Security Audit Report

MANTRA Chain v5.0.0

Phases 1-3

v1.3

November 3, 2025

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 MANTRA Ventures Limited to perform a security audit of MANTRA Chain v5.0.0 Updates.

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 in multiple phases on the following targets:

Phase 1

Repository	https://github.com/MANTRA-Chain/mantrachain
Scope	The scope is restricted to the changes applied in the following git diff: • https://github.com/MANTRA-Chain/mantrachain/compare/v1.0.0...1aec224716e1f14dfcde07e7b0d125444f0f8584 with tag v1.0.0 at commit `4e6cfdee274df38885e859201d2860c98d076165`.
Fixes verified at commit	`3f144706adf71b82a8f4b82f8edf7bcd333d5d9a` 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.

Phase 2

Repository	https://github.com/MANTRA-Chain/mantrachain
Commit	`bd389403f26ee41a0d1cf8f57fef79a78106db58`
Scope	The scope is restricted to the changes applied in the following git diff: • https://github.com/MANTRA-Chain/mantrachain/compare/1aec224716e1f14dfcde07e7b0d125444f0f8584...bd389403f26ee41a0d1cf8f57fef79a78106db58
Fixes verified at commit	`fc28f99d9166600c982a2496ff1b084ecdd2c556` 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.

Phase 3

Repository	https://github.com/MANTRA-Chain/mantrachain
Commit	`9bb277a24c6fce7c22a9ecf021bd718dbb1413d1`
Scope	The scope is restricted to the changes applied in the following git diff: • https://github.com/MANTRA-Chain/mantrachain/compare/bd389403f26ee41a0d1cf8f57fef79a78106db58...9bb277a24c6fce7c22a9ecf021bd718dbb1413d1
Fixes verified at commit	`8dc83ce9ba8b1cdb43f33e2ff32c9140acf3ad34` 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

MANTRA Chain is a security-focused, Cosmos-based Layer 1 blockchain designed for tokenizing real-world assets (RWAs) to bridge the gap between traditional and decentralized finance.

This audit covers changes introduced in v5.

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	Low-Medium	-
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	-
Test coverage	Medium-Low	Overall, the unit test coverage is very low, and most of the functionality remains untested by unit tests. However, an extensive end-to-end test suite is available.

Summary of Findings for Phase 1

No	Description	Severity	Status
1	Module accounts are not blocked from receiving bank coins, resulting in a denial-of-service	Critical	Resolved
2	Overwriting the tokenfactory hook call's gas meter with a fixed gas amount allows for a denial-of-service attack	Critical	Resolved
3	Incorrect denom deconstruction may lead to `DenomAuthorityMetadata` query failing	Minor	Resolved
4	`GetBlacklist` query command missing standard query flags	Minor	Resolved
5	Escrow addresses are not removed when an IBC channel is closed	Minor	Resolved
6	Unbounded `QueryBlacklistRequest` query can lead to performance degradation	Minor	Resolved
7	Missing validation for `FeegrantKeeper` in `EVMHandlerOptions` may cause fee grants to fail	Minor	Resolved
8	Events emitted from the `tokenfactory` before-send hook message are discarded	Minor	Resolved
9	Newly created `tokenfactory` denoms have incomplete metadata	Minor	Resolved
10	Hash collision can lead to an ERC-20 token pair being overwritten	Minor	Resolved
11	EVM transaction senders are not checked against the sanction list	Minor	Resolved
12	Sanction list enforcement potential bypass	Minor	Acknowledged
13	Sanction list size can grow and impact ABCI functionality	Minor	Acknowledged
14	`AddBlacklistAccounts` should prevent the blacklisting of critical accounts	Minor	Acknowledged
15	Documentation incorrectly describes sanction messages as singular	Informational	Resolved

16	Incorrect test description may lead to confusion	Informational	Resolved
17	Incorrect `burn_from_address` attribute may lead to confusion	Informational	Resolved
18	Redundant byte conversion when storing escrow addresses	Informational	Resolved
19	Misleading comments may lead to confusion	Informational	Resolved
20	Use of deprecated params module	Informational	Acknowledged
21	Remove todo messages	Informational	Partially Resolved
22	Sanction messages do not emit any events	Informational	Acknowledged

Summary of Findings for Phase 2

No	Description	Severity	Status
23	Incorrect IBC transfer stack description	Informational	Resolved
24	Redundant byte conversion when removing an escrow address	Informational	Resolved
25	Events emitted from the tokenfactory before-send hook message are discarded	Informational	Resolved
26	Unnecessary use of deprecated `AllowUnprotectedTxs` EVM param	Informational	Acknowledged
27	Set `Base` and `Name` denom metadata fields during v5 upgrade	Informational	Acknowledged

Summary of Findings for Phase 3

No	Description	Severity	Status
28	`ProxyGasMeter` allows for uncharged computation when the gas limit is exceeded or an overflow occurs	Major	Resolved
29	Potential overflow in `ProxyGasMeter`'s gas limit calculation	Informational	Resolved
30	Missing `nil` check on `gasMeter` parameter can cause panic	Informational	Acknowledged
31	Potential integer overflow in chain ID parsing	Informational	Acknowledged
32	Modified chain ID persists in context after fallback signature verification	Informational	Resolved
33	`ProxyGasMeter` does not implement `GasConsumedToLimit`	Informational	Acknowledged

Detailed Findings for Phase 1

1. Module accounts are not blocked from receiving bank coins, resulting in a denial-of-service

Severity: Critical

The `BlockedAddresses` function in `app/app.go:1434-1448` is intended to return a list of addresses that are blocked from receiving funds via regular bank coin transfers. Currently, this function only includes EVM precompile addresses in the blocklist and omits Cosmos SDK module accounts. This allows any user to send tokens directly to module accounts.

This presents two denial-of-service vectors:

1. An attacker can send small amounts of many different tokens to a module account. If any on-chain logic, such as in `BeginBlocker` or `EndBlocker`, iterates over all balances of that account using a function like `GetAllBalances`, it could lead to excessive gas consumption and cause a denial-of-service.
2. The `x/tokenfactory` module enables the creation of tokens with transfer hooks that can block transfers. For example, an attacker could create such a token and send it to the fee collector module account. During `BeginBlocker`, the `AllocateMcaTax` function in the `x/tax` module will attempt to transfer a portion of it as a fee to the `McaAddress` treasury. However, this transfer fails due to the malicious transfer hook, causing the entire `BeginBlocker` function to error and block processing to fail, ultimately halting the chain.

Recommendation

We recommend modifying the `BlockedAddresses` function to include all module account addresses from the `maccPerms` map in the returned list of blocked addresses. This will prevent direct bank transfers to module accounts.

Status: Resolved

2. Overwriting the tokenfactory hook call's gas meter with a fixed gas amount allows for a denial-of-service attack

Severity: Critical

The `callBeforeSendListener` function in `x/tokenfactory/keeper/before_send.go:83-141` executes a transfer hook for tokens with a registered CosmWasm contract.

In line `130`, a new gas meter for the contract execution is created with a fixed gas limit of 500,000 gas. However, the amount of gas remaining in the parent context's gas meter is not

checked. This results in the contract execution context being supplied with 500,000 gas, even if the parent context has less gas remaining.

A malicious actor could deploy a CosmWasm contract that, within the `Sudo` hook message, triggers another transfer of the same token. This results in a re-entrant call to the `callBeforeSendListener` function. In each re-entrant call, a new gas meter with the full fixed gas limit is created, effectively resetting the gas limit for the nested execution. This can be exploited to create a nested loop of `Sudo` message calls that will not immediately run out of gas and exhaust the node's resources, leading to a denial-of-service attack. Note that due to the default `DefaultMaxCallDepth = 100` limit of recursive message calls, it will not result in an infinite number of recursions. Once the recursion is unwound, it will error with out-of-gas.

However, it remains a possible DoS vector, as the attacker receives up to 100 * 500,000 = 50 million gas. Since the transaction fee does not account for this additional gas, an attacker could exploit this at scale to receive subsidized computation, potentially leading to resource exhaustion for validators. Additionally, this is executed per coin in the `callBeforeSendListener`, so its impact could be amplified even more.

Recommendation

We recommend checking the gas remaining in the parent gas meter before creating the new gas meter for the `Sudo` call. The new gas meter should be initialized with the lesser of the fixed `BeforeSendHookGasLimit` and the parent's remaining gas.

Status: Resolved

3. Incorrect denom deconstruction may lead to `DenomAuthorityMetadata` query failing

Severity: Minor

In `x/tokenfactory/client/cli/query.go:74-78`, in the `DenomAuthority Metadata` query, the denom is split with the "/" delimiter, and a check is made that there are exactly 3 parts after the split.

However, a tokenfactory subdenom is allowed to have "/" character in it, as seen in `DeconstructDenom` function, meaning that there may be more than 3 parts when splitting with the "/" delimiter.

Recommendation

We recommend using the same logic as in the `BeforeSendHook` query to check the validity of the denom, and extract the `creator` and `subdenom`:

```
creator, subdenom, err := types.DeconstructDenom(denom)
if err != nil {
```

```
        return err
}
```

Status: Resolved

4. `GetBlacklist` query command missing standard query flags

Severity: Minor

In `x/sanction/client/cli/query.go:59-81`, the `GetBlacklist` module query command does not include common query flags that are typically available for CLI commands. This omission forces users to accept default values for important query-related flags such as `--node`, `--output`, `--height`, and other standard query options that enhance usability and functionality.

Recommendation

We recommend adding `[flags]` to `Use` in line `61`, and then `flags.AddQueryFlagsToCmd(cmd)` before returning in line `80`.

Status: Resolved

5. Escrow addresses are not removed when an IBC channel is closed

Severity: Minor

In `x/tokenfactory/ibc_module.go`, the `OnChanOpenAck` and `OnChanOpenConfirm` callbacks store IBC channel escrow addresses. This is done to prevent these addresses from being used in `tokenfactory` bank actions, such as minting and burning tokens for an escrow address.

However, the corresponding channel closing callback, `OnChanCloseConfirm`, does not remove these addresses from storage when a channel is closed. This leads to two negative consequences:

- **State Bloat**: The set of stored escrow addresses grows indefinitely as channels are opened and closed, leading to unnecessary state bloat. An attacker could intentionally create and close many channels to exacerbate this issue.
- **Incorrect Lifecycle Management**: The address of a closed channel remains permanently blocked from `tokenfactory` admin actions. While it is unlikely this address would be needed for another purpose, it is poor hygiene not to clean up resources when their lifecycle ends.

Recommendation

We recommend implementing a mechanism to remove escrow addresses from storage when an IBC channel is closed. This could involve adding a `DeleteEscrowAddress` function to the `tokenfactory` keeper and calling it from the `OnChanCloseConfirm` IBC callback.

Status: Resolved

6. Unbounded `QueryBlacklistRequest` query can lead to performance degradation

Severity: Minor

In `x/sanction/keeper/query.go:21-35`, the `QueryBlacklistRequest` query retrieves a list of all sanctioned accounts. The query iterates through all such accounts and returns them in a single response.

However, the query does not support pagination. If the number of sanctioned accounts becomes large, this query can consume a significant amount of resources, potentially leading to performance degradation or even a denial-of-service scenario on the affected node.

Recommendation

We recommend adding pagination support to the `Blacklist` query.

Status: Resolved

7. Missing validation for `FeegrantKeeper` in `EVMHandlerOptions` may cause fee grants to fail

Severity: Minor

In `app/ante/handler_options.go:34-63`, the `Validate` function checks that all required keepers are defined, but skips `FeegrantKeeper`.

`ExtensionOptionChecker` and `MaxTxGasWanted` don't require validation since they have acceptable default behaviors (`nil` defaults to `rejectExtensionOption`, `zero` is allowed).

Without `FeegrantKeeper` however, transactions using fee grants will fail with "fee grants are not enabled" errors.

Recommendation

We recommend adding a validation check for `FeegrantKeeper` if fee grants should work in MANTRA. Otherwise, we recommend adding a comment explaining that `FeegrantKeeper` is optional.

Status: Resolved

8. Events emitted from the `tokenfactory` before-send hook message are discarded

Severity: Minor

The `callBeforeSendListener` function in `x/tokenfactory/keeper/before_send.go:83-141` calls a configured CosmWasm contract via a `Sudo` message to execute a before-send hook for the coin that is being sent. For this, it creates a child context with a new event manager, which is passed to the `Sudo` call in line `131`.

However, after the contract execution, only the gas consumption is propagated from the child context to the parent context. The events collected in the new event manager are not propagated. This results in the events emitted by the hook contract being discarded, which can prevent off-chain monitoring and indexing of contract interactions.

Recommendation

We recommend propagating the events from the child context's event manager to the parent context's event manager by calling `c.EventManager().EmitEvents(em.Events())`.

Status: Resolved

9. Newly created `tokenfactory` denoms have incomplete metadata

Severity: Minor

In `x/tokenfactory/keeper/createdenom.go:36-45`, the `createDenomAfterValidation` function creates metadata for a new token denomination.

However, the created `banktypes.Metadata` object is not fully populated. The `Symbol` and `Display` fields are not set. According to the Cosmos SDK's `Validate` method for `banktypes.Metadata`, the `Symbol` field is required and cannot be empty. This results in the creation of invalid denomination metadata.

This can cause issues with clients, such as wallets and block explorers, that rely on this metadata to display token information. It can also lead to failures in other modules or during future chain upgrades if metadata validation is enforced.

Recommendation

We recommend populating the `Symbol` and `Display` fields when creating the `banktypes.Metadata`.

Status: Resolved

10. Hash collision can lead to an ERC-20 token pair being overwritten

Severity: Minor

In the `createDenomAfterValidation` function in `x/tokenfactory/keeper/createdenom.go:57-60`, a new ERC-20 token pair is created when a new native token is created. The process involves generating an Ethereum contract address from the native token's denomination string.

A 32-byte SHA256 hash of the denom is calculated and then truncated to 20 bytes to create the contract address. Due to this truncation, there is a theoretical, albeit extremely low, possibility of a hash collision where two different denoms produce the same contract address.

In the event of a collision, the system would overwrite the previously registered token pair. This would sever the link between the original native token and its corresponding ERC-20 contract, potentially disrupting functionality for users interacting with the original token's ERC-20 representation.

Recommendation

We recommend adding a check to verify that no token pair is already registered with the generated contract address `ethContractAddr` before setting the new token pair. If a token pair already exists for that address, the operation should fail to prevent the overwrite.

Status: Resolved

11. EVM transaction senders are not checked against the sanction list

Severity: Minor

EVM transactions, which are of type `MsgEthereumTx`, are routed to a separate EVM ante handler, `newEVMAnteHandler`, implemented in `app/ante/evm.go`. This handler does not include the `SanctionDecorator` that is present for native Cosmos transactions. As a result,

transactions originating from an Ethereum-compatible wallet are not checked against the on-chain sanction list. This prevents adding EVM addresses to the sanction list, only allowing Cosmos SDK addresses to be sanctioned.

Recommendation

We recommend implementing an ante handler decorator specifically for EVM transactions, which checks the sender's address against the sanction list. Additionally, the `MsgAddBlacklistAccount.Validate` and `MsgRemoveBlacklistAccount.Validate` methods should be updated to accept both Bech32 and Ethereum-style 0x addresses.

Status: Resolved

12. Sanction list enforcement potential bypass

Severity: Minor

In `x/sanction/keeper/ante.go:31-39`, the ante handler prevents transactions signed by blocked addresses from being processed by checking if any signer is contained in the sanction list. However, this enforcement mechanism has a fundamental flaw that allows targeted accounts to evade sanctions.

Since sanctions require passing a governance proposal, malicious users have advanced notice of potential sanctions. During the governance voting period (2 days on MANTRA chain), a targeted account can create an `authz` grant to delegate transaction authority to a non-sanctioned address. Once the updated sanction list takes effect, the original account can continue operating through the authorized delegate, effectively bypassing the sanction since the transaction signer would be the non-blocked delegate rather than the sanctioned account.

Additionally, the governance delay provides sufficient time for the targeted account to transfer assets to alternative addresses, further limiting the effectiveness of the sanction mechanism. While staking tokens require an 8-day unbonding period that exceeds the governance timeline, liquid assets can be moved immediately.

Recommendation

We recommend implementing a two-tier sanctioning system to address this timing vulnerability. A trusted multisig authority should be able to impose immediate, temporary sanctions in urgent situations, followed by formal governance confirmation for permanent sanctions. This approach would eliminate the advance warning period that enables bypass strategies while maintaining decentralized oversight through governance.

Additionally, when the multisig authority decides to block an account, the multisig members should ensure that they include any `authz` grantees in the sanction list to prevent a malicious user from bypassing the sanction with pre-granted accounts.

Status: Acknowledged

The client acknowledges that under the current design, sanctioned accounts may have temporary awareness of pending proposals due to the governance process. This is an inherent consequence of ensuring democratic participation and transparency on-chain. No individual or centralised entity possesses the authority to directly modify the deny list.

The client further confirms that this governance-driven approach, while introducing a delay before sanctions become active, aligns with the chain's philosophy of decentralisation and community oversight.

The client intends to deprecate the existing sanction module in a future release and replace it with a redesigned compliance system that maintains decentralisation while introducing optional real-time enforcement capabilities governed by a multi-tier or delegated authority framework.

This change will ensure that urgent compliance actions can be taken without undermining the decentralised nature of the protocol, while still requiring on-chain ratification to preserve transparency and accountability.

13. Sanction list size can grow and impact ABCI functionality

Severity: Minor

In the `AddBlacklistAccounts` function in `x/sanction/keeper/msg_server_blacklist.go:11` there is no limit on the total sanction list size. This is potentially problematic because for each signer in each cosmos transaction a lookup in `k.BlacklistAccounts` is performed in `x/sanction/keeper/ante.go`.

While this is very unlikely because only the module authority can sanction addresses, the message type can provide some guardrails or improve documentation to prevent the sanction list size from increasing over time or unintentionally to the point where it impacts antehandler performance and ultimately causes transaction timeouts.

Recommendation

We recommend adding improved code comments and documentation around the sanction functionality to make governance cognizant of the potential impact of unconstrained sanction list growth.

Status: Acknowledged

The client acknowledges this finding with the note that they have significant control over what is added to the deny list, which makes the lookup time negligible.

14. `AddBlacklistAccounts` should prevent the sanctioning of critical accounts

Severity: Minor

The `AddBlacklistAccounts` function in `x/sanction/keeper/msg_server_blacklist.go:11` does not restrict any addresses from being added to the sanction list. While this is a privileged message, it could still be very impactful if a malicious proposal or an accidental action resulted in a privileged address being sanctioned.

Recommendation

We recommend ensuring that privileged addresses can not be sanctioned and added to the deny list.

Status: Acknowledged

The client acknowledges this finding with the note that they have significant control over what is added to the deny list.

15. Documentation incorrectly describes sanction messages as singular

Severity: Informational

In `x/sanction/README.md:33-57`, the messages `MsgAddBlacklistAccount` and `MsgRemoveBlacklistAccount` are described as allowing adding or removing a single sanctioned account. However, the actual messages are `MsgAddBlacklistAccounts` and `MsgRemoveBlacklistAccounts` (plural), allowing for multiple accounts to be added to or removed from the deny list simultaneously.

The documentation states that message handling can fail if the "sanctioned account is not an account in Bech32 string", but this should reflect that failure occurs when "at least one of the

accounts is not in Bech32 string" since multiple accounts can be processed in a single message.

Recommendation

We recommend updating the `README` to present the correct message names and explain that these messages can add or remove multiple accounts (separated by commas). The failure conditions should also be updated to reflect that validation applies to all accounts in the list, with failure occurring if at least one account is not in proper Bech32 format.

Status: Resolved

16. Incorrect test description may lead to confusion

Severity: Informational

In `x/sanction/types/genesis_test.go:29-36`, a test with an invalid genesis state is described as "valid genesis state". It could lead to confusion as the valid parameter is set to false.

Recommendation

We recommend changing the description to "invalid genesis state"

Status: Resolved

17. Incorrect `burn_from_address` attribute may lead to confusion

Severity: Informational

In `x/tokenfactory/keeper/msg_server.go:139`, the `types.AttributeBurnFrom Address` (`burn_from_address`) attribute is filled with `msg.Sender`. However, tokens are burned from `msg.BurnFromAddress` address, which is not necessarily `msg.Sender`.

Recommendation

We recommend replacing `msg.Sender` with `msg.BurnFromAddress` as the value of the `types.AttributeBurnFromAddress` attribute.

Status: Resolved

18.Redundant byte conversion when storing escrow addresses

Severity: Informational

The `StoreEscrowAddress` function in `x/tokenfactory/keeper/escrow_addresses.go:11-15` is responsible for storing IBC escrow addresses and expects its `address` parameter to be of type `sdk.AccAddress`.

However, in several instances, the `StoreEscrowAddress` function is called with the result of `escrowAddress.Bytes()`. The `escrowAddress` variable is returned by `transfertypes.GetEscrowAddress` and is already of type `sdk.AccAddress`. Calling `Bytes()` on it converts it to a `[]byte` slice. Although `sdk.AccAddress` is an alias for `[]byte`, this conversion is redundant and is not considered a best practice, as it can affect code clarity.

This occurs in the following locations:

1. `x/tokenfactory/ibc_module.go:62`
2. `x/tokenfactory/ibc_module.go:73`
3. `x/tokenfactory/ibcv2_module.go:36`

Recommendation

We recommend removing the redundant `Bytes()` call and passing the `sdk.AccAddress` typed variable directly to the `StoreEscrowAddress` function in all instances.

Status: Resolved

19. Misleading comments may lead to confusion

Severity: Informational

In `x/tokenfactory/keeper/escrow_addresses.go:10`, the comment above `StoreEscrowAddress` function reads "sets the total set of params", which is misleading, as this is not what the function does.

Recommendation

We recommend revising the comment to accurately reflect the function's purpose.

Status: Resolved

20. Use of deprecated params module

Severity: Informational

The `x/params` module is deprecated as of Cosmos SDK v0.53 and will be removed in the next release. Instead, modules are now responsible for managing their own parameters directly, allowing for more efficient storage and better control over state transitions..

Recommendation

We recommend managing parameters directly in the module rather than with the params module.

Status: Acknowledged

21. Remove todo messages

Severity: Informational

It is best practice to remove todo messages from the codebase before releasing into production. Specifically, in `app/app.go:1169` there is a todo comment that specifies `"Implement post handler"`. This todo should be handled as a posthandler is an important component, and if it is missing, it could impact the functionality of the application.

Recommendation

We recommend resolving todos, especially the posthandler comment mentioned above.

Status: Partially Resolved

22. Sanction messages do not emit any events

Severity: Informational

The `RemoveBlacklistAccounts` function in `x/sanction/keeper/msg_server_blacklist.go:62` does not emit any events that can provide helpful information to provide context to sanction operations. In addition, the `AddBlacklistAccounts` function in `x/sanction/keeper/msg_server_blacklist.go:35` does not emit any events.

Recommendation

We recommend emitting events relevant to the state changes that have occurred. It is important to note that when choosing to emit multiple addresses, it should be done so in a deterministic manner by pre-sorting any lists.

Status: Acknowledged

Detailed Findings for Phase 2

23. Incorrect IBC transfer stack description

Severity: Informational

In `app/app.go:767-782`, the IBC transfer stack is described with the list of middleware and its order (from outside to inside).

However, the specific `SendPacket` and `RecvPacket` flow descriptions show incorrect orders in lines `777-781`.

Note: The `SendPacket` flow does not traverse the entire stack and begins with `ratelimit.SendPacket` (set as the `ic4wrapper` in `TransferKeeper`).

Recommendation

We recommend correcting the flow order for:

- `SendPacket: transfer.SendTransfer -> ratelimit.SendPacket -> channel.SendPacket`
- `RecvPacket: channel.RecvPacket -> ratelimit.OnRecvPacket -> tokenfactory.OnRecvPacket -> callbacks.OnRecvPacket -> erc20.OnRecvPacket -> transfer.OnRecvPacket`

Status: Resolved

24. Redundant byte conversion when removing an escrow address

Severity: Informational

The `RemoveEscrowAddress` function in `x/tokenfactory/keeper/escrow_addresses.go:24-28` is responsible for removing IBC escrow addresses and expects its `address` parameter to be of type `sdk.AccAddress`.

However, in `x/tokenfactory/ibc_module.go:93`, the `RemoveEscrowAddress` function is called with the result of `escrowAddress.Bytes()`. The `escrowAddress` variable is returned by `transfertypes.GetEscrowAddress` and is already of type `sdk.AccAddress`. Calling `Bytes()` on it converts it to a `[]byte` slice. Although `sdk.AccAddress` is an alias for `[]byte`, this conversion is redundant and is not considered a best practice, as it can affect code clarity.

Recommendation

We recommend removing the redundant `Bytes()` call and passing the `sdk.AccAddress` typed variable directly to the `RemoveEscrowAddress` function.

Status: Resolved

25. Events emitted from the tokenfactory before-send hook message are discarded

Severity: Informational

The `callBeforeSendListener` function in `x/tokenfactory/keeper/before_send.go:83-142` calls a configured CosmWasm contract via a `Sudo` message to execute a before-send hook for the coin that is being sent. For this, it creates a child context with a new event manager, which is passed to the `Sudo` call in line `132`.

However, after the contract execution, only the gas consumption is propagated from the child context to the parent context. The events collected in the new event manager are not propagated. This results in the events emitted by the hook contract being discarded, which can prevent off-chain monitoring and indexing of contract interactions.

This finding was already raised in the previous audit and fixed in commit `153b7af1375720c31a5b12faaebf524b441ea910`. However, the currently audited code does not contain the fix.

Recommendation

We recommend including the fix from commit `153b7af1375720c31a5b12faaebf524b441ea910` in the v5 upgrade.

Status: Resolved

26. Unnecessary use of deprecated `AllowUnprotectedTxs` EVM param

Severity: Informational

In `app/upgrades/v5/upgrades.go:110-111`, the EVM param `AllowUnprotectedTxs` is set to `true`, in order to allow unprotected transactions to be run as per ADR 006 (`adr/adr-006-allow-unprotected-tx.md`).

However, in a recent commit in the Cosmos EVM upstream repository, Cosmos Labs removed the `AllowUnprotectedTxs` field from the EVM parameters, which is equivalent to setting it to `true`. Indeed, the behavior is now to always accept unprotected txs at the VM level (it can

still be disabled at the RPC level using the `allow-unprotected-txs` parameter in the JSON-RPC configuration).

Recommendation

We recommend synchronizing the MANTRA EVM fork with the Cosmos EVM upstream repository to include the change in commit [1d8b9e1e3903761ab7a3482aaabfd4438c6b3139](), and removing lines `110-111` in `app/upgrades/v5/upgrades.go`, as well as updating the ADR 006 content accordingly.

Status: Acknowledged

The client acknowledges this finding, noting that they are currently using a fork of the EVM module and will eventually base it on the latest Cosmos EVM, with this parameter deprecated.

27. Set `Base` and `Name` denom metadata fields during v5 upgrade

Severity: Informational

In `x/tokenfactory/keeper/createdenom.go:82-100`, the `UpdateDenomWithERC20` function is used as part of a v5 upgrade to update the metadata for a given token denomination (`denom`). The function retrieves the existing `DenomMetaData` and conditionally sets the `Symbol` and `Display` fields if they are empty.

However, the logic does not set the `Base` and `Name` fields of the metadata. Leaving these fields unset can result in incomplete or inconsistent token metadata within the bank module. This could lead to issues for clients, explorers, or wallets that rely on this on-chain metadata to display token information correctly.

Recommendation

We recommend updating the `UpdateDenomWithERC20` function to also set the `Base` and `Name` fields of the `DenomMetaData` if they are not already populated.

Status: Acknowledged

The client acknowledges this finding, noting that it is for the migration of existing bank metadata to upgrade to version 5. `Symbol` and `Display` are only set as those are needed for the ERC20 precompiles, the rest of the metadata stays unchanged as it was in the past. If they have `Base` and `Name`, they will still have those. Those that do not have `Base` and `Name` have existed as such even before the upgrade and can remain as such without issues.

Detailed Findings for Phase 3

28. `ProxyGasMeter` allows for uncharged computation when the gas limit is exceeded or an overflow occurs

Severity: Major

In `x/tokenfactory/types/proxygasmeter.go:64-75`, the `ConsumeGas` function of the `ProxyGasMeter` consumes gas, as specified by the provided `amount` parameter. It checks if the proxy gas limit has been exceeded before it delegates the gas consumption to the underlying, original gas meter.

If an operation's gas cost exceeds the proxy's limit, the function will panic before `ConsumeGas` is called on line `74`, which consumes the gas from the wrapped gas meter. This can be exploited through the tokenfactory's `before-send` hook, which utilizes the `ProxyGasMeter` for `sudo` calls to CosmWasm contracts. Although the hook is designed to recover from such panics without the transaction failing, the gas consumed by the failed operation is not charged.

A malicious actor can repeatedly call a CosmWasm contract that is designed to fail by exceeding the `ProxyGasMeter` limit. Each failed attempt will consume validator resources, but because the hook's gas consumption is undercharged, the attacker pays fewer fees.

Recommendation

We recommend modifying the `ConsumeGas` function of the `ProxyGasMeter` to consume the gas regardless of the panic.

Status: Resolved

29. Potential overflow in `ProxyGasMeter`'s gas limit calculation

Severity: Informational

In `x/tokenfactory/types/proxygasmeter.go:31`, the `NewProxyGasMeter` function calculates `limit + gasMeter.GasConsumed()` without checking for potential integer overflow. Since both `limit` and `gasMeter.GasConsumed()` are of type `uint64`, their addition could overflow and wrap around to a smaller value.

If an overflow occurs, the resulting limit would be much smaller than intended, potentially causing the `ProxyGasMeter` to incorrectly panic with an out-of-gas error even when sufficient gas is available in the parent meter.

We classify this issue as informational, since `limit` is currently hard-coded to `500,000`, and adding it to gas consumed should not overflow in realistic situations.

Recommendation

We recommend using the existing `addUint64Overflow` function from lines `56-62` to safely perform the addition and handle overflow cases appropriately, either by panicking with a clear error message or returning the original gas meter when overflow would occur.

Status: Resolved

30. Missing `nil` check on `gasMeter` parameter can cause panic

Severity: Informational

In `x/tokenfactory/types/proxygasmeter.go:25`, the `NewProxyGasMeter` function does not validate that the `gasMeter` parameter is not `nil` before calling `gasMeter.GasRemaining` and `gasMeter.GasConsumed`.

If a `nil gasMeter` is passed to this function, the application will panic when attempting to call methods on the `nil` interface.

We classify this issue as informational, since `NewProxyGasMeter` is currently only called from `x/tokenfactory/keeper/before_send.go:127` with a valid `gasMeter`.

Recommendation

We recommend adding a `nil` check at the beginning of the `NewProxyGasMeter` function to validate the input parameter and handle the error with a clear error message.

Status: Acknowledged

The client acknowledges this finding, noting that `ctx.GasMeter()` should never be `nil`, it is initialized in `NewContext`, and it is assumed to be not `nil` everywhere.

31. Potential integer overflow in chain ID parsing

Severity: Informational

In `app/config.go:177`, the `ParseChainID` function uses `strconv.Atoi` to parse the EVM chain ID, then casts it to `uint64`.

Large chain ID values in genesis files could cause overflow and create platform-dependent behavior:

- 32-bit systems: Overflow at `2,147,483,648`
- 64-bit systems: Overflow at `9,223,372,036,854,775,808` (half of `uint64` range)

Recommendation

We recommend using `strconv.ParseUint(matches[2], 10, 64)` instead of `strconv.Atoi` for platform-independent parsing that covers the full `uint64` range.

Status: Acknowledged

The client acknowledges this finding, noting that they will avoid using such a high chain ID because of Cosmos SDK chain IDs' length limits.

32. Modified chain ID persists in context after fallback signature verification

Severity: Informational

In `app/ante/multichainid.go:56`, the `AnteHandle` function within the `MultiChainIDDecorator` passes a modified context to subsequent ante handlers. The decorator is designed to support an alternative chain ID for signature verification. If the initial verification with the primary chain ID fails, it attempts to verify the signature again using a derived chain ID, such as "mantra-evm-1".

If this second verification succeeds, the decorator forwards the context, which now contains the modified chain ID, to the next handlers in the chain. This is an unexpected side effect, as the change to the chain ID is intended only for signature verification within this specific decorator.

While there is no observable negative impact in the current implementation, as subsequent decorators do not seem to use the chain ID from the context, it introduces a potential risk. Future code changes could rely on the chain ID from the context, leading to unexpected behavior if it has been modified.

Recommendation

We recommend restoring the original chain ID to the context after the secondary signature verification is complete, but before calling the next handler in the chain. This would isolate the chain ID modification to this decorator and prevent the modified context from leaking into the rest of the transaction execution flow.

At a minimum, a comment should be added to the `AnteHandle` function to clarify that the context's chain ID may be altered for subsequent handlers.

Status: Resolved

33. `ProxyGasMeter` does not implement `GasConsumedToLimit`

Severity: Informational

In `x/tokenfactory/types/proxygasmeter.go`, the `ProxyGasMeter` wraps an existing `storetypes.GasMeter` to strictly enforce a gas limit for the `TrackBeforeSend` tokenfactory hook.

However, the `ProxyGasMeter` embeds the original `storetypes.GasMeter`, but does not override the `GasConsumedToLimit` method. Consequently, when `GasConsumedToLimit` is invoked on a `ProxyGasMeter` instance, it executes the implementation of the embedded (original) gas meter, potentially resulting in inconsistencies in the calculation of gas consumption.

We classify this finding as informational, since this code path is not reachable in this context and thus has no actual impact.

Recommendation

For completeness, we recommend implementing the `GasConsumedToLimit` method on `ProxyGasMeter` to ensure it correctly reports consumption relative to its own configured limit, rather than falling back to the embedded meter's implementation.

Status: Acknowledged

Threat Model for Tokenfactory Module with `BeforeSendHook` Support

System Assets

- Token Supply Integrity: Accuracy and trustworthiness of total supply, minting, burning.
- Admin Privileges: Access to mint, burn, transfer tokens, and modify metadata.
- BeforeSendHook Contracts: Contracts called before token transfers.
- User Funds: User token balances must remain secure and intact.
- Network Resources: Gas, storage, and processing bandwidth.
- Token Namespace: Canonical structure of denoms used to distinguish legitimate tokens.

Actors

- Token Creators: Have mint/burn/transfer/hook creation/admin authority.
- Token Admins: May differ from original creator but retain permissions.
- Hook Contract Admins: May differ from token admin, can change hook behavior.
- Token Holders: Users interacting with tokens.
- Smart Contracts: Recipients of token transfers, including hooks.
- Module and Privileged Accounts: May be able to receive tokens that may impact their functionality or accounting.
- Malicious Actors: External or internal parties attempting to exploit the system.

Security Assumptions

- Admin keys are kept secure and uncompromised.
- BeforeSendHook contracts are safe and do not include malicious behavior.
- Token admins are acting in good faith.
- Users are aware of token-level risks and permissions.
- Frontends minimize spoofing risk by showing the token denom, not just metadata.
- Modules do not trust non-whitelisted tokenfactory denoms when performing accounting functionality.
- Tokenfactory admins do not implement hooks from contracts with untrusted admins.
- Force-transfer is not used maliciously.

Threat Vectors

1. BeforeSendHook-Related Threats

- Denial of Service (DoS): Malicious or buggy hook contracts can revert or consume excessive gas during transfers, potentially crashing BeginBlocker/EndBlocker operations.
- Reentrancy: though CosmWasm is usually considered safe regarding reentrancy attacks, reentrancy can still occur via hooks calling back into itself or external contracts.
- Unfair Treatment: Hook logic may allow/deny transfers based on address, resulting in censorship, favoritism, or intentionally unexpected behavior.
- Rug Pull: Hook contract could be upgraded at some point to prevent anyone from selling and only allowing buying, by blocking sending the token to a pool address on a DEX, except for the attacker address, allowing to drain the pool. Additionally, the owner of the hook contract can be different than the token admin, a malicious hook contract admin can upgrade the contract to rug an otherwise benign token.

2. Admin & Centralization Threats

- Unlimited Minting / Rug Pull: Admins can mint tokens arbitrarily and dump on DEXs or manipulate markets.
- Unauthorized Transfers: Admins can force transfer tokens between accounts without user consent.
- Metadata Manipulation: Admins can change token name/symbol/description, misleading users.
- Key Compromise: If admin keys are stolen, an attacker can fully control token logic and supply.

3. Denial of Service (DoS) Threats

- Resource Exhaustion: Malicious actors can create many tokens to bloat state and increase GetAllBalances gas costs. Additionally, any areas in ABCI functionality or ante-handlers that perform looping over account tokens will be impacted.
- Transaction Spam: Large volumes of tokenfactory-related transactions combined with malicious/buggy hook contracts can saturate block gas limits and slow/halt the chain.

4. Spoofing and Name Collision Threats

- Token Denom Spoofing: A malicious actor could try to create a subdenom that mimics or closely resembles an existing token to confuse users.

Mitigations & Countermeasures

1. Existing Protections

- Denom Namespacing: Creator's address in denom ensures uniqueness and traceability.
- Token Creation Fees: Discourages spam token creation via economic cost.
- Token Denom Validation: The subdenom must be unique and not collide with an existing denom on-chain. The final denom is computed deterministically as `factory/{creator}/{subdenom}`. The full denom string must pass a strict regex check: `[a-zA-Z][a-zA-Z0-9/:._-]{1,127}`. If a full denom already exists, it cannot be recreated.

2. Recommended Safeguards

- Permissioned Hook Registration: Similar to what Neutron does, not allowing anyone to set the hook contracts themselves, but requiring a request (off-chain) from the contract uploader, allowing the chain to review the request, potentially audit the contract, before allowing it to register the hook.
- Hook Contract Immutability or Chain-Governed Upgrades: If having a permissioned hook contract setup, it can be good to augment it by making sure that the contract cannot be upgraded after registration. Or that the chain is the upgrade authority.
- Admin Education: Admins should be carefully selected and trained in secure key management practices to reduce risk of key compromise or misuse.
- Admin Role Transparency: Document admin powers clearly for users. Consider optional read-only tokens where admin permissions are renounced.
- Admin Action Monitoring: On-chain or off-chain monitoring dashboards to track mint, burn, force-transfer events as well as changes to the token supply.
- Balance Query Hardening: Avoid expensive GetAllBalances operations in BeginBlocker/EndBlocker where possible. Specific whitelisted token balances can be queried instead.
- Subdenom Similarity Detection: Add tooling or governance screening to flag confusingly similar subdenoms (e.g., homoglyph attacks, lookalikes).
- Frontend Denom Display: Frontends should always display the full token denom (e.g., `factory/{creator}/{subdenom}`) in addition to token metadata, to mitigate spoofing and user confusion.