Arkadiko Audit

Vaults-v2

November 2023

By CoinFabrik

v202311

Executive Summary

CoinFabrik was asked to audit the contracts for the Arkadiko project.

During this audit we found one critical issue, one high issue, three medium issues and one minor issue.

One medium issue was mitigated and the other five issues were resolved.

Scope

The audited files are from the git repository located at https://github.com/arkadiko-dao/arkadiko/. The audit is based on the commit `cbb0ed52fd06780f3d167e94138a6ad51b44cc44`. Fixes were checked on commit `8efdd55d91a5d3593b3612ab3b0939795ad1469c`.

The scope for this audit includes and is limited to the following files:

- `clarity/contracts/vaults-v2/arkadiko-vaults-data-trait-v1-1.clar`: Trait for the vaults-data contract.
- `clarity/contracts/vaults-v2/arkadiko-vaults-data-v1-1.clar`: Vault registry.
- `clarity/contracts/vaults-v2/arkadiko-vaults-helpers-trait-v1-1.clar`: Trait for the helper contract.
- `clarity/contracts/vaults-v2/arkadiko-vaults-helpers-v1-1.clar`: Helper functions for calculating fees and collateralization ratios.
- `clarity/contracts/vaults-v2/arkadiko-vaults-manager-v1-1.clar`: Contract for liquidating and redeeming vaults.
- `clarity/contracts/vaults-v2/arkadiko-vaults-migration-v1-1.clar`: Contract for migrating data from previous version.
- `clarity/contracts/vaults-v2/arkadiko-vaults-operations-v1-1.clar`: User operations on vaults.
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-active-trait-v1-1.clar`: Trait for pool active contract.
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-active-v1-1.clar`: Contract for storing tokens. Only deposit and withdraw public functions.
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-fees-v1-1.clar`: Vault for storing fees
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-liq-trait-v1-1.clar`: Trait for pool-liq contract.
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-liq-v1-1.clar`: Contract for staking usda.

- `clarity/contracts/vaults-v2/arkadiko-vaults-sorted-trait-v1-1.clar`: Trait for sorted contract.
- `clarity/contracts/vaults-v2/arkadiko-vaults-sorted-v1-1.clar`: Contract for storing vaults in order.
- `clarity/contracts/vaults-v2/arkadiko-vaults-tokens-trait-v1-1.clar`: Traits for the tokens contract.
- `clarity/contracts/vaults-v2/arkadiko-vaults-tokens-v1-1.clar`: Contract which keeps registry of whitelisted tokens.
- `clarity/contracts/wstx-token.clar`: Wrapped STX token.
- `clarity/contracts/arkadiko-oracle-v2-2.clar`: Oracle contract for token prices.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior. Our auditors spent two weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code

effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

ID	Title	Severity	Status
CR-01	Signature Replay in Price Oracle	Critical	Resolved
HI-01	Update Codebase for Mainnet	High	Resolved
ME-01	Rewards Lost because of Race Condition	Medium	Resolved
ME-02	Lost Funds due to Fee Front-Running	Medium	Mitigated
ME-03	Authentication via tx-sender	Medium	Resolved
MI-01	Panicking on Possible Error	Minor	Resolved

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
- **High:** These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and **demand immediate attention**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.

- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit has one of the following statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

CR-01 Signature Replay in Price Oracle

Location:

- `clarity/contracts/arkadiko-oracle-v2-2.clar: 96-113`

Classification:

- CWC-002: Authentication & Authorization Flaws.

Oracle prices are updated with a multi-signature scheme. However, besides validating the signature's content and verifying the signer, the function does not check whether the signatures were already used. Also, while the function ensures prices are not outdated for more than 10 blocks, it does not validate the input is newer than the price already set. Considering this, in a context of high price volatility, an attacker can manipulate the oracle in order to arbitrate the system, extracting value from the system.

Recommendation

Signatures used can be stored in a map buffer to boolean. Then, the contract checks against that map if the signature was already used. Also, with this implementation, there is no need for `check-unique-signatures()`.

Otherwise, the block of the last price update can be stored in order to validate the input is newer than the latest price.

Status
Resolved. Signatures are now stored in a map and the validation function checks against it.

High Severity Issues

HI-01 Update Codebase for Mainnet

Location:
- `clarity/contracts/vaults-v2/arkadiko-vaults-tokens-v1-1: 137, 146, 165, 184, 203`

The codebase has many instances of comment with the text “TODO: update for mainnet”, referring to hardcoded addresses and other environment variables.
These instances should be resolved before deploying to mainnet.

Recommendation
Execute planned modifications for mainnet.

Status
Resolved. Mainnet modifications implemented.

Medium Severity Issues

ME-01 Rewards Lost because of Race Condition

Location:
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-liq-v1-1.clar: 240`

Classification:
- CWC-009: Funds Manipulation - Freezing.

The interaction between the burning of `USDA` tokens and the subsequent adjustment of the `fragments-per-token` variable in the liquidity contract results in lost rewards for the users.

When `USDA` tokens are burnt, causing an increment in `fragments-per-token`, occurs a reduction in the calculated value within the `get-pending-rewards()` function. The issue is manifested when rewards are added to the pool, and subsequently, `USDA` tokens are burnt before users claim their rewards. In this scenario, stakers who delay claiming their rewards experience a decrease in the rewards they receive due to the elevated `fragments-per-token`. Consequently, those rewards not transferred to the staker get locked within the contract.

Recommendation
The rewards for each user should be calculated when they are added to the pool instead of at withdrawal time.

Status
Resolved. A mechanism independent from the `fragments-per-token` variable was implemented.

ME-02 Lost Funds due to Fee Front-Running

Location:
- `clarity/contracts/vaults-v2/arkadiko-vaults-operations-v1-1.clar: 73, 146`

Classification:
- CWC-010: Funds Manipulation - Theft.

Fees are charged when minting USDA through `open-vault()` or `update-vault()`, if the user adds collateral to the vault. The minting fee is set in the function `set-mint-fee()`.

The owner can front-run users to change the fee percentage to 100% and charge users for the total of the value to be minted. As a consequence, the user gets no value from the vault and has to pay for getting back the collateral.

Recommendation
Add a parameter allowing the user to provide a maximum accepted fee and check it against the contract fee.

Otherwise, define a maximum value for the fee that can be set in `set-mint-fee()`.

Status
Mitigated. Maximum minting fee implemented.

ME-03 Authentication via tx-sender

Location:
- `clarity/contracts/wstx-token: 55, 122, 130`
- `clarity/contracts/vaults-v2/arkadiko-vaults-data-v1-1: 81`
- `clarity/contracts/vaults-v2/arkadiko-vaults-tokens-v1-1: 82, 116`
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-active-v1-1: 22`
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-fees-v1-1: 20`
- `clarity/contracts/vaults-v2/arkadiko-vaults-pool-liq-v1-1: 148, 388, 412, 421, 431`
- `clarity/contracts/vaults-v2/arkadiko-vaults-migration-v1-1.clar: 32, 81`

- `clarity/contracts/vaults-v2/arkadiko-vaults-manager-v1-1.clar: 261`
- `clarity/contracts/vaults-v2/arkadiko-vaults-operations-v1-1.clar: 215, 225`

Classification:
- CWC-014: Social Engineering & User-Based Attacks.

The system utilizes `tx-sender` for its authentication processes. This method, while functional, presents latent vulnerabilities, particularly exposing actors within the system to threats known as phishing[1].

Actors could inadvertently activate a malicious contract. Once activated, the deceptive contract can access and initiate certain functions, presenting actions as if they were done by the original actor. This impersonation potential poses risks, depending on the specific function being accessed.

The system only uses them for owner authentication. Therefore, it is less likely to be exploited while using a dedicated account as owner.

Recommendation
It is advisable to switch from using `tx-sender` to `contract-caller` for a more reliable and secure authentication method. Furthermore, introducing a mapping for trusted callers can add an extra layer of security, particularly if the system needs to interact with specific intermediary contracts.

Status
Resolved. Authentication method replaced by `contract-caller`.

Minor Severity Issues

MI-01 Panicking on Possible Error

Location:
- `clarity/contracts/vaults-v2/arkadiko-vaults-tokens-v1-1: 85`

Classification:
- CWC-007: Denial of Service - Error Handling.

Using `unwrap-panic` results in the transaction being finished because of a runtime error when the provided value is an error or a `none`. The runtime error does not allow the caller to handle that error and act in response. Also, this kind of error does not provide any information about the reason for the reverted transaction to the user.

[1] https://www.coinfabrik.com/blog/tx-sender-in-clarity-smart-contracts/

While that form is a convenient method to unwrap values, it should not be used unless it is seemingly impossible to trigger the panic.

In this case, the unwrapping will fail if the token list already has 25 items before calling `set-token()` to add a new one.

Recommendation
Replace `unwrap-panic` for `unwrap!` when there is a flow which might trigger an error or none value.

Status
Resolved. Fixed according to the recommendation.

Enhancements

No enhancements proposed.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

The system parameters are defined by an owner. Valid tokens, fees, prices, and collateralization ratios are set by that role.

Upgrades

The system can be upgraded by deploying new contracts and replacing the previous ones in the dao contract, which whitelists system contracts.

Protocol Assumes Prices Are Updated

Protocol consumes the oracle in order to consult the prices (through the function `fetch-price()`). However, the protocol always assumes those prices are updated. Therefore, the system heavily relies on an off-chain service constantly updating the prices. Otherwise, users will extract value from the system through arbitrage.

Changelog

- 2023-11-29 – Initial report based on commit `cbb0ed52fd06780f3d167e94138a6ad51b44cc44`.
- 2023-11-29 – Reaudit report based on the fixes in commit 8efdd55d91a5d3593b3612ab3b0939795ad1469c.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the Arkadiko project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.