Smart contract vulnerability assessment report

Date: 11 December 2024

Project: Open source dynamic assets (Token/ NFT) generator (CIP68)

Version: 1.0

Created by: **Do Trung Nhan - VTECHCOM**

Disclosure

This report evaluates the security of two smart contracts implemented in the Aiken language (**mint.ak** and **store.ak**) and their supporting module (**utils.ak**) based on the functional logic built into the smart contracts and the functions in the supporting module. The goal is to identify potential vulnerabilities and evaluate the robustness of the contracts.

Disclaimer and Scope

This review is based solely on the smart contract code provided at the time of review. It assumes the code is deployed as described and does not consider external factors such as off-chain dependencies, integration points, or the security of the runtime environment, future feature upgrades, or updates.

Assessment overview

The implementation contract aims to build a dynamic asset creation platform that complies with CIP-68 standards, including main functions such as Mint, burn token, Update, Remove Token metadata.

1. Assessment is performed on each source code file provided with the following main functions:

- **Mint.ak:** Mint and Burn Token
- **Store.ak:** Update and Remove Token metadata
- **Utils.ak:** Manage functions that support signature verification and management of UTXOs

2. Detect common security vulnerabilities by manually reviewing and testing functions.

3. Classify the severity of identified issues.

Assessment components

1. mint.ak: the main function is to process Mint and Burn tokens with the following conditions to ensure the transaction is valid:

a. Mint Token Conditions

- The transaction must contain the author's signature.
- Verify the platform address where the user must pay a platform maintenance fee
- Check the transaction's utxo to ensure that the correct minimum fee is sent to the correct platform address.
- Ensure that the information of the valid Metadata token is issued in the correct CIP-68 structure

in the format cip68.prefix_100, cip68.prefix_222

- Verify that the reference token is sent correctly to the store address to store the text
- The transaction creates enough output utxo as required.

b. Burn Token Conditions

- Verify that only the creator can Burn the token
- Check that the transaction utxo ensures that the minimum fee is sent to the platform address
- Check that the reference token amount and the actual token must be burned in the correct amount not exceeding the existing amount.

The contract is well organized with clear functionality, thorough testing conditions, and ensures CIP-68 standards.

2. store.ak: Handles the permission to update and delete metadata for tokens. Verify the integrity of metadata and authorship with the following constraints:

a. Update

- Author signature must be valid and unaltered
- Metadata must have required fields such as name, image, media_type, author
- Check that the minimum fee is sent to the platform address
- Check the reference token constraint and transaction outputs.

b. Delete

- Author signature authentication
- Minimum transaction fee must be sent to the platform address
- Transaction output is only 2 UTXO

The contract has strict checking and binding conditions on data and transaction structure

3. utils.ak: Provides smart contract support functions with main functions such as:

- Checking the validity of utxo and assets
- Filtering and processing asset lists
- Checking transaction inputs and outputs
- Verifying metadata and signatures.
- Processing asset checking and matching logic

=> The module has a clear structure; the functions are divided into specific and flexible functions with high reusability.

The project's code overview is designed as a module with a clear structure and easy to maintain. Data types and functions are broken down into reusable functions that efficiently manage assets and UTXOs while saving transaction costs to the network.

Severity Classification.

1. **Critical**: High risk of loss of contract or asset.
2. **High:** May result in loss of functionality or security.

3. **Medium:** Potential for disruption of functionality but not immediate.
4. **Low:** Minor issues or deviations from best practice.

Evaluation output:

Based on the processing logic and constraints of the smart contract, it is possible to build test functions that change the constraints in each case to verify, for example, the cases described below can use aiken or pycardano to build unit tests. The following is a table of results evaluation:

1. **Mint token**

NO	Signed	Fee	exchange address	Store address	Metadata	Utxo output number	Expected	Result
1	True	True	True	True	True	>= 4	Pass	Pass
2	False	True	True	True	True	>= 4	False	False
3	True	False	True	True	True	>= 4	False	False
4	True	True	False	True	True	>= 4	False	False
5	True	True	True	False	True	>= 4	False	False
6	True	True	True	True	False	>= 4	False	False
7	True	True	True	True	True	< 4	False	False

2. **Burn tokens.**

NO	Signed	Fee	exchange address	Number token	Author	Expected	Result
1	True	True	True	True	True	Pass	Pass
2	False	True	True	True	True	False	False
3	True	False	True	True	True	False	False
4	True	True	False	True	True	False	False
5	True	True	True	False	True	False	False
6	True	True	True	True	False	False	False

3. **Update token.**

NO	Signed	Fee	exchange address	Metadata	Utxo output number	Expected	Result
1	True	True	True	True	>= 3	Pass	Pass
2	False	True	True	True	>= 3	False	False
3	True	False	True	True	>= 3	False	False
4	True	True	False	True	>= 3	False	False
5	True	True	True	False	>= 3	False	False
6	True	True	True	True	< 3	False	False

4. **Remove token.**

NO	Signed	Fee	exchange address	Metadata	Utxo output number	Expected	Result
1	True	True	True	True	>= 2	Pass	Pass
2	False	True	True	True	>= 2	False	False
3	True	False	True	True	>= 2	False	False
4	True	True	False	True	>= 2	False	False
5	True	True	True	False	>= 2	False	False
6	True	True	True	True	< 2	False	False

Conclusion

With the results achieved, no security holes were found for Mint, Burn, Update, Remove token operations.