DIA GENESIS STAKING SECURITY AUDIT REPORT

Dec 03, 2024

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals

- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

The DIA Prestaking protocol enables users to lock up their DIA tokens for a predetermined period to accumulate yield rewards. The protocol allows its owners to create epochs with distinct start times, end times, and durations, allowing users to select from available epochs to deposit their DIA tokens. Tokens can be withdrawn after the designated release time.

1.4 Project Dashboard

Project Summary

Title	Description
Client	DIA
Project name	Genesis Staking
Timeline	15.10.2024 - 23.10.2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
15.10.2024	6ee4beee6cbcdb3b7e7ea1feb9a8b1192bc8c8ba	Commit for the audit
22.10.2024	d1aa7ee0ea9ff1a256c65d30a6ff635edb9cd180	Commit for the re-audit
23.10.2024	cd5035149d4363d2833490196cc4810cc6acd5f1	Commit with updates

Project Scope

The audit covered the following files:

File name	Link
prestaking/src/roles/PreStakingRoles.sol	PreStakingRoles.sol
prestaking/src/Prestaking.sol	Prestaking.sol

Deployments

File name	Contract deployed on mainnet	Comment
Prestaking.sol	0x96DB120A7E0A5d91d32BAFa5c7857930Ba172142	

1.5 Summary of findings

Severity	# of Findings
Critical	1
High	0
Medium	3
Low	12

ID	Name	Severity	Status
C-1	Epoch Manipulation	Critical	Fixed
M-1	Missing Validation for Input Parameters in `createNewEpoch` Function	Medium	Fixed
M-2	DoS Risk Due to Small Deposits and Front-Running in `deposit` Function	Medium	Fixed
M-3	Yield Amount Not Reset in `withdrawPrincipal` Function	Medium	Fixed
L-1	Function Visibility	Low	Fixed
L-2	Epoch Index Validation in `deposit` Function	Low	Fixed
L-3	Missing Check for Zero Address in `updateBeneficiary` Function	Low	Fixed
L-4	Potential Reentrancy Risk in `withdrawPrincipal` Function	Low	Fixed
L-5	Missing Validation for `newEndTime` in `updateEndDepositTime` Function	Low	Fixed

L-6	`SafeERC20` is not used	Low	Fixed
L-7	Unnecessary `require` checks for token transfers	Low	Fixed
L-8	Missing events for epoch creation and update of `endDepositTime`	Low	Fixed
L-9	Missing checks for array indexes	Low	Fixed
L-10	`renounceOwner` can be called by anyone	Low	Fixed
L-11	`withdrawPrincipal` can be called multiple times for the same wallet	Low	Fixed
L-12	`updateBeneficiary` can be called on the already withdrawn wallets	Low	Fixed

1.6 Conclusion

During the audit, we thoroughly tested critical attack vectors and verified the following:

1. It is not possible to make a deposit into a non-existent epoch, as this would fail the combination of checks `amount <= currentEpoch.maxTokens` and `amount > 0`.
2. It is impossible to withdraw more tokens than there were initially deposited.
3. There is no chance for a user's funds to be stolen, as sufficient checks are in place when tokens are withdrawn.
4. No tokens can be forever locked on the contract, as a `releaseTime` is configured, after which tokens deposited to a particular epoch become withdrawable.
5. All functions that require additional authorization perform necessary checks or use `AccessControl` modifiers.

2.FINDINGS REPORT

2.1 Critical

C-1	Epoch Manipulation
Severity	Critical
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `deposit` function of the `Prestaking` contract.
The `epochIndex` is not properly set or stored within the staking wallet structure. As a result, users can manipulate the staking mechanism by selecting any epoch during deposit, allowing them to exploit epochs with more favorable conditions, such as earlier unlock times. This can lead to users bypassing the intended locking periods and gaining access to their staked tokens earlier than allowed, breaking the fairness of the staking system.
The issue is classified as **Critical** severity because it allows users to bypass staking conditions, potentially undermining the entire staking system.

Recommendation

We recommend setting and storing the `epochIndex` in the `StakingWallet` structure to ensure that the staking conditions, including the lockup and release times, are tied to the correct epoch.

Client's Commentary

> The issue has been fixed in commit 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.
> The `epochIndex` is stored in `StakingWallet` when `deposit()` is called by a staker.

2.2 High

Not Found

2.3 Medium

M-1	Missing Validation for Input Parameters in `createNewEpoch` Function
Severity	Medium
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `createNewEpoch` function of the `Prestaking` contract.
The input parameters for creating a new epoch are not validated, which could lead to incorrect or unintended initialization. Specifically:

- `newStartDepositTime` should be greater than or equal to `block.timestamp`.
- `newEndDepositTime` should be greater than `newStartDepositTime`.
- The sum of `newEndDepositTime` and `newDuration` should not be set too far in the future to prevent indefinite token lockups.
- `newBasisPoints` should be capped at a reasonable maximum value to avoid excessively high yields.
  Without these checks, it could lead to epochs with incorrect or undesirable configurations that might lock tokens indefinitely or apply excessive yields.
  The issue is classified as **Medium** severity because it affects the integrity of the prestaking process and could lead to operational issues for both the contract and its users.

Recommendation

We recommend adding validation checks for each of the input parameters to ensure:

- `newStartDepositTime >= block.timestamp`;
- `newEndDepositTime > newStartDepositTime`;
- `newEndDepositTime + newDuration` is not too far in the future;
- `newBasisPoints` is capped at a reasonable maximum.
  This will ensure the correct initialization of epochs and prevent unintended behavior.

Client's Commentary

> The issue has been fixed in commit 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002 and the checks have been implemented.

M-2	DoS Risk Due to Small Deposits and Front-Running in `deposit` Function
Severity	Medium
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `deposit` function of the `Prestaking` contract.
There are two potential attack vectors that could be used to DoS the system:

1. **Small Deposits (1 wei deposits)**: Malicious users can fill the `stakingWallets` with extremely small deposits (e.g., 1 wei), which would make it harder to use the system effectively. Adding a minimum deposit amount would prevent such attacks by ensuring that deposits are meaningful, thus limiting the attacker's ability to spam the system with tiny amounts.

2. **Front-Running the Max Token Limit**: A malicious user could front-run other transactions by depositing a small amount (e.g., 1 wei) just before another user's deposit, which would cause the second transaction to revert due to the `maxTokens` limit being breached. This is a denial-of-service risk that can be mitigated by adjusting the logic to ensure that deposits do not push the total amount beyond the `maxTokens` limit.

The issue is classified as **Medium** severity because it can be exploited to disrupt the normal operation of the staking system, potentially preventing valid users from depositing tokens.

Recommendation

- **Minimum Deposit Requirement**: Implement a minimum deposit amount to ensure that users cannot spam the system with tiny deposits.
- **Max Token Check**: Ensure that deposits do not exceed the `maxTokens` limit by using `maxTokens` instead of `amount` if `amount > maxTokens` is true.

Client's Commentary

> The minimum deposit has been set to 1 token in this commit:
> d1aa7ee0ea9ff1a256c65d30a6ff635edb9cd180.
> The frontrunning issue has been addressed in this commit:
> bebc1f2acc1d97cd27c2e8ead9aadf458f1ca438.

M-3	Yield Amount Not Reset in `withdrawPrincipal` Function
Severity	Medium
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `withdrawPrincipal` function of the `Prestaking` contract.
While the principal balance is correctly set to zero after the withdrawal, the `yieldAmount` remains unchanged. If the `yieldAmount` is not reset to zero, it could lead to incorrect state data.
The issue is classified as **Medium** severity because it can lead to inconsistencies in the staking wallet's state and possible incorrect yield calculations in the future.

Recommendation

We recommend resetting the `yieldAmount` to zero along with the principal balance. This will ensure that both the principal and the yield are correctly cleared after withdrawal.

Client's Commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

2.4 Low

L-1	Function Visibility
Severity	Low
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `getStakingWalletsForAddress` function of the `Prestaking` contract.
The function is currently marked as `public`, but it does not require access within the contract and can be marked as `external`.

Recommendation

We recommend changing the visibility of the `getStakingWalletsForAddress` function to `external`.

Client's Commentary

> The issue has been fixed in commit 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-2	Epoch Index Validation in `deposit` Function
Severity	Low
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `deposit` function of the `Prestaking` contract.
The function does not validate whether the provided `epochIndex` exists before using it.

Recommendation

We recommend adding a validation check to ensure that the `epochIndex` exists before proceeding with the deposit logic.

Client's Commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-3	Missing Check for Zero Address in `updateBeneficiary` Function
Severity	Low
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `updateBeneficiary` function of the `Prestaking` contract. The function allows the current beneficiary to update the wallet that will receive the staked tokens. However, there is no check to ensure that the `newBeneficiary` address is not the zero address.

Recommendation

We recommend adding a validation check to ensure that the `newBeneficiary` address is not the zero address (`require(newBeneficiary != address(0), "Beneficiary cannot be zero address.");`) before updating the staking wallet's beneficiary.

Client's Commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-4	Potential Reentrancy Risk in `withdrawPrincipal` Function
Severity	Low
Status	Fixed in d1aa7ee0

Description

This issue has been identified in the `withdrawPrincipal` function of the `Prestaking` contract.
The function transfers the staked tokens to the beneficiary before setting the balance of the staking wallet to zero, creating a potential reentrancy vulnerability.

Recommendation

We recommend updating the beneficiary's balance to zero before transferring tokens to prevent any potential reentrancy attack.

Client's Commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-5	Missing Validation for `newEndTime` in `updateEndDepositTime` Function
Severity	Low
Status	Fixed in cd503514

Description

This issue has been identified in the `updateEndDepositTime` function of the `Prestaking` contract. The function currently allows updating the `endDepositTime` without validating whether the new time is between the `startDepositTime` and `releaseTime` of the epoch.

Recommendation

We recommend adding validation to ensure that `newEndTime` is between the `startDepositTime` and `releaseTime` of the epoch.

Client's Commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-6	`SafeERC20` is not used
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the contract `Prestaking`. There is an import at Prestaking.sol#L4. But, `SafeERC20` logic is never used, as there are regular `transfer` and `transferFrom` functions in use.

Recommendation

We recommend resolving the unused import or switching to `safeTransfer` and `safeTransferFrom` functions.

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-7	Unnecessary `require` checks for token transfers
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the contract `Prestaking`. There are two calls to ERC20 token functions: Prestaking.sol#L132 and Prestaking.sol#L167. Those functions always return `true` in the case of `DIA` token implementation.

Recommendation

We recommend removing unnecessary `require` checks.

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-8	Missing events for epoch creation and update of `endDepositTime`
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the contract `Prestaking`. There are two functions: Prestaking.sol#L63 and Prestaking.sol#L173. Those functions don't emit events with the updated parameters of epochs.

Recommendation

We recommend introducing special events, which may be emitted when the mentioned functions are called.

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-9	Missing checks for array indexes
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the contract `Prestaking`. There are multiple functions that accept `stakingWalletNumber` as a parameter, but don't check whether it exists in the `stakingWallets` array. This issue is present in the Prestaking.sol#L81, Prestaking.sol#L152 and Prestaking.sol#L163 functions. This issue may lead to unexpected reverts.

Recommendation

We recommend introducing a special check for `stakingWalletNumber`, which ensures that it exists in the `stakingWallets` array

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-10	`renounceOwner` can be called by anyone
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the function PreStakingRoles.sol#L35 of contract `PrestakingRoles`. It is possible to call that function for anyone, what will lead to emitting unnecessary `OwnerRemoved` event, even if there were no actual role renouncement.

Recommendation

We recommend restricting `renounceOwner` function to be called only by the current owner.

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-11	`withdrawPrincipal` can be called multiple times for the same wallet
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the function Prestaking.sol#L163 of contract `Prestaking`. It is possible to call that function multiple times for the same staking wallet, but after the first call it will have no effect except for the emitted unnecessary event `LogLockupWithdrawal`.

Recommendation

We recommend restricting `withdrawPrincipal` function to be callable only if `w.balance` is not equal to zero.

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

L-12	`updateBeneficiary` can be called on the already withdrawn wallets
Severity	Low
Status	Fixed in d1aa7ee0

Description

The issue is identified within the function Prestaking.sol#L152 of contract `Prestaking`. It is possible to call this function with an already withdrawn staking wallet index as a parameter, which would be an unnecessary action with no effect as the beneficiary address has never been used since the withdrawal.

Recommendation

We recommend adding a check that `w.balance` is not equal to zero to `updateBeneficiary` function not to allow emitting unnecessary events.

Client's commentary

> The issue has been fixed in 27f10741d9b0ca6eda6649d8a040a6b0bcd9f002.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes