Assure DeFi®

THE VERIFICATION GOLD STANDARD

Security Assessment

Cellex

Date: 12/06/2025

Audit Status: PASS

Audit Edition: Advanced

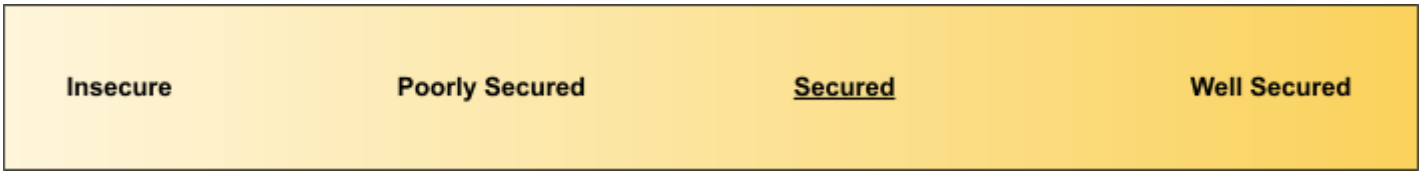
Risk Analysis

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Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.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 **Secured.**



Scope

Target Code And Revision

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

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	https://etherscan.io/token/0x9c4ab3c926ad52e1281f6e8783cb4fe4512c5e3f
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

No high vulnerabilities were found.

MEDIUM

1. Reentrancy Vulnerability in Fee Distribution

Issue: The contract swaps tokens for ETH and then uses Address.sendValue() to forward ETH to each fee-receiver. sendValue issues a low-level call forwarding all available gas, enabling reentrant calls into the contract while inSwapAndLiquify is still true.

Recommendation: Use ReentrancyGuard, set inSwapAndLiquify = false before external calls, adopt pull-payment model.

2. Logical Operator Precedence Bug in Fee Application

Issue: The condition for applying trade fees is written as:

```
else if (from == uniswapV2Pair || to == uniswapV2Pair && from != owner()) {
    _totalFees = tradeFee;
}
```

Due to Solidity's operator precedence (&& binds tighter than ||), this evaluates as:

```
(from == uniswapV2Pair) || (to == uniswapV2Pair && from != owner())
```

Consequently, all buys (where from == pair) incur fees—even if from or to is excluded—while sell-fee exemption via owner address only applies to sells (to == pair).

Recommendation: Add explicit parentheses to reflect intended logic. For example, if the goal is "apply fees on buys or (sells by non-owner)"

```
else if ((from == uniswapV2Pair || to == uniswapV2Pair) && from != owner()) {}
```

3. Unrestricted Slippage in Uniswap Swaps

Issue: The call to Uniswap uses amountOutMin = 0:

```
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
    tokenAmount,
    0,
```

```
    path,
    address(this),
    block.timestamp
);
```

This permits arbitrageurs to sandwich the swap, capturing significant price impact.

Recommendation: Compute a realistic amountOutMin based on on-chain reserves and acceptable slippage (e.g., 1–2%) or alternatively, allow the owner to set a configurable slippage parameter.

4. Missing SafeERC20 Usage in claimStuckTokens

Issue: Withdrawals of ERC-20 tokens rely on IERC20(token).transfer() without checking the returned boolean. Non-compliant tokens that do not revert on failure but return false may cause funds to become irretrievable.

Recommendation: Import and use OpenZeppelin's SafeERC20 library and call IERC20(token).safeTransfer(...) to ensure operations revert on failure.

LOW

1. Owner Can Unboundedly Set Wallet and Swap Thresholds

Issue: The owner may inadvertently set maxWalletSize or maxSwapThreshold to extremely low values (e.g., zero), blocking normal trading, or to astronomically high values, circumventing any intended limit. There are no upper or lower bounds.

Recommendation: Implement sensible min/max bounds.

INFORMATIONAL

No informational issues were found.

Technical Findings Summary

Findings

Vulnerability Level	Total	Mitigated	Not Apply	Acknowledged	Partially Fixed	Fixed
High						
Medium	4					
Low	1					
Informational						

Assessment Results

Score Results

Review	Score
Global Score	**85/100**
Assure KYC	Not completed
Audit Score	85/100

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 PASS

Following our comprehensive security audit of the token contract for the Cellex project, we inform you that the project has met the necessary security standards.

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 adCellex 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 adCellex, 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 serCellexs 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 serCellexs, 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 serCellexs may access, and depend upon, multiple layers of third parties.