Assure DeFi®

THE VERIFICATION GOLD STANDARD

VERIFIED BY ASSURE DEFI ® INTEGRITY ✦ TRUST ✦ CREDIBILITY

Security Assessment

TIXER

Date: 27/06/2025

Audit Status: FAIL

Audit Edition: Advanced+

tixer

ASSURE DEFI®

THE VERIFICATION GOLD STANDARD

Risk Analysis

Vulnerability summary

Classification	Description
🔴 High	High-level vulnerabilities can result in the loss of assets or manipulation of data.
🟠 Medium	Medium-level vulnerabilities can be challenging to exploit, but they still have a considerable impact on smart contract execution, such as allowing public access to critical functions.
🟡 Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.
🟢 Informational	Informational vulnerabilities, code style violations, and informational statements do not affect smart contract execution and can typically be disregarded.

Executive Summary

According to the Assure assessment, the Customer's smart contract is **Insecure.**

Insecure	**Poorly Secured**	**Secured**	**Well Secured**

Scope

Target Code And Revision

For this audit, we performed research, investigation, and review of the TIXER contracts followed by issue reporting, along with mitigation and remediation instructions outlined in this report.

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	AssetBettingV3.sol [SHA256]: eba86a1f15ce4fba3a912bccd84162ac06be78cd5c5e74ccaa33ed94f740967a
Audit Methodology	Static, Manual

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Category	Item
Code review & Functional Review	● Compiler warnings. ● Race conditions and Reentrancy. Cross-function race conditions. ● Possible delays in data delivery. ● Oracle calls. ● Front running. ● Timestamp dependence. ● Integer Overflow and Underflow. ● DoS with Revert. ● DoS with block gas limit. ● Methods execution permissions. ● Economy model. ● Private user data leaks. ● Malicious Event log. ● Scoping and Declarations. ● Uninitialized storage pointers. ● Arithmetic accuracy. ● Design Logic. ● Cross-function race conditions. ● Safe Zeppelin module. ● Fallback function security. ● Overpowered functions / Owner privileges

AUDIT OVERVIEW

◆ ◆ ◆ **HIGH**

1. Fee Logic Missing

Function: -

Issue: FEE_PERCENTAGE & feeCollector declared but never applied, so no fees are ever collected.

Recommendation: Either remove the unused fee parameters or implement fee deduction (for example deduct amount * FEE_PERCENTAGE / BASIS_POINTS in deposit/withdrawFor and send to feeCollector).

2. Unrestricted Emergency Drain

Function: recoverERC20

Issue: Owner can drain all USDC/USDT user deposits at any time, stealing funds.

Recommendation: Restrict recoverERC20 to only non-USDC/USDT tokens, or require a timelock/multisig, or split into recoverFees that only withdraws accrued fees tracked separately.

3. Unbounded Recipients Array

Function: distributeTokens(address,uint256,recipients[])

Issue: Large recipient arrays can exhaust gas and revert, potentially locking funds.

Recommendation: Enforce a maximum recipients.length (for example ≤ 20), or paginate distributions or consider to remove dynamic array and use only hard-coded recipients.

4. No ERC20 Validation

Function: updateToken

Issue: Owner could set a non-ERC20 address, breaking deposits/withdrawals and locking funds.

Recommendation: After newTokenAddress != address(0), perform a low-gas check like IERC20(newTokenAddress).balanceOf(address(this)) to confirm ERC20 compliance (reverting if not).

5. No separate Fee Accounting

Function: -

Issue: If fees are implemented, fee funds mix with user balances, risking accidental withdrawal.

Recommendation: Maintain a distinct mapping(address => uint256) feeBalances; and update it on each fee capture, so that user withdrawals can never touch feeBalances.

6. Centralized Withdrawal Authority

Function: withdrawFor

Issue: Only owner can withdraw for users funds can be censored or permanently locked if owner fails.

Recommendation: Introduce a user-initiated withdraw(uint256 amount) entrypoint gated by off-chain validation or a timelock fallback allowing users to withdraw themselves after X days.

7. Integer Rounding Loss

Function: distributeTokens…hardcoded

Issue: (totalAmount * pct) / 10000 can leave a residual remainder stranded in contract.

Recommendation: Track distributed sum in the loop and, on the last recipient, assign amount = totalAmount - distributed to ensure full distribution.

◆ ◆ ◆ MEDIUM

1. Unnecessary String Storage

Function: deposit/withdrawFor

Issue: Storing a string transactionType in each record costs ~16 k gas more per tx.

Recommendation: Replace string transactionType with enum TxType { DEPOSIT, WITHDRAWAL } to reduce storage and simplify logic.

2. Gas Heavy Arraw Shifting

Function: _recordTransaction

Issue: Looping to shift elements on every tx costs ~45 k extra gas once full.

Recommendation: Use a circular buffer with a head pointer per user, overwriting at head % MAX_TRANSACTIONS to keep costs constant.

3. Overloaded Distribute Functions

Function: -

Issue: Owner may call the wrong overload, leading to unexpected behavior.

Recommendation: Rename the two distributions for example distributeCustomRecipients vs. distributeConfiguredRecipients to avoid ABI confusion.

LOW

1. Missing Event Emission

Function: recoverERC20

Issue: Emergency token recoveries are not logged on-chain.

Recommendation: Emit a Recovered(address indexed token, uint256 amount) event at the end of recoverERC20.

INFORMATIONAL

No informational issues were found.

Testing coverage

During the testing phase, custom use cases were written to cover all the logic of contracts. **Check “Annexes” to see the testing code.*

Pending

Annexes

Testing code:

Pending

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
🔴 High	7					
🟠 Medium	3					
🟡 Low	1					
🟢 Informational						

Assessment Results

Score Results

Review	Score
Global Score	**40/100**
Assure KYC	Not completed
Audit Score	40/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit FAIL

Following our comprehensive security audit of the token contract for the TIXER project, the project did not fulfill the necessary criteria required to pass the security audit.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial adTIXER in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment adTIXER, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment serTIXERs provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any serTIXERs, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The serTIXERs may access, and depend upon, multiple layers of third parties.