net/http@go1.19.7`.
   Fixed in: `net/http@go1.20.6`.
4. GO-2023-1840. Unsafe behavior in `setuid/setgid` binaries in runtime. Found in: `runtime@go1.19.7`.
   Fixed in: `runtime@go1.20.5`.
5. GO-2023-1705. Excessive resource consumption in `net/http`, `net/textproto`, and `mime/multipart`.
   Found in: `net/textproto@go1.19.7`.
   Fixed in: `net/textproto@go1.20.3`.

Recommendation

We recommend updating the dependencies and the Go compiler version.

Status: Partially Resolved

14. Contracts should implement a two-step ownership transfer

Severity: Informational

The contracts within the scope of this audit allow the current owner to execute a one-step ownership transfer. While this is common practice, it presents a risk for the ownership of the contract to become lost if the owner transfers ownership to the incorrect address. A two-step ownership transfer will allow the current owner to propose a new owner, and then the account that is proposed as the new owner may call a function that will allow them to claim ownership and actually execute the config update.

Recommendation

We recommend implementing a two-step ownership transfer. The flow can be as follows:

1. The current owner proposes a new owner address that is validated.
2. The new owner account claims ownership, which applies the configuration changes.

Status: Acknowledged

15. Misspelled error

Severity: Informational

In `abstract-account:x/abstractaccount/types/errors.go:8`, the error `ErrNotSingleSignautre` is misspelled.

Recommendation

We recommend updating the error to `ErrNotSingleSignature`.

Status: Resolved

16. Unused errors

Severity: Informational

The codebase contains several errors that are unused. These errors should be removed if they are not intended to be used.

Recommendation

We recommend removing the following errors:

- Remove unused error `NoTokensDefined` in `cw-std:sellable/src/errors.rs:43`.
- Remove unused error `CustomError` in `cw-std:allowable/src/errors.rs:15`.

- Remove unused error `RedeemableError` in `cw-hubs:seat/src/error.rs:22`.
- Remove unused error `Unauthorized` in `cw-hubs:seat/src/error.rs:10`.
- Remove unused error `Serde` in `account-contract:account/src/error.rs:7`.

Status: Resolved

17. Unused functions

Severity: Informational

The codebase contains several functions that are unused. These functions should be removed if they are not intended to be used.

Recommendation

We recommend removing the following functions:

- Remove unused function `checkTxFeeWithValidatorMinGasPrices` in `xion:x/globalfee/ante/validator_tx_fee.go:13`.
- Remove unused function `getTxPriority` in `xion:x/globalfee/ante/validator_tx_fee.go:52`.
- Remove unused function `checkGas` in `xion:x/globalfee/ante/fee_utils.go:125`.
- Remove unused function `checkTxFee` in `xion:x/globalfee/ante/fee_utils.go:133`.

Status: Resolved

18. Execute messages of allowable contract do not emit any attributes

Severity: Informational

The allowable contract's execute messages in `cw-std:allowable/src/lib.rs:95` do not emit any attributes. It is best practice to emit attributes whenever execute messages are handled to allow off-chain listeners such as indexing service observe state changes.

Recommendation

We recommend emitting relevant attributes for the executed messages of the allowable contract.

Status: Resolved

19. Outstanding TODO comments in codebase

Severity: Informational

Across the codebase, instances of TODO comments have been found. These markers often signal areas where additional work or improvements are needed:

- `xion:x/globalfee/queirer.go:34`
- `abstract-account:x/abstractaccount/ante.go:77`

Recommendation

We recommend resolving all TODOs in the codebase before the production release, especially those which may relate to important logic.

Status: Resolved