CoinFabrik

Security Audit Report

Paxos

December 2024

V202411

Executive Summary **3**
Scope **3**
Findings **4**
Critical Severity Issues 4
High Severity Issues 4
Medium Severity Issues 5
ME-01 Unbound Instance Storage 5
ME-02 Changing Multiplier 5
Low Severity Issues 6
LO-01 Double Multiplier DOS 6
Enhancements 7
EN-01 No Need For Checked Arithmetic 8
Other Considerations **8**
Balance Tracking 8
Centralization 9
Upgrades 9
Privileged Roles 10
ybs_contract 10
About CoinFabrik **12**
Methodology **12**
Severity Classification **13**
Issue Status **14**
Disclaimer **15**
Changelog **15**

Executive Summary

CoinFabrik was asked to audit the contracts for the **Paxos** project.

During this audit we found two medium severity issues and one low severity issue. Also an enhancement was proposed. The two medium issues were acknowledged, the low severity issue was mitigated and the enhancement was implemented.

Scope

The audited files are from the git repository located at https://github.com/paxosglobal/stellar-paxos-token-contracts-internal.git. The audit is based on the commit `5581ff364341e68e0c09424d37d80218048ce59d`. Fixes were checked on commit `ae682c6c4f145ab24e416ccf54cd19572ab8f8f5`.

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

- `contracts/access_control` directory: Library that checks for privileged roles.
  - `src/lib.rs` file: Library implementation.
- `contracts/address_tag` directory: Library to tag accounts.
  - `src/lib.rs` file: Library implementation.
- `contracts/common_token` directory: Helper library to implement SEP-41 tokens.
  - `src/allowance.rs` file: Utilities to manage token allowances.
  - `src/lib.rs` file: Just module declarations.
  - `src/metadata.rs` file: Utilities to manage the token metadata.
  - `src/roles.rs` file: Constant definitions for privileged roles.
  - `src/storage_types.rs` file: Constants used in persistence.
- `contracts/initializable` directory: Library used to implement initializable contracts.
  - `src/lib.rs` file: Library implementation.
- `contracts/pausable` directory: Library used to implement pausable contracts.
  - `src/lib.rs` file: Library implementation.
- `contracts/ybs_contract` directory: Rebasing token contract.
  - `src/asset_protection.rs` file: Utilities to handle blocked accounts.
  - `src/contract.rs` file: Contract implementation.
  - `src/lib.rs` file: Just module declarations.
  - `src/rebase.rs` file: Utilities to handle rebasing and multipliers.
  - `src/roles.rs` file: Constant definitions for privileged roles specific to this contract.

- `src/shares.rs` file: Utilities to manage shares.

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.

After the audit was completed, the development team pushed the contents of the repository to the https://github.com/paxosglobal/stellar-paxos-token-contracts.git git repository and rewrote its history. The commit this audit was based on has the same contents as commit **`9ddbe5781e8bb154cb5ea73b8d3684fc3ccac1e1`** in this repository. The commit we used to check the fixes has the same contents as commit **`ca5445694c5a5019135c9b7de75e4cbcb8e1a197`**.[1]

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.

Each severity label is detailed in the Severity Classification section. Additionally, the statuses are explained in the Issues Status section.

Id	Title	Severity	Status
ME-01	Unbound Instance Storage	🟧 Medium	**Acknowledged**
ME-02	Changing Multiplier	🟧 Medium	**Acknowledged**
LO-01	Double Multiplier DOS	🟨 Low	**Mitigated**

Critical Severity Issues

No issues found.

High Severity Issues

No issues found.

[1] We checked that the contents are the same by calculating the sha256 hash of each file in the 2 repositories.

Medium Severity Issues

ME-01 Unbound Instance Storage

Location

- `contracts/access_control/src/lib.rs: 116-118`

Classification

- CWE-770: Allocation of Resources Without Limits or Throttling[2]

Description

In the `ybs_contract`, the role information is stored as instance entries. Given that instance entries are capped at 64Kb, this may make the admin unable to grant a role to an account by invoking the `grant_role` function.

Recommendation

Store each role, account pair on its own persistent storage slot.

Status

Acknowledged. The development team acknowledges that it is technically possible to exhaust instance storage to the point where grant_role would be unusable. However, given the current and intended future usage of instance storage for `ybs_contract`, they do not foresee ever hitting this limit. They prefer the ease of TTL management using instance storage over the unlikely event we exhaust instance storage.

ME-02 Changing Multiplier

Classification

- CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')[3]

[2] https://cwe.mitre.org/data/definitions/770.html

[3] https://cwe.mitre.org/data/definitions/362.html

Description

All the operations that receive amounts that need to be converted to shares in the `ybs_contract` (`transfer`, `transfer_from`, `mint`, `burn`, `burn_from`, `unblock_addrs`) are susceptible of being attacked if the attacker can somehow make the operation enter before or after a multiplier is being changed, effectively changing the number of shares being used in the operation.

This leaves EOAs exposed to these attacks, as they cannot check in the same transaction that the accounts balances change as expected.

Recommendation

In order to protect EOAs from these attacks, and given that the SEP-41 API[4] does not support shares, we recommend adding new functions for these operations where the expected multiplier is passed as a parameter and then check that the expected multiplier is the actual multiplier used for the action, while keeping the actual ones.

Also these shortcomings should be documented in the `README.md` file and other places where the API is explained.

Status

Acknowledged. It must be noted that the multiplier can only be increased, making this issue less impactful.

Low Severity Issues

LO-01 Double Multiplier DOS

Location

- `contracts/ybs_contract/src/contract.rs: 175, 294, 390, 395`

Description

The `require_no_pending_multiplier` function, defined in `contracts/ybs_contract/src/rebase.rs:18-26`, checks that there is no pending multiplier change.

```
pub(crate) fn require_no_pending_multiplier(e: &Env) {
    let mult_incr_time: u64 = get_mult_incr_time(e);
```

[4] See https://github.com/stellar/stellar-protocol/blob/master/ecosystem/sep-0041.md

```
    let before_incr_mult: i128 = get_before_multiplier(e);
    let after_incr_mult: i128 = get_after_multiplier(e);

    if e.ledger().timestamp() < mult_incr_time && before_incr_mult !=
after_incr_mult {
        panic!("CannotChangeRebaseSharesWithPendingMultiplier");
    }
}
```

This function is used in the `mint`, `burn` and `burn_from` operations that can be executed by a supply controller and the `unblock_addrs` function that can be executed by an asset protector.

This can be exploited by a rogue rebaser calling the `increase_next_multiplier` function as soon as it is possible or by a rogue rebase admin by calling the `set_next_multiplier` function, which can be called at any time, making the attack more persistent.

The severity of this issue was reduced as the attack requires the control of a privileged account and only affects operations of other privileged accounts.

Recommendation

None of the operations should be blocked by having a different future multiplier. If the intent of the check is to make sure that a specific multiplier is being used, then pass the expected multiplier as a parameter to the function and check that the expected multiplier and the actual multiplier are equal[5].

Status

Mitigated. The development team informed us that all roles are centrally owned by Paxos, with use of offline weighted multisig accounts. In the event a rebaser or rebase admin account is compromised, Paxos would immediately pause the contract and would revoke the compromised role, thus mitigating the attack.

Enhancements

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

Id	Title	Status
EN-01	No Need For Checked Arithmetic	**Implemented**

[5] Also see ME-02 Changing Multiplier

EN-01 No Need For Checked Arithmetic

Description

Through the audited several `checked_*` functions are used for arithmetic calculations even though the `overflow-checks` flag is set to `true` in the `Cargo.toml` file, making them not necessary.

Recommendation

Use standard arithmetic operations instead of checked functions.

Status

Implemented.

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.

Balance Tracking

In the `ybs_contract`, the balance of each non-blocked account is stored as shares, and then multiplied by the active multiplier, as it can be seen in the `convert_rebase_shares_to_tokens` function defined in `contracts/ybs_contract/src/shares.rs:239-244`.

```
pub(crate) fn convert_rebase_shares_to_tokens(e: &Env, shares: i128) -> i128 {
    let numerator: i128 = shares
        .checked_mul(get_active_multiplier(e))
        .expect("no overflow");
    numerator / BASE
}
```

The inverse function, `convert_to_rebase_shares` is also defined in the same file on lines 246-251 and makes a division.

```
fn convert_to_rebase_shares(e: &Env, amount: i128) -> i128 {
    let numerator: i128 = amount.checked_mul(BASE).expect("no overflow");
```

```
   numerator
        .checked_div(get_active_multiplier(e))
        .expect("no divide by 0")
}
```

The active multiplier is administered by the rebase admin and rebaser roles.

Given that integer arithmetic is used, even though values are multiplied and divided by `BASE` (defined in `contracts/ybs_contract/src/rebase.rs:4` as `1e18`) some small rounding errors will happen when converting from tokens to shares, which will be carried onto all the operations that make that conversion, which are:

1. `transfer`
2. `transfer_from`
3. `mint`
4. `burn`
5. `burn_from`
6. `unblock_addrs`

Given that the requested amount is logged when minting, burning and transferring tokens. Special care should be taken when analyzing those events, as they will not add up as expected.

Centralization

The admin is all powerful in the context of the `ybs_contract`, as they can change the executed code at any time.

Besides that, it must be noted that the asset protector can burn all the tokens owned by any account by blocking it first via the `block_addrs` function and then invoking the `wipe_blocked_addr` function.

Also, the rest of the privileged roles can also impact the earnings of a user in a centralized manner.

Upgrades

The `ybs_contract` can be upgraded by the admin of the contract via the `upgrade` function. This implies that the admin needs to be fully trusted as they can upload any arbitrary implementation.

Privileged Roles

ybs_contract

Admin

The admin of the contract can:

- upgrade the code of the contract via the `upgrade` function.
- grant an account the following roles via the `grant_role` function[6]:
    - Supply controller.
    - Pauser.
    - Asset Protector.
    - Rebase admin.
    - Rebaser.
- revoke the role for an account via the `revoke_role` function[7]:
    - Supply controller.
    - Pauser.
    - Asset Protector.
    - Rebase admin.
    - Rebaser.
- Initiate the process to transfer the admin role to a new account via the `start_admin_transfer` function.

The initial admin is set in the `initialize` function.

Asset Protector

An asset protector can:

- block accounts from minting, burning, transferring away, approving, being approved, receiving yields or receiving funds via the `block_addrs` function.
- block accounts from receiving funds by transfer or mint via the `block_addrs_from_receiving` function.
- reallow accounts to mint, burn, transfer away, approve, be approved, receive yields or receive funds via the `unblock_addrs` function.
- reallow account to receive funds by transfer or mint via the `unblock_addrs_from_receiving` function.

[6] Other roles can be granted but it will not have any effect given the current code of the contract.
[7] Other roles can be revoked but it will not have any effect given the current code of the contract.

- burn all the tokens of a blocked account via the `wipe_blocked_addr` function.

A single initial asset protector is set in the `initialize` function. The admin can give or take this role to/from any account via the `grant_role` and `revoke_role` functions.

Rebase Admin

A rebase admin can:

- set the next multiplier via the `set_next_multiplier` function. This function bypasses some of the checks in the `increase_rebase_multiplier` function that can be executed by a rebaser.
- set the rebase period, measured in seconds via the `set_rebase_period` function.
- set the upper bound on multiplier increases via the `set_max_rebase_rate` function.

A single initial rebase admin is set in the `initialize` function. The admin can give or take this role to/from any account via the `grant_role` and `revoke_role` functions.

Rebaser

A rebaser can increase the rebase multiplier via the `increase_rebase_multiplier` function.

A single initial rebaser is set in the `initialize` function. The admin can give or take this role to/from any account via the `grant_role` and `revoke_role` functions.

Pauser

A pauser can:

- halt approvals and transfers via the `pause` function.
- resume approvals and transfers via the `unpause` function.

A single initial pauser is set in the `initialize` function. The admin can give or take this role to/from any account via the `grant_role` and `revoke_role` functions.

Pending admin

The pending admin can accept the admin role by invoking the `accept_admin_transfer` function.

Supply controller

A supply controller can:

- mint tokens for any account via the `mint` function.

- burn tokens owned by itself via the `burn` and `burn_from` functions.

A single initial supply controller is set in the `initialize` function. The admin can give or take this role to/from any account via the `grant_role` and `revoke_role` functions.

About CoinFabrik

CoinFabrik is a research and development company specialized in Web3, with a strong background in cybersecurity. Founded in 2014, we have worked on over 500 decentralization projects, including EVM-based and other platforms like Solana, Algorand, and Polkadot. Beyond development, we offer security audits through a dedicated in-house team of senior cybersecurity professionals, working on code in languages such as Substrate, Solidity, Clarity, Rust, TEAL, and Stellar Soroban.

Our team has an academic background in computer science, software engineering, and mathematics, with accomplishments including academic publications, patents turned into products, and conference presentations. We actively research in collaboration with universities worldwide, such as Cornell, UCLA, and École Polytechnique in Paris, and maintain an ongoing collaboration on knowledge transfer and open-source projects with the University of Buenos Aires, Argentina. Our management and people experience team has extensive expertise in the field.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior, and general documentation about the project. 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
- 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.

Severity Classification

Security risks are classified as follows[8]:

🟥 Critical	• Manipulation of governance voting result deviating from voted outcome and resulting in a direct change from intended effect of original results • Direct theft of any user funds, whether at-rest or in-motion, other than unclaimed yield • Direct theft of any user NFTs, whether at-rest or in-motion, other than unclaimed royalties • Permanent freezing of funds • Permanent freezing of NFTs • Unauthorized minting of NFTs • Predictable or manipulable RNG that results in abuse of the principal or NFT • Unintended alteration of what the NFT represents (e.g. token URI, payload, artistic content) • Protocol insolvency

[8] This classification is based on the smart contract Immunefi severity classification system version 2.3. https://immunefi.com/immunefi-vulnerability-severity-classification-system-v2-3/

▮ High	• Theft of unclaimed yield • Theft of unclaimed royalties • Permanent freezing of unclaimed yield • Permanent freezing of unclaimed royalties • Temporary freezing of funds • Temporary freezing NFTs
▮ Medium	• Smart contract unable to operate due to lack of token funds • Block stuffing • Griefing (e.g. no profit motive for an attacker, but damage to the users or the protocol) • Theft of gas • Unbounded gas consumption • Security best practices not followed
▮ Low	• Contract fails to deliver promised returns, but doesn't lose value • Other security issues with minor impact

Issue Status

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

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

Disclaimer

This audit report has been conducted on a **best-effort basis within a tight deadline defined by time and budget constraints**. We reviewed only the specific smart contract code provided by the client at the time of the audit, detailed in the Scope section. We do not review other components that are part of the solution: neither implementation, nor general design, nor business ideas that motivate them.

While we have employed the latest tools, techniques, and methodologies to identify potential vulnerabilities, **this report does not guarantee the absolute security of the contracts, as undiscovered vulnerabilities may still exist.** Our findings and recommendations are suggestions to enhance security and functionality and are not obligations for the client to implement.

The results of this audit are valid solely for the code and configurations reviewed, and any modifications made after the audit are outside the scope of our responsibility. CoinFabrik disclaims all liability for any damages, losses, or legal consequences resulting from the use or misuse of the smart contracts, including those arising from undiscovered vulnerabilities or changes made to the codebase after the audit.

This report is intended exclusively for the **Paxos** team and should not be relied upon by any third party without the explicit consent of CoinFabrik. Blockchain technology and smart contracts are inherently experimental and involve significant risk; users and investors should fully understand these risks before deploying or interacting with the audited contracts.

Changelog

Date	Description
2024-12-13	Initial report based on commit `5581ff364341e68e0c09424d37d80218048ce59d`.
2024-12-18	Final report with fixes on commit `ae682c6c4f145ab24e416ccf54cd19572ab8f8f5`.
2025-02-05	Update the report with new commits and repository.