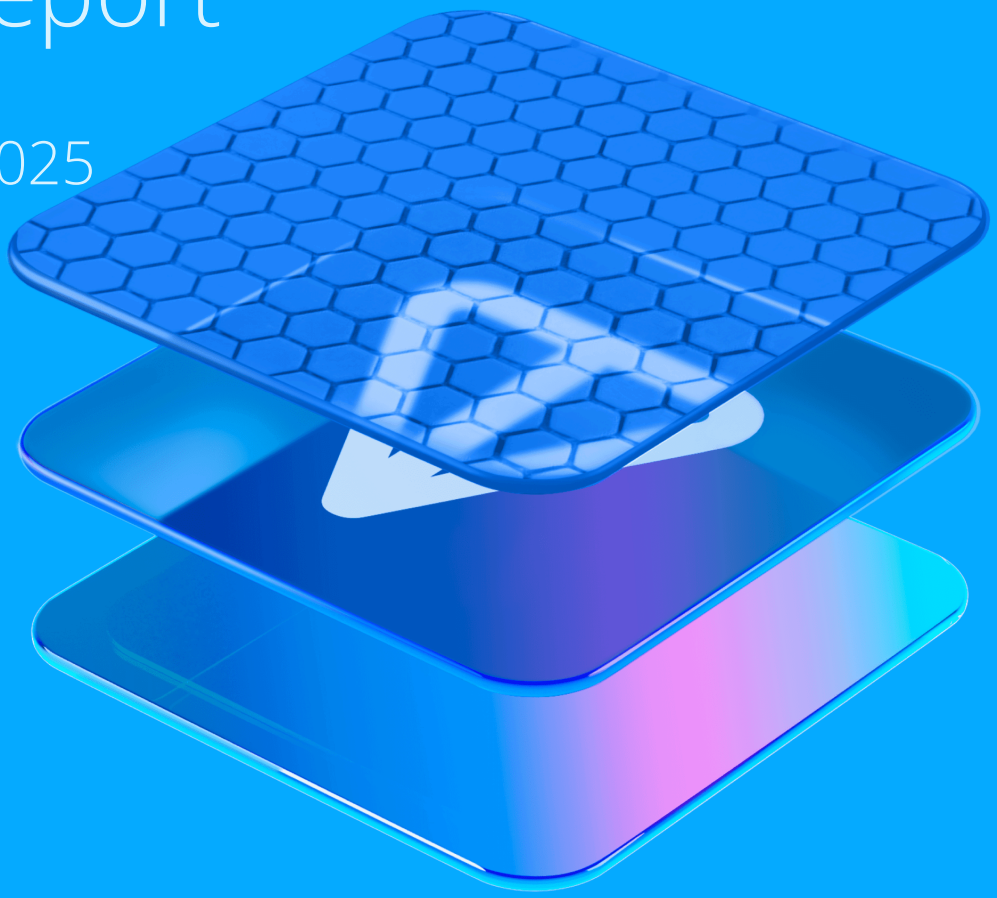
Torch
Audit Report

Fri Feb 14 2025



contact@bitslab.xyz https://twitter.com/tonbit_

TonBit

Torch Audit Report

1 Executive Summary

1.1 Project Information

Description	https://x.com/torchton?s=21&t=oaJalSXJZRmw46WZf6MIyw
Type	DeFi
Auditors	TonBit
Timeline	Sat Jan 25 2025 - Mon Feb 10 2025
Languages	Tolk
Platform	Ton
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/torch-core/universal-reward-distributor
Commits	970aff060c19c7c9a50574a608cc700171eb38e4 9e0f5bb24c7e862b7b9a1b74770b673d3d7b4fe5

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MER	contracts/library/merkle.tolk	c046b2d9e6cdcd7356f03afd4725d1aef84d829e
FEE	contracts/library/fee.tolk	f9aa72b7a9709bf3196bdf3bd40afcdb8dc31001
MES	contracts/library/message.tolk	c88c3361200909c079b52ce79545f9d3ec803bf0
ADD	contracts/library/address.tolk	9d2bf5780e53d9e2fbb7269b8355240232373cff
LOG	contracts/library/log.tolk	7cfb0cec8c512f2fe08b8fc1673d957bd9384c45
MAI	contracts/rewardAccount/main.tolk	64dd055fa3bb38cf08ab4265125c1861a349325e
STO	contracts/rewardAccount/storage.tolk	9684fb44909452a3412225b8ab98047f19f438cc
GUA	contracts/rewardAccount/guard.tolk	bc3358fe42a0977e6e4d1b2a12166b3e021ebd0d
SER	contracts/rewardAccount/serdes.tolk	526f975e491740d294f3e9063cabe9a7f66a1851
GAS	contracts/rewardAccount/gas.tolk	22c970b91c3098e8d608953625692daaddc0b1a6
CBCMT	contracts/baseContract/main.tolk	5ba16437beebdb6e9eaa0694409c4f1d05205a38

CBCST	contracts/baseContract/storages.tolk	0fec3a1236cb9cbebd101cceb3b7ee9f23fdc659
CFMT	contracts/factory/main.tolk	61eea2b7fd67fc4357837009778b838eb8492feb
HAN	contracts/factory/handler.tolk	23a39f5d0cc88b3ab2cba0da6720c4134b180fcf
CFST	contracts/factory/storage.tolk	9eed5a4682e9d9ab7b8b748fe593958c35a00855
UTI	contracts/factory/utils.tolk	f8efc1ac570e13748f8cfddd7f708da5560c1084
CFST	contracts/factory/serdes.tolk	ee7b520df5d7d36898da71ed2c15c85ea7464b12
CFGT	contracts/factory/gas.tolk	3d301fbd5b0f7aff51d8bcae9c484ff41705b3c4
CFLT	contracts/factory/log.tolk	85db9dba83e0b48cad699beab1e70dbc7177a019
OPC	contracts/common/opcodes.tolk	9cfd86a7e2d813b54da85e59bbbfda8363f26e86
SEN	contracts/common/send.tolk	d92529fc16ca80e3457a03d54e809f86db92d318
CCGT	contracts/common/guard.tolk	77d5c23d2f86bafed5c682a47261e2a6f47cbae0
CON	contracts/common/constants.tolk	de50d87bef7c8267bf603a659eb54fe7d153cec4
EXI	contracts/common/exitcodes.tolk	4c6a928add1969d4cd540cadb086beaf14385842

CCST	contracts/common/serdes.tolk	f1bdee78053580b6350aab84786828e2e4f68ccc
CCGT	contracts/common/gas.tolk	0fdc498dfe9ee461a14c4a0f205a8e06bca8edad
CCLT	contracts/common/log.tolk	23d3daf582ed7d1c65cd2d477b7475e40e2d97c3
CDTDMT	contracts/distributor/tonDistributor/main.tolk	99ec28a11ee9770e9b84be1a83bc5c1f446bd7e7
CDTDHT	contracts/distributor/tonDistributor/handler.tolk	62658f71e7ddbed008bec9de04be0bdc7af38961
CDTDST	contracts/distributor/tonDistributor/storage.tolk	f107a5e081e6604605a98bacf6410795469f4556
CDTDST	contracts/distributor/tonDistributor/serdes.tolk	1da3dedc094894addb2730842a33618e92080492
CDHT	contracts/distributor/handler.tolk	c4fa25607f7a9a427c2e9e4df1e6edb22d5fe84d
CDJDMT	contracts/distributor/jettonDistributor/main.tolk	24e074272e62dcbb3d3ed19a403eabb3336d9504
CDJDHT	contracts/distributor/jettonDistributor/handler.tolk	a62706b6e06ba7ce24bb39d37ea939ae0638cd52
CDJDST	contracts/distributor/jettonDistributor/storage.tolk	1d5abf1f4116d871380e781c6040be0008c87b1a
CDJDGT	contracts/distributor/jettonDistributor/guard.tolk	447b2acebd750aeb85eecd550911582f5a505d3f
CDJDST	contracts/distributor/jettonDistributor/serdes.tolk	62a88fcc82517fcb4d943aa3751778ef8ad64de4

CDGT	contracts/distributor/guard.tolk	821a8d8f5b7c8d9762059e16530a566e4ff1f38e
CDUT	contracts/distributor/utils.tolk	0656a8df08a0708c262441d76edc1b4ba522dae4
CDST	contracts/distributor/serdes.tolk	109676ce571cf6d9c000b4c955f86928c2c496cd
CDGT	contracts/distributor/gas.tolk	8e23d229003211ab56a47e451c5f307cc649689e
CDLT	contracts/distributor/log.tolk	478b54547269f6eef6d0a94f8962931fbc462eb5

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	14	13	1
Informational	1	1	0
Minor	7	6	1
Medium	4	4	0
Major	2	2	0
Critical	0	0	0

1.4 TonBit Audit Breakdown

TonBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Torch to identify any potential issues and vulnerabilities in the source code of the Torch smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

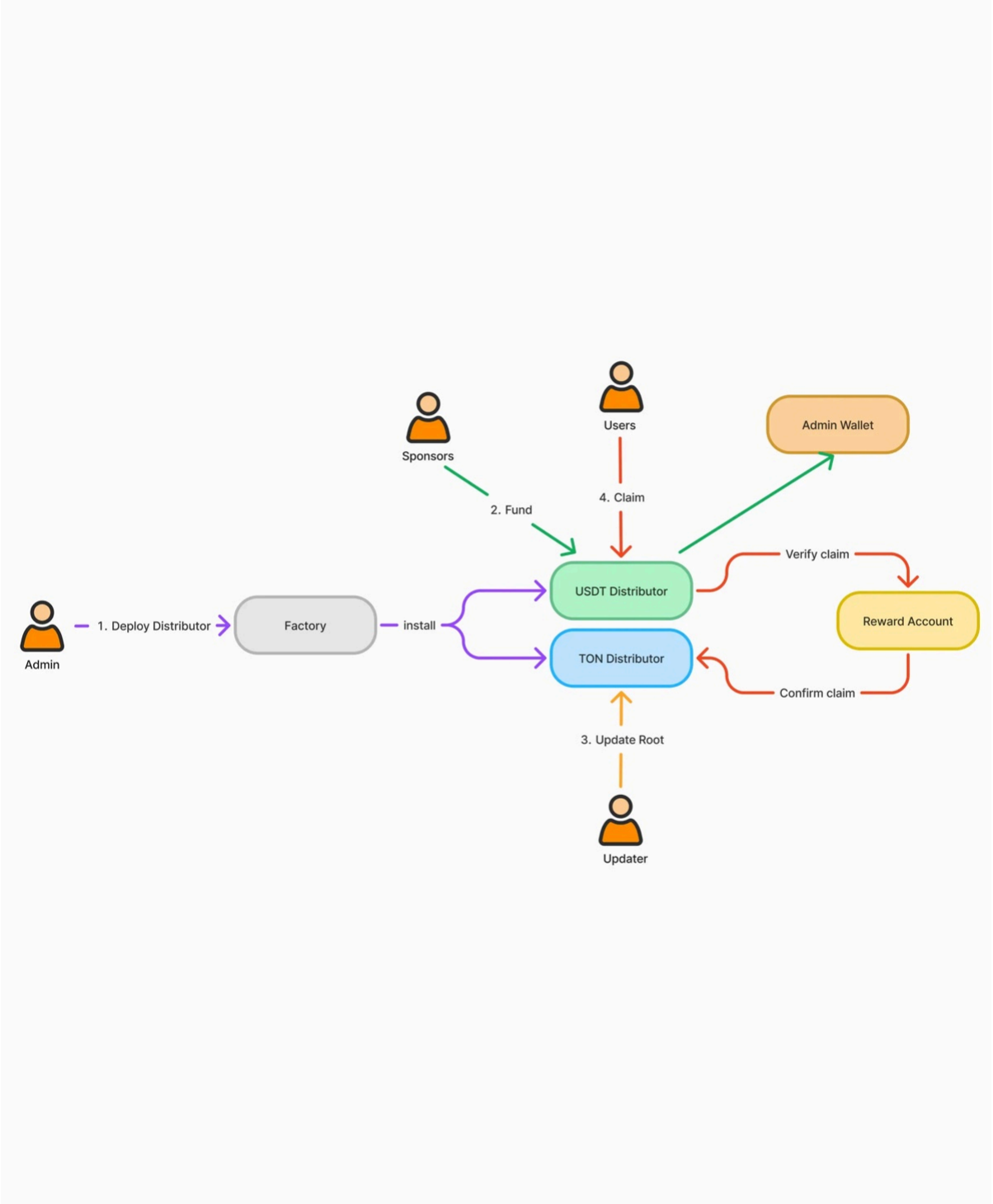
During the audit, we identified 14 issues of varying severity, listed below.

ID	Title	Severity	Status
GAS-1	StorageFee Collected at OP_VERIFY_CLAIM Not Accounted for in `totalFee`	Major	Fixed
HAN-1	Healthier Approach to Calculating Handling Fees	Minor	Fixed
MAI-1	Parameters that Can Be Ignored	Minor	Fixed
GAS1-1	Floor Rounding in `fundDurationInMonths` Calculation	Medium	Fixed
MAI1-1	Delayed `ctxSender` Check	Major	Fixed
MAI1-2	Rollback Error Trigger	Medium	Fixed
MAI1-3	`targetAsset` Field Lacks a Fee	Medium	Fixed
MAI1-4	Repeated Updates to the Same Data	Medium	Fixed
MAI1-5	Single-step Ownership Transfer Can be Dangerous	Minor	Acknowledged

MAI1-6	`amount` Field Lacks of check	Minor	Fixed
MAI1-7	Centralization Risk	Minor	Fixed
MAI1-8	Logging Messages with Potential Errors	Minor	Fixed
MAI1-9	Missing `Claimable` Validation or Rollback Mechanism	Minor	Fixed
MAI1-10	Missing Logging on OP_PAUSE/OP_ACTIVATE	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Torch Smart Contract :



The flowchart of the program is shown above.

1. Admin can call factory contract to deploy two types of distributor. Ton or USDT
2. Sponsors can call fund to add the reward amount to the distributor and pay the fee to Admin Wallet.
3. Updater can update the Merkle Root to update the amount that User can collect.

4. Users can call claim to get the rewards by calling the confirm claim of distributor by Reward Account after passing the verification.

4 Findings

GAS-1 StorageFee Collected at OP_VERIFY_CLAIM Not Accounted for in `totalFee`

Severity: Major

Status: Fixed

Code Location:

contracts/distributor/gas.tolk#72;

contracts/rewardAccount/gas.tolk#11

Descriptions:

In the **FUND_DISTRIBUTOR** flow, **OP_CLAIM** calculates totalFee without calculating the 1.5 times **StorageFee** that was collected at **OP_VERIFY_CLAIM**, which may result in the ctxValue being insufficient to complete the full message flow. This issue is one that Tonbit and Torch have discovered together.

Suggestion:

Suggested to take StorageFee charged in **OP_VIRIFY_CLAIM** into account

Resolution:

- Doubled the reward account storage fee reserve for more accurate future storage fee estimation
- Enhanced gas fee computation to better account for potential storage costs during claim verification

HAN-1 Healthier Approach to Calculating Handling Fees

Severity: Minor

Status: Fixed

Code Location:

contracts/distributor/jettonDistributor/handler.tolk#42;

contracts/distributor/tonDistributor/handler.tolk#37

Descriptions:

We found that the calculation used to calculate the handling fee was rounded down, in which case there was an extreme case in which the handling fee could be zero, and we recommend replacing it with rounding up.

```
...
 // Charge adminFee from the sponsor and send it to the admin
   val adminFee: int = mulDivFloor(amount, adminFeeRate, ADMIN_FEE_DENOMINATOR);
   val amountToDistribute: int = amount - adminFee;

   // Update distributable
   distributable += amountToDistribute;
...
```

Suggestion:

Change mulDivFloor method to mulDivCeil

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MAI-1 Parameters that Can Be Ignored

Severity: Minor

Status: Fixed

Code Location:

contracts/rewardAccount/main.tolk#74

Descriptions:

The `myBalance` parameter is not used or called in the function, which may be detrimental to code maintenance and may cause parameter pollution and misoperation during use.

```
fun onInternalMessage(myBalance: int, msgValue: int, inMsgFull: cell, inMsgBody: slice) {
```

Suggestion:

It is recommended to delete invalid parameters or ignore them to reduce the possibility of misoperation.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

GAS1-1 Floor Rounding in `fundDurationInMonths` Calculation

Severity: Medium

Status: Fixed

Code Location:

contracts/common/gas.tolk

Descriptions:

In `computeFundFee` method, the calculation of `fundDurationInMonths` is rounded down, which results in, for example: (2* ONE_MONTH_IN_SECONDS-1) being calculated as one month.

Suggestion:

Suggest rounding up when calculating fundDurationInMonths.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MAI1-1 Delayed `ctxSender` Check

Severity: Major

Status: Fixed

Code Location:

contracts/distributor/jettonDistributor/main.tolk#162

Descriptions:

When the jetton distributor processes **OP_FUND**, the check on **ctxSender** is performed too late. This means that under the paused condition of the jetton distributor, any message sender can trigger the distributor to enter the refund process. Additionally, since the **jettonAmount** field is user-input, this could result in sending an arbitrary amount of jettons from the `jettonWallet` to the user.

```
    if (op == OP_TRANSFER_NOTIFICATION) {
        // Parse jetton notification message
        var (jettonAmount: int, jettonSender: slice, forwardPayload: slice) =
ctxBody.parseJettonNotification();

        // Check if jetton amount is greater than 0 (Don't allow non-positive jetton amount)
        assert(jettonAmount > 0, ERROR_NON_POSITIVE_AMOUNT);

        // Get jetton op codes
        val jettonOp: int = forwardPayload.loadMessageOp();

        // Sponsor fund this jetton reward distributor
        if (jettonOp == OP_FUND) {
            // Only fund when distributor is active and init
            if ((!isActive) || (addressIsNone(jettonWallet))) {
                val (errorCode: int , refundWallet: slice) = !isActive ?
(ERROR_DISTRIBUTOR_PAUSE, jettonWallet) : (ERROR_NOT_INITIALIZED, ctxSender);
                sendJettonMessage(
                    BOUNCEABLE,
                    jettonSender,
                    jettonSender,
                    refundWallet,
                    jettonAmount,
                    ctxValue - computeRefundFee(JETTON_PREFIX), // Remaining TON after refund
```

```
fee
                queryID,
                packCommentPayload("Refund: Distributor is paused"),
                NO_FOWARD_TON,
                SEND_MODE_REGULAR
            );
            commitContractDataAndActions();
            throw errorCode;
        }

        // Check if ctxSender is jetton wallet (Since everyone can send jetton notification)
        ctxSender.requireJettonWallet(jettonWallet);
```

Suggestion:

Suggest to check ctxSender earlier.for example:

```
    if (op == OP_TRANSFER_NOTIFICATION) {
        // Check if ctxSender is jetton wallet (Since everyone can send jetton notification)
        ctxSender.requireJettonWallet(jettonWallet);

        // Parse jetton notification message
        var (jettonAmount: int, jettonSender: slice, forwardPayload: slice) =
ctxBody.parseJettonNotification();

        // Check if jetton amount is greater than 0 (Don't allow non-positive jetton amount)
        assert(jettonAmount > 0, ERROR_NON_POSITIVE_AMOUNT);

        // Get jetton op codes
        val jettonOp: int = forwardPayload.loadMessageOp();

        // Sponsor fund this jetton reward distributor
        if (jettonOp == OP_FUND) {
            // Only fund when distributor is active and init
            if ((!isActive) || (addressIsNone(jettonWallet))) {
                val (errorCode: int , refundWallet: slice) = !isActive ?
(ERROR_DISTRIBUTOR_PAUSE, jettonWallet) : (ERROR_NOT_INITIALIZED, ctxSender);
                sendJettonMessage(
                    BOUNCEABLE,
                    jettonSender,
                    jettonSender,
                    refundWallet,
```

```
            jettonAmount,
            ctxValue - computeRefundFee(JETTON_PREFIX), // Remaining TON after refund
fee
            queryID,
            packCommentPayload("Refund: Distributor is paused"),
            NO_FOWARD_TON,
            SEND_MODE_REGULAR
        );
        commitContractDataAndActions();
        throw errorCode;
    }
```

Resolution:

Regardless of whether the contract is in a paused state or the jetton wallet has not been initialized, ctxSender should always be used as the refund wallet. This ensures that the sender can only reclaim the jettons they originally transferred and cannot withdraw additional jettons from the distributor.

MAI1-2 Rollback Error Trigger

Severity: Medium

Status: Fixed

Code Location:

contracts/distributor/tonDistributor/main.tolk;

contracts/distributor/jettonDistributor/main.tolk

Descriptions:

Summary

The rollback process was triggered by an error

Vulnerability Detail

The rollback check mechanism is delayed when the sponsor injects funds

```
if (op == OP_CONFIRM_CLAIM) {
...
 if ((claimable > distributable) || (!isActive)) {
```

Root Cause

When distributing, the sponsor injection time overlaps and the distributable is shared

Internal pre-conditions

When the reward calculation value is too large

External pre-conditions

Miscalculation when there is new sponsor funding

Problem Path

1. admin -> A Sponsor -> distributable += amountToDistribute
2. merkle -> B, C, D user -> B, C, D claimable == distributable
3. B, C claimed -> distributable = distributable - claimable
4. B Sponsor -> distributable += amountToDistribute;
5. D claimed (bypass check `claimable > distributable`)
6. The correct funds of other users cannot be withdrawn and a rollback is triggered

Impact

Current user of the distributor contract. The rollback mechanism may be triggered late, and users who are correctly rewarded may be DOS

Suggestion:

Limit the process and roll back correctly when errors occur.

Resolution:

The client confirms that this problem is handled manually through off-chain monitoring. When a calculation error occurs, the processing is manually suspended. When a rollback is triggered, it only serves as a check prompt to alleviate this problem.

MAI1-3 `targetAsset` Field Lacks a Fee

Severity: Medium

Status: Fixed

Code Location:

contracts/distributor/tonDistributor/main.tolk#127

Descriptions:

When the distributor processes **OP_FUND**, it emits a log. The **targetAsset** field is filled in by the user, but currently, the contract does not impose any restrictions or fees on targetAsset. If the user provides a large targetAsset, the contract balance will decrease when the distributor emits the log. This could potentially lead to the contract being frozen.

Suggestion:

Suggest charging for or limiting the length of the `targetAsset` field.

Resolution:

Add a validation check for targetAsset to ensure it can only be a TON asset or a Jetton asset.

MAI1-4 Repeated Updates to the Same Data

Severity: Medium

Status: Fixed

Code Location:

contracts/distributor/jettonDistributor/main.tolk#301-318 321-337 340-358;

contracts/distributor/tonDistributor/main.tolk#242-259 262-277 282-297

Descriptions:

We found that the above code content can be called successfully if the same data is updated repeatedly, and doing so for repeated calls will consume gas and cause non-essential economic loss.

Suggestion:

A check to reject the update if the updated data has not changed should be added.

Resolution:

To address this issue, we will implement a check to determine whether the update information is identical to the existing stored data.

- If the new update matches the existing data, the transaction will be reverted or ignored.
- This ensures that only meaningful updates proceed, reducing unnecessary gas consumption and improving contract efficiency.

MAI1-5 Single-step Ownership Transfer Can be Dangerous

Severity: Minor

Status: Acknowledged

Code Location:

contracts/factory/main.tolk

Descriptions:

The `OP_TRANSFER_ADMIN()` (factory) function has a problem with single-step permission transfer.

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract.

Suggestion:

It is recommended to update the code to fix this issue.

MAI1-6 `amount` Field Lacks of check

Severity: Minor

Status: Fixed

Code Location:

contracts/distributor/tonDistributor/main.tolk#130

Descriptions:

When the TON distributor processes **OP_FUND**, it does not check if the **amount** field filled in by the user is greater than **ctxValue**.

Because if the amount specified by the user exceeds the amount they actually sent, it will lead to subsequent calculation errors. For example, remainingTON may be a negative number, affecting the normal operation of the contract.

Although there is an implicit addition check in the code, it will cause an overflow error and fail.

```
val remainingTON: int = ctxValue - gasConsume - amount;
```

```
// Parse fund TON message
val (amount: int, targetAsset: cell, startTime: int, endTime: int, blacklist: cell) =
ctxBody.parseFundTON();
// Check if amount is greater than 0 (Don't allow non-positive amount)
assert(amount > 0, ERROR_NON_POSITIVE_AMOUNT);
// If startTime is not within now() to now() + 60 days, refund
val currentTime: int = now();
if (verifyStartTime(startTime, currentTime)) {
    sendSimpleMessage(
        BOUNCEABLE,
        ctxSender,
        ctxValue - computeTimeRefundFee(TON_PREFIX),
        packCommentBody("Refund: Incorrect start time"),
        SEND_MODE_REGULAR
    );
    commitContractDataAndActions();
    throw ERROR_INCORRECT_START_TIME;
}
```

Suggestion:

Suggest adding checks for `amount` and `ctxValue` For example:

```
if (op == OP_FUND) {
    ...
    assert(amount <= ctxValue, ERROR_NOT_ENOUGH_GAS);

    // Check if amount is greater than 0 (Don't allow non-positive amount)
    assert(amount > 0, ERROR_NON_POSITIVE_AMOUNT);

    // ... existing code ...

    if (amount > ctxValue + fee) {
        //return logic
    }
```

Resolution:

Ensure that when checking ctxValue, the contract also verifies that it is greater than or equal to the specified amount. The following assertion should be added:

assert(ctxValue >= totalFee + amount, ERROR_NOT_ENOUGH_GAS);

MAI1-7 Centralization Risk

Severity: Minor

Status: Fixed

Code Location:

contracts/distributor/jettonDistributor/main.tolk

Descriptions:

Centralization risk was identified in the smart contract:

- Admin(single node permissions) can directly modify the data in the contract through `setContractData` and `setContractCodePostponed` in `OP_UPGRADE_CONTRACT`.
- The maximum percentage of handling fee that the admin can set is up to 100%.
- Admin(non-updater robots) can directly modify `merkleRoot`, which will affect the rewards that can be collected.

Suggestion:

It is recommended that measures be taken to reduce the risk of centralization, such as a multi-signature mechanism.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MAI1-8 Logging Messages with Potential Errors

Severity: Minor

Status: Fixed

Code Location:

contracts/distributor/tonDistributor/main.tolk#184-202;

contracts/distributor/jettonDistributor/main.tolk#234-252

Descriptions:

```
...
ctxSender.requireAuthorized(admin, updater);
...
logRootUpdated(updater, newRoot);
```

We observe that the above code location can be called by admin and updater, but the update information of the log is only updater, and does not include admin, so every time admin and updater update the log information is not effective to determine who the change is.

Suggestion:

Add judgment conditions to trigger more detailed log messages.

Resolution:

update root update logging to use context sender

MAI1-9 Missing `Claimable` Validation or Rollback Mechanism

Severity: Minor

Status: Fixed

Code Location:

contracts/distributor/tonDistributor/main.tolk#101

Descriptions:

When processing **OP_CONFIRM_CLAIM**, if the value of `claimable` is insufficient to cover the cost of sending the reward message, then the transaction will be partially executed and the `totalClaimed` field in the `rewardAccount` will not be rolled back.

Suggestion:

Suggest adding a check on `claimable` field, or adding a rollback mechanism.

Resolution:

If claimable < 0.03 TON, the transaction will roll back to prevent the user's reward distribution from failing.

MAI1-10 Missing Logging on OP_PAUSE/OP_ACTIVATE

Severity: Informational

Status: Fixed

Code Location:

contracts/distributor/tonDistributor/main.tolk#224,238

Descriptions:

Calls to **logDistributorPaused/logDistributorActivated** functions are missing when **tonDistributor** handles **OP_PAUSE/OP_ACTIVATE**.

Suggestion:

Suggest adding a call to **logDistributorPaused/logDistributorActivated** to the **OP_PAUSE/OP_ACTIVATE** processing

Resolution:

Add calls to logDistributorPaused and logDistributorActivated within the respective handling functions for OP_PAUSE and OP_ACTIVATE in TON distributor.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.