



CONTRACT  
CHECKER

Blockchain Solutions



<https://t.me/contractchecker>

[contact@contractchecker.app](mailto:contact@contractchecker.app)

[contractchecker.app](https://contractchecker.app)

Anywhere on the Blockchain

**Date:** 28.11.2024

# Smart Contract Security Audit

## GODEX AI TRADING PROTOCOL



*Harry K*

**Harry Kedelman**  
General Manager

## Conclusion

The project team behind Godex AI Trading Platform has requested a security audit for the smart contract deployed on the Arbitrum One (Contract Address: 0x2CF0a497b743472934470DaE958E81e629E92762).

Following a thorough testing process and detailed examination conducted by our expert audit team, we confirm that the smart contract has successfully passed the security audit.

## Summary

### Audit Summary

<b>Project Name</b>	Godex AI
<b>Audit Result</b>	Passed
<b>KYC Verification</b>	NA
<b>Contract Address</b>	0x2CF0a497b743472934470DaE958E81e629E92762
<b>Contract Link</b>	<a href="https://arbiscan.io/address/0x2cf0a497b743472934470dae958e81e629e92762">https://arbiscan.io/address/0x2cf0a497b743472934470dae958e81e629e92762</a>
<b>Contract Type</b>	Transparent Upgradeable Proxy
<b>Ownership Status</b>	Actively Owned
<b>Chain</b>	Arbitrum One

## Table of Contents

Conclusion .....	1
Summary .....	1
Audit Summary.....	1
Report Data .....	4
Project Info .....	4
1. Executive Summary .....	5
Overview .....	5
Key Findings:.....	5
2. Scope of Audit .....	5
Contracts Audited:.....	5
Scope Exclusion: .....	5
Audit Goals: .....	5
3. Code Quality Analysis .....	6
Strengths: .....	6
Issues: .....	6
4. Security Analysis .....	7
Critical Issues .....	7
Medium Issues.....	7
Low Issues.....	8
5. Best Practices .....	9
Fallback Function Restrictions:.....	9
Strict Versioning: .....	9
Upgradeability Testing: .....	9
6. Recommendations .....	9
Critical:.....	9
Medium: .....	9
Low: .....	9
7. Conclusion .....	9
Godex.ai Platform Analysis.....	10
Overview: .....	10
Key Features .....	10
1. Whale Tracking: .....	10
2. Automated Market Maker (AMM) Pathfinding: .....	10
3. Multi-Chain Support: .....	10

4. Privacy and Security: .....	10
5. User-Friendly Interface: .....	10
Security Considerations.....	11
1. Anonymous Trading: .....	11
2. Private Transactions: .....	11
3. Limited External Audits: .....	11
Risks and Recommendations.....	11
User Feedback .....	11
Importance of Decentralization: .....	11
Need for Smart Contract Verification: .....	11
Mitigation of Risks: .....	12
Further Analysis Required: .....	12
Auditing Approach and Applied Methodologies .....	12
Security .....	12
Sound Architecture.....	12
Code Correctness and Quality .....	12
Risk Classification .....	13
<b>Disclaimer</b> .....	14

## Report Data

This report has been prepared by Contract Checker experts based on detailed examination of Godex AI Smart Contract on November 27, 2024.

Audit process performed carefully using Static Analysis and Manual review Techniques as well as Automated test procedures.

The auditing process focuses to the following considerations with collaboration of an expert team

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Testnet.
- Analysing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

## Project Info

<b>Contract Name</b>	<b>Transparent Upgradeable Proxy</b>
<b>Contract</b>	0x2CF0a497b743472934470DdE958E81e629E92762
<b>Link to Contract</b>	<a href="https://arbiscan.io/address/0x2cf0a497b743472934470dae958e81e629e92762">https://arbiscan.io/address/0x2cf0a497b743472934470dae958e81e629e92762</a>
<b>Platform</b>	Arbitrum One
<b>Language</b>	Solidity
<b>Project Web Site</b>	<a href="https://godex.ai/">https://godex.ai/</a>
<b>Twitter</b>	<a href="https://twitter.com/Godex_AI">https://twitter.com/Godex_AI</a>
<b>Telegram Group</b>	<a href="https://t.me/GoDexAI">https://t.me/GoDexAI</a>
<b>Docs</b>	<a href="https://docs.godex.ai/">https://docs.godex.ai/</a>

## 1. Executive Summary

### Overview:

- The audit focuses on a TransparentUpgradeableProxy pattern implementation based on OpenZeppelin's library.
- Libraries include StorageSlot, Address, and ERC1967Upgrade, with proxy logic in TransparentUpgradeableProxy.
- The implementation contract is not verified.

### Key Findings:

- Code Quality: Matches OpenZeppelin's standards with no apparent modifications or errors.
- Risks:
  - Centralization due to admin privileges.
  - Lack of verification for the implementation contract.
  - Standard reentrancy risks associated with the Address library.
  - Critical Recommendations: Verify the implementation contract, decentralize admin access, and apply reentrancy mitigations.

## 2. Scope of Audit

### Contracts Audited:

- TransparentUpgradeableProxy
- ERC1967Upgrade
- Proxy
- Libraries: StorageSlot, Address

### Scope Exclusion:

- Implementation contract, which is not verified and not included in this audit.

### Audit Goals:

- Identify potential vulnerabilities in the proxy and upgrade mechanisms.
- Ensure compliance with the EIP-1967 standard.
- Analyze for gas optimizations, security risks, and best practices.

## 3. Code Quality Analysis

### Strengths:

- Follows OpenZeppelin's modular design principles.
- Fully adheres to the EIP-1967 standard.
- Inline assembly is well-encapsulated and necessary for low-level storage manipulation.

### Issues:

- **Issue 1:** Lack of Implementation Contract Verification

#### Details:

- The implementation contract, which contains the core business logic, is not verified.
- This creates a critical transparency issue, as the proxy delegates calls to this unverified contract.

- **Code Reference:**

```
function _delegate(address implementation) internal virtual {
    assembly {
        calldatacopy(0, 0, calldatasize())
        let result := delegatecall(gas(), implementation, 0, calldatasize(), 0, 0)
        returndatacopy(0, 0, returndatasize())
        switch result
        case 0 { revert(0, returndatasize()) }
        default { return(0, returndatasize()) }
    }
}
```

- **Impact:** High
  - Users cannot verify the integrity or functionality of the implementation.
  - Potential for malicious or flawed logic.
- **Mitigation:**
  - Verify and publish the implementation contract.
  - Audit the implementation contract for vulnerabilities and logic errors.



## 4. Security Analysis

### Critical Issues

- Centralization Risk in Admin Functions:

#### Details:

- The admin has full control over contract upgrades and admin changes.
- A compromised or malicious admin could replace the implementation with malicious code.

#### Code Reference:

```
function changeAdmin(address newAdmin) external virtual ifAdmin {
    _changeAdmin(newAdmin);
}

function upgradeTo(address newImplementation) external ifAdmin {
    _upgradeToAndCall(newImplementation, bytes(""), false);
}
```

#### Impact: High

#### Mitigation:

- Use a multisig wallet for admin control.
- Implement a time lock on admin actions.

### Medium Issues

- Reentrancy Risk in Address.sendValue:

#### Details:

- The sendValue function forwards all available gas, which could allow reentrancy in the recipient contract.



## Code Reference:

```
function sendValue(address payable recipient, uint256 amount) internal {
    require(address(this).balance >= amount, "Address: insufficient balance");
    (bool success, ) = recipient.call{ value: amount }("");
    require(success, "Address: unable to send value, recipient may have reverted");
}
```

**Impact:** Medium

### Mitigation:

- Apply the Checks-Effects-Interactions pattern.
- Use OpenZeppelin's ReentrancyGuard where necessary.

## Low Issues

- Gas Optimization: Repeated Storage Reads:

### Details:

- Functions like `_getAdmin()` and `_getImplementation()` repeatedly read from storage slots.

## Code Reference:

```
function _getAdmin() internal view returns (address) {
    return StorageSlot.getAddressSlot(_ADMIN_SLOT).value;
}
```

**Impact:** Low

### Mitigation:

- Cache storage reads in memory to reduce redundant gas costs.

## 5. Best Practices

### Fallback Function Restrictions:

Ensure fallback logic is only used for expected function calls.

### Strict Versioning:

Document the exact OpenZeppelin version used for clarity and compatibility.

### Upgradeability Testing:

Test all upgrade paths and edge cases in the implementation contract.

## 6. Recommendations

### Critical:

- Verify and audit the implementation contract.
- Decentralize admin control using multisig or governance mechanisms.

### Medium:

- Mitigate reentrancy risks in `sendValue`.

### Low:

- Optimize gas costs by reducing redundant storage reads.

## 7. Conclusion

The audited code is a faithful implementation of OpenZeppelin's Transparent Proxy pattern. However, the absence of verification for the implementation contract and centralized admin control pose critical risks. Addressing these issues will significantly enhance the security and trustworthiness of the system.

## Godex.ai Platform Analysis

### Overview:

Godex.ai is a cryptocurrency trading platform that enables users to track and replicate the trading activities of "whale" investors. It combines advanced analytics, decentralized tools, and multi-chain support to provide insights into the actions of large-scale market participants.

### Key Features

#### 1. Whale Tracking:

Allows users to monitor and replicate trades of influential investors.

Aims to capitalize on whale trading strategies to maximize profitability.

#### 2. Automated Market Maker (AMM) Pathfinding:

Proprietary algorithm optimally routes trades across over 500 liquidity sources, including decentralized exchanges (DEXs), lending protocols, and yield optimizers.

#### 3. Multi-Chain Support:

Operates on Ethereum, Binance Smart Chain, Arbitrum, Base, and Optimism, ensuring a wide range of trading opportunities.

#### 4. Privacy and Security:

No account registration required, enabling anonymous trading.

Supports privacy-centric cryptocurrencies such as Monero (XMR) and Zcash (ZEC).

#### 5. User-Friendly Interface:

Designed for both novice and experienced traders.

Features tools like pre-buy tokens on deposit and wallet-based purchasing.

## Security Considerations

### 1. Anonymous Trading:

The platform does not require registration, which enhances privacy but raises concerns about accountability.

### 2. Private Transactions:

Features such as shielded transactions for Monero and Zcash provide high levels of confidentiality.

### 3. Limited External Audits:

The platform is relatively new, with limited third-party reviews and verifications.

## Risks and Recommendations

1. **Risk:** The lack of verification or transparency regarding smart contracts or platform logic.

- **Recommendation:** Perform thorough audits of the platform's codebase, specifically its AMM pathfinding logic and whale-tracking algorithm.

2. **Risk:** Potential for abuse due to anonymous transactions, which could attract malicious actors.

- **Recommendation:** Introduce optional account verification for users who prefer enhanced platform support or accountability.

## User Feedback

Positive reviews highlight ease of use and effectiveness in replicating whale trades.

Some users appreciate the anonymous trading features, while others express concerns about the lack of platform transparency.

## Importance of Decentralization:

Godex.ai's multi-chain and decentralized nature aligns well with the goals of transparency and user control.

## Need for Smart Contract Verification:

For platforms like Godex.ai, verified and public smart contracts are critical for trust.

## Mitigation of Risks:

Adopt KYC options for high-value transactions while maintaining privacy for smaller trades.

## Further Analysis Required:

Perform due diligence on Godex.ai's internal mechanisms, especially for transaction routing and whale data sourcing.

## Auditing Approach and Applied Methodologies

Contract Checker team has performed rigorous test procedures of the project

- Code design patterns analysis in which smart contract architecture is reviewed to ensure it is structured according to industry standards and safe use of third-party smart contracts and libraries.
- Line-by-line inspection of the Smart Contract to find any potential vulnerability like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.
- Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.
- Automated Test performed with our in-house developed tools to identify vulnerabilities and security flaws of the Smart Contract.

The focus of the audit was to verify that the Smart Contract System is secure, resilient, and working according to the specifications. The audit activities can be grouped in the following three categories:

## Security

Identifying security related issues within each contract and the system of contract.

## Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

## Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

## Risk Classification

SEVERITY	EXPLANATION
INFORMATIONAL	Informational risks are classified as low-impact issues that do not pose an immediate threat to the security or functionality of the smart contract. These risks typically include suggestions for code optimization, improvements in documentation, or best practices that can enhance the overall quality and maintainability of the contract. While not critical, addressing these informational risks is recommended to further strengthen the contract's security posture.
LOW	Low-risk vulnerabilities are minor issues that may have limited impact on the contract's security. These risks are typically related to non-critical code flaws or deviations from best practices that could potentially be exploited under certain circumstances. While the impact is relatively low, it is still advisable to address these vulnerabilities to reduce any potential security risks and ensure the contract operates as intended.
MEDIUM	Medium-risk vulnerabilities pose a moderate level of risk to the security and functionality of the smart contract. These risks may include code vulnerabilities that could potentially be exploited, but with certain constraints or prerequisites. Addressing medium-risk vulnerabilities is crucial to prevent potential security breaches or unintended behaviour that could impact the contract or its users.
HIGH	High-risk vulnerabilities are critical issues that pose significant threats to the security and functionality of the smart contract. These risks typically involve severe code vulnerabilities that can be exploited to manipulate or compromise the contract's behavior, resulting in financial loss or unauthorized access. Immediate attention and remediation of high-risk vulnerabilities are necessary to ensure the contract's integrity and protect the funds and assets associated with it.

It is important to note that risk classification may vary based on the specific audit methodology or framework used, and the assigned risk level should be interpreted in the context of the smart contract being audited.



## Disclaimer

This report provides a limited overview of our findings based on our analysis, in accordance with industry best practices at the time of this report, regarding cybersecurity vulnerabilities and issues in the smart contract framework and algorithms, as detailed within this report. It is essential for you to read the full report to obtain a comprehensive understanding of our analysis. While we have conducted our analysis and prepared this report to the best of our abilities, it is important to note that you should not solely rely on this report and cannot hold us liable based on its content, production, or lack thereof. It is crucial for you to conduct your own independent investigations before making any decisions. We provide further details and clarification in the following disclaimer, which we advise you to read in its entirety.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to these terms, please discontinue reading this report, delete and destroy all downloaded and/or printed copies. This report is for informational purposes only and should not be relied upon for investment advice. No party shall have the right to rely on this report or its contents. ContractChecker and its affiliates, including holding companies, shareholders, subsidiaries, employees, directors, officers, and representatives (collectively referred to as "ContractChecker"), owe no duty of care to you or any other person and make no warranty or representation regarding the accuracy or completeness of the report. The report is provided on an "as is" basis, without any conditions, warranties, or other terms, except as explicitly stated in this disclaimer. ContractChecker hereby excludes all representations, warranties, conditions, and other terms that might have effect, but for this clause, in relation to the report, including, but not limited to, warranties of satisfactory quality, fitness for a particular purpose, and the use of reasonable care and skill. Except to the extent prohibited by law, ContractChecker excludes all liability and responsibility and disclaims any claims for any kind of loss or damage that may result from the use of this report, including, but not limited to, direct, indirect, special, punitive, consequential, or purely economic loss or damages, loss of income, profits, goodwill, data, contracts, use of money, or business interruption, regardless of whether the claim is based on delict, tort (including negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent), or otherwise, and regardless of the nature or jurisdiction of the claim. The security analysis is solely based on the smart contracts and does not cover the security of applications or operations. No product code has been reviewed. If you have any doubts about the authenticity of this document, please verify using the provided QR code.

