ALEX's Auto Token v3 Audit

May 2024

By CoinFabrik

v202405

Executive Summary

CoinFabrik was asked to audit the contracts for the Auto Token v3.1 project for ALEX.

The AutoALEX token enables auto-staking, which means that the token is automatically harvested and re-staked, allowing holders to earn passive compound interest simply by holding it.

During this audit we found 2 high-severity issues, 2 medium-severity issues and several minor issues. Also, various enhancements were proposed.

The development team created new versions of the files for version 3.2, denoted by the suffix v3-2, while retaining the audited versions associated with version 3.1, which end with v3-1. Fixes were analyzed over the new files.

Additionally, the atALEXv3 and watALEXv3 tokens have been renamed in version 3.2 to LiALEX and vLiALEX, respectively.

Scope

The audited files are from the git repository located at https://github.com/alexgo-io/alex-v1. The audit is based on the commit `ae0bdb4204136735c5207266ebe53b0300ad4cbe`. Fixes were checked on commit `7143851fb230a750992fbf2f79863235edc42f83`.

The scope for this audit includes and is limited to the following 4 files:

- `contracts/auto-token/auto-alex-v3-1.clar`: Implements the Auto ALEX v3 (atALEXv3) token, a rebasing token backed by staked ALEX tokens and their accruing rewards. The main idea behind Auto ALEX is that it auto-compounds by systematically re-staking the rewards. In order to do this, the contract handles staking and claiming processes, interacting with alex-reserve-pool contract as a member on behalf of atALEXv3 token holders. Users can mint atALEXv3 in exchange for ALEX.

- `contracts/auto-token/auto-alex-v3-1-endpoint.clar`: Centralizes the core rebasing and re-staking logic for Auto ALEX and acts as the operational interface for users interested in staking ALEX with Auto ALEX. Operations include: minting and redeeming; claiming and re-staking regular rewards as atALEXv3; upgrading from atALEXv2 to atALEXv3.

- `contracts/auto-token/auto-alex-v3-1-registry.clar`: Acts as a registry for stake and redeem operations, serving as the data counterpart to the aforementioned endpoint contract. It stores data such as redeem request details and tracks which cycles have already been claimed and re-staked.

- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`: Implements the non-rebase version of the atALEXv3 rebase token.

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, tests defining limited use cases, with no documentation. Our auditors spent one week 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
- 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

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
HI-01	Double Conversion In add-to-position() Could Lead to Losses	High	Resolved
HI-02	Incorrect Quantities In *-redeem functions	High	Resolved
ME-01	Authentication For Privileged Calls	Medium	Mitigated
ME-02	Authorization In Reserve Operations	Medium	Acknowledged
MI-01	Misleading SIP-010 Implementation	Minor	Resolved
MI-02	Fixed Functions Implementations Are Misleading	Minor	Resolved
MI-03	Division by Zero in Token to Shares Conversion	Minor	Resolved
MI-04	Sum Over List Could Skip Errors Leading To Unclaimed Funds	Minor	Resolved
MI-05	Arbitrary Decimals Could Be Dangerous	Minor	Resolved
MI-06	Problems With set-decimals()	Minor	Resolved
MI-07	Use Of Named Constant Instead for uint	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

No issues were identified in this category.

High-Severity Issues

HI-01 Double Conversion In add-to-position() Could Lead to Losses

Location:

- `contracts/auto-token/auto-alex-v3-1-endpoint.clar: 148, 178`

The variable `new-supply` (computed as

```
(new-supply (get-tokens-to-shares dx))
```

is used in `auto-token-v3-1-endpoint::add-to-position():L158` when dx should be used for calling `mint-fixed()` which will actually mint

```
(get-tokens-to-shares amount)
```

tokens (now dx is being converted twice to tokens from shares).

Something analogous happens with the `upgrade()` function (`L178`) where `intrinsic-dx` should replace `new-supply`.

Recommendation

Use the correct variables as noted. Moreover, implement tests that assert that values are converted correctly.

Status

Resolved. In `auto-alex-v3-2-endpoint.clar` the `new-supply` variable declaration was removed from both functions, and corrected arguments are now passed to `mint-fixed()`. Also, the functions' payload was updated to output the minted supply of atALEXv3 (now renamed LiALEX) tokens instead of the shares. No tests or documentation were added.

HI-02 Incorrect Quantities In *-redeem functions

Location:

- `contracts/auto-token/auto-alex-v3-1-endpoint.clar`

The function `auto-token-v3-1-endpoint::request-redeem()` calls `transfer-fixed()` using the parameter `shares: amount` , but the function expects an amount in token units. The same thing happens with the other redeem functions (`revoke-redeem()` and `finalize-redeem`()).

Recommendation

Use the correct variables as noted. Moreover, implement tests that assert that values are converted correctly. Finally, we recommend the use of unambiguous variable and parameter names.

Note, however, that the series of function calls involved does the conversion from shares to tokens and back with the caveat that there may be small losses due to arithmetic precision (see EN-05).

Status

Fixed. In `auto-alex-v3-2-endpoint.clar` the function was modified so that token amounts are stored and passed; conversion happens when necessary. No tests or documentation were added.

Medium Severity Issues

ME-01 Authentication For Privileged Calls

Location:

- `contracts/auto-token/auto-alex-v3-1.clar`
- `contracts/auto-token/auto-alex-v3-1-endpoint`
- `contracts/auto-token/auto-alex-v3-1-registry.clar`
- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

Authentication is done via `tx-sender` in `check-is-owner()` function in all of the above contracts. This is probably unnecessarily too general eluding the least privilege principle as the use will be typically direct and using `contract-caller` will suffice.

The authorization for functions `set-contract-owner()` and `set-approved-contract()` in the different contracts should adhere to the least-privilege policy. If we can ensure that the contract-caller necessarily is the owner and will not require a more general permission, make this explicit.

Authentication via `tx-sender` is done with the `check-is-approved()` function of `auto-alex-v3-1` and `auto-alex-v3-1-registry` contracts also. This is the way privileged calls are guarded.

Recommendation

Use the principle of least privilege. Define functions that strictly adhere to this principle for each use case, and use these functions instead of custom combinations of `is-owner()`, `is-approved()` and conjunctions.

Status

Mitigated. Remediation relates to the `auto-alex-v3-2-*.clar` files. The development team stated that they plan to assign the ownership of all of these contracts to the DAO. If deployment will include setting as owner the DAO, which is not a standard user, this would reduce risk for the `is-owner()` checks. For the checks implemented via the `is-approved()` function, the risk exposed in ME-01 remains unchanged.

Developers mentioned they appreciate that a granular control over privileges is better than the current model and will look to improve as part of future contract upgrades.

ME-02 Authorization In Reserve Operations

Location:

- `contracts/auto-token/auto-alex-v3-1.clar`
- `contracts/auto-token/auto-alex-v3-1-endpoint`
- `contracts/auto-token/auto-alex-v3-1-registry.clar`
- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

The function `auto-alex-v3-1::set-reserve()` authorization scheme appears to be too lax, as it requires that the transaction sender is the owner or an approved contract, when it is only called from `rebase()` in the endpoint contract.

The same happens with the functions `set-staked-cycle()`, `set-staked-cycle-shares-to-tokens()`, `set-redeem-request()`, `set-redeem-shares-per-cycle()`, `set-redeem-tokens-per-cycle()` in the registry contract that probably can only be called from a single principal as `contract-caller`.

Again, function `auto-alex-v3-1::transfer()::152` uses `tx-sender` to authenticate the user, allowing phishing attacks to happen via malicious contracts that could entice Auto Token users to call a seemingly innocuous function that calls this transfer function. We recommend asserting just with `contact-caller`, and if the use case necessitates the use of `tx-sender`, then we suggest using an owner-updateable whitelist of trusted contracts as contract callers. Analogous comments apply to `auto-alex-v3-1-wrapped::transfer()`.

Recommendation

Use the principle of least privilege. Define functions that strictly adhere to this principle for the use case.

Status

Acknowledged. The development team mentioned they appreciate that a granular control over privileges is better than the current model and will look to improve as part of future contract upgrades.

Minor Severity Issues

MI-01 Misleading SIP-010 Implementation

Location:

- `contracts/auto-token/auto-alex-v3-1.clar`
- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

Misimplementation of SIP-010 Fungible Token Standard. Tokens `auto-alex-v3-1` and `auto-alex-v3-1-wrapped` do not implement sip-010-trait strictly since the `transfer()` function is implemented using the parameter `(optional (buff 2048))` instead of the optional memo field of 34 bytes. This does not trigger a problem immediately, but can lead to compatibility issues when operating with external DeFi protocols that expect a memo field of 34 bytes.

Similarly, the `token-symbol` variable allocated size is `(string-ascii 10)`, which makes `auto-alex-v3-1::get-symbol()` return `(response (string-ascii 10) uint)` instead of `(response (string-ascii 32)` as the trait dictates.

Recommendation

Make ALEX tokens fully compatible with SIP-010 Fungible Token Standard to avoid compatibility issues. Adding an `impl-trait` line at the beginning of each contract is strongly recommended.

Status

Resolved. Changes have been applied to the `auto-alex-v3-2-*.clar` files. Both size values were updated to match sip-010-trait.

MI-02 Fixed Functions Implementations Are Misleading

Location:

- `contracts/auto-token/auto-alex-v3-1.clar`
- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

The `*-fixed()` functions in token contracts are misleading, as they do not implement the fixed functionality. Although the change would only enter into effect if the owner changed the decimals (via an owner-only call to `set-decimals()`), this could lead to avoidable losses.

Recommendation

Either implement the functionality, or have these functions return an error.

Status

Resolved. Changes have been applied to the `auto-alex-v3-2-*.clar` files. The issue has been effectively resolved by setting the `token-decimals` variable as a constant equal to `u8`. This means fixed precision is the same as the decimal configuration and cannot be changed. Consequently, `*-fixed()` functions invoking their corresponding non-fixed versions is a consistent behaviour with no risks.

MI-03 Division by Zero in Token to Shares Conversion

Location:

- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

Function `auto-alex-v3-1-wrapped::get-tokens-to-shares()` makes a zero check in the numerator of a division when the denominator should be used. Division by zero happens if the balance of atALEX is zero.

Recommendation

Check the denominator for zero.

Status

Resolved. Changes have been applied to the `auto-alex-v3-2-*.clar` files.

MI-04 Sum Over List Could Skip Errors Leading To Unclaimed Funds

Location:

- `contracts/auto-token/auto-alex-v3-1-endpoint.clar`

The function `auto-alex-v3-1-endpoint::sum-claimed()` (used by `claim-and-mint (reward-cycles (list 200 uint))`) does a sum over a list of reward-cycle items using `fold`; when some of these terms include errors, the items they represent are added as zero.

No error or log is thrown. Hence, where one of these failing items represents a positive value, it will be ignored and the values lost (see `alex-reserve-pool::claim-staking-reward-at-cycle()` for context).

Recommendation

We recommend using a pattern analogous to that used in `auto-alex-v3-1::check-err()` or notifying users of the error and allowing them to recover unclaimed rewards .

Status

Resolved. In `auto-alex-v3-2-endpoint.clar` the `sum-claimed()` function within the `auto-alex-v3-1-endpoint` contract was modified in order to explicitly throw an error if any of the reward cycles fail during the claiming process.

MI-05 Arbitrary Decimals Could Be Dangerous

Location:

- `contracts/auto-token/auto-alex-v3-1.clar`
- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

There should be a maximum for `set-decimals()` lest the owner create a DOS, with many operations overflowing, by setting decimals to 10000 might, or setting it too small might create money losses.

Recommendation

Add upper and lower bounds to `set-decimals()` functions in token contracts.

Status

Resolved. In the `auto-alex-v3-2-*.clar` files, the `set-decimals()` function was removed from both token contracts and `token-decimals` was declared as a constant.

MI-06 Problems With set-decimals()

Location:

- `contracts/auto-token/auto-alex-v3-1.clar`
- `contracts/auto-token/auto-alex-v3-1-wrapped.clar`

The `set-decimals()` functionality is used by different contracts independently. Yet, there are some synchronicity requirements that should be carefully considered. First, if one of the decimals variables were to change while the others would not, this could lead to losses. Also, there is a race condition when a user calls `transfer()` and decimals change immediately before without him noticing this.

Recommendation

Consider synchronizing decimal variables in a single function call.

Consider carefully announcing decimal changes to mitigate any risk for users being surprised with decimal changes. Although pausing transfers is a second option, it also implies a negative impact and we discourage following this path.

Status

Resolved. As with MI-05, the issue was addressed in the `auto-alex-v3-2-*.clar` files by removing `set-decimals()` function from both token contracts and declaring `token-decimals` as a constant.

MI-07 Use Of Named Constant Instead for uint

Location:

- `contracts/auto-token/auto-alex-v3-1-endpoint.clar`

In the endpoint contract, the value `u32` is used at several places instead of the named constant `(define-constant redeem-delay-cycles u32)` defined above. If the constant were to change on a contract update, the developer is forced to change every occurrence.

Recommendation

Consider adhering to best practices and using the named constant at the code and defining its value in the contract only once.

Status

Resolved. As recommended, the value `u32` was replaced by a constant in `auto-alex-v3-2-endpoint.clar`. Additionally, `redeem-delay-cycles` constant was renamed to `max-cycles` to improve naming consistency.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

ID	Title	Status
EN-01	Apocryphal SIP-10 trait	Not implemented
EN-02	Dead Code	Implemented
EN-03	Function set-start-cycle() Accepting Arbitrary Values	Not implemented
EN-04	Precision Loss in Double Conversion	Not implemented

ID	Title	Status
EN-05	Unnecessary Functionality in set-staked-cycle()	Not implemented
EN-06	Computation Savings And Precision Gains	Implemented

EN-01 Apocryphal SIP-10 trait

The contract `trait-sip-010.clar` defines a `sip-010-trait` which is a copy of the traditional trait but with additional helper functions for 8-digit notation. There could arise incompatibility problems for contracts used within this ecosystem that assume the traditional implementation and fail to comply with this one.

Recommendation

Use a different name for this trait and specify in the documentation.

Status

Not implemented.

EN-02 Dead Code

The funcion `auto-alex-v3-1-wrapped::check-is-approved()` is not used. Neither `approved-contracts` mapping. There are no privileged functions in `auto-alex-v3-1-wrapped.clar`. Same happens in `auto-alex-v3-1-endpoint.clar`.

Recommendation

Remove dead code.

Status

Implemented.

EN-03 Function set-start-cycle() Accepting Arbitrary Values

The registry function `set-start-cycle()` accepts arbitrary input values when it probably should be called once, or at least with increasing and bounded values. Consider restricting it to prevent errors.

Recommendation

Restrict input as necessary..

Status

Not implemented.

EN-04 Unnecessary Functionality in set-staked-cycle()

Function `set-staked-cycle()` is only used to change a mapping from false to true, but it is not required to map any value from true to false.

Recommendation

Restrict input as necessary..

Status

Not implemented.

EN-05 Precision Loss in Double Conversion

Note that, due to losses in integer division, it happens that `get-token-to-shares()` and `get-shares-to-tokens()` are not the inverse of each other.

Recommendation

Avoid making conversions from tokens to shares and back to tokens, and vice versa by carefully keeping records of quantities.

Status

Not implemented.

EN-06 Computation Savings And Precision Gains

At the endpoint contract in `finalize-redeem()::L228` the consecutive calls of `div-down` and `mul-down`

```
(redeem-tokens (div-down (mul-down (get shares request-details)
```

first divide and then multiply by the constant 1e8. Consider simply using * and `/`.

Recommendation

Replace div-down with / and mul-down with *.

Status

Implemented. More explicitly, the code was modified so that the problem highlighted by this enhancement was eliminated. There is no need to compute redeem-tokens.

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

As described in the authentication and authorization issues (HI-01 and HI-02) there are some functions that have excessive power, and while they should only be called in certain contexts, they can be called by the owner at arbitrary situations, e.g., `set-reserve()` or `set-decimals()`. These capabilities should be restricted, and its uses documented for the users.

About Testing and Documentation

We found that there is no overall documentation for the auto token v3.1, and only a few functions include comments documenting their input. It is important for the underlying protocol and use cases to be documented.

The tests in `tests/auto-alex-v3_tests.ts` only cover three restricted use cases. Some of the issues detected in this report would have been caught with higher coverage testing. We suggest that the team documents and develops tests covering use cases, boundary cases and ensuring that the expected quantities are computed.

Changelog

- 2024-04-24 – Initial report based on commit `ae0bdb4204136735c5207266ebe53b0300ad4cbe`.
- 2024-05-15 – Check fixes on commit `7143851fb230a750992fbf2f79863235edc42f83`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the ALEX Auto Token v3.1/v3.2 project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.