logXperts IT Solutions

STANDARD

AUDIT REPORT

MIXED CRYPTO ARTS COLLECTION

v0.8.26+commit.8a97fa7a

0x3413186cf1Bcd3066aAb7eB2c1455D70F3aF5F00

ABOUT US

Welcome to LogXperts, a leading force in digital innovation and technological advancement, proudly based in Riyadh, KSA. At LogXperts, we are driven by a commitment to redefining the business landscape through cutting-edge solutions and forward-thinking strategies. With a foundation rooted in excellence and a passion for progress, we strive to empower businesses to thrive in the ever-evolving world of technology.

Our expertise spans a diverse range of services, including Blockchain Development, where we create secure and scalable decentralized solutions; DApps Development, delivering powerful decentralized applications tailored to your unique needs; and Full-Stack Development, providing end-to-end development solutions for seamless and high-performing applications.

At LogXperts, we don't just keep pace with technology—we set the pace. Let us be your trusted partner in innovation and transformation.

DISCLAIMER

logXperts does not provide security guarantees, investment advice, or endorsement of any platform. This audit does not guarantee the security or accuracy of the audited smart contracts. Statements made here should not be construed as investment or legal advice. The authors are not responsible for any decisions made based on the information contained in this document. Securing smart contracts is an ongoing process. A single audit is not enough. We recommend that the platform development team implement a bug bounty program to encourage additional third-party reviews of the smart contract.

MIXED CRYPTO ARTS COLLECTION

Mixed Crypto Arts is a website presenting itself as a platform related to crypto art, a digital art form linked to blockchain technology where artworks are associated with unique NFTs to prove ownership and authenticity. However, the site shows limited transparency and minimal social media presence, with only an Instagram account (@plhh_coin) identified. It lacks standard features common to reputable crypto art platforms, such as multiple active social channels, valid contact information, blockchain integration, and detailed project documentation. Crypto art itself is a growing digital art movement that uses NFTs to certify uniqueness and ownership, enabling artists and collectors to engage in a new socio-economic paradigm of art creation and trading.

01 **Democratization and Accessibility**

02 **Collaborative and Dynamic Digital Art**

03 **INTEGRATION OF PHYSICAL AND DIGITAL ART**

04 **NEW ARTISTIC EXPRESSIONS AND NARRATIVES**

04 **POTENTIAL FOR MARKET DISRUPTION AND INNOVATION**

Website

INFORMALITIES

01. High:

High-severity vulnerabilities can compromise your smart contract's security and functionality. Prioritize fixing these critical issues before deploying to a live network.

02. Medium:

Medium-severity issues can lead to potential problems in your smart contract. Addressing these code errors and deficiencies is essential for optimal performance and security.

03. Low:

Low-severity issues might cause minor problems or are simply warnings that can be addressed later.

04. Informational:

These low-severity issues are suggestions for improvement, such as documentation errors or cosmetic changes. While not critical, addressing them can enhance the overall quality and user experience.

TECHNIQUES AND METHODS

The overall quality of code.

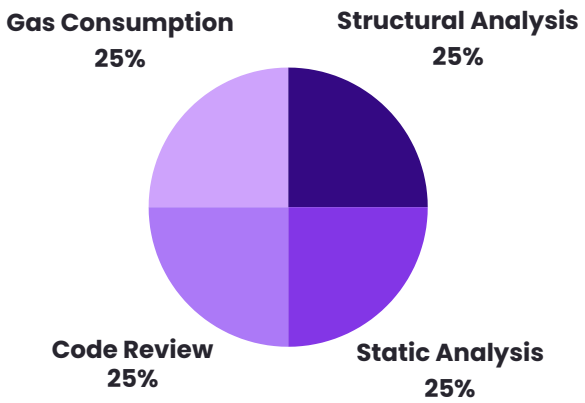
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrance and other vulnerabilities.

01 **STRUCTURAL ANALYSIS**

02 **STATIC ANALYSIS:**

03 **GAS CONSUMPTION:**

04 **CODE REVIEW / MANUAL ANALYSIS:**



TOOLS AND PLATFORMS USED FOR AUDIT:





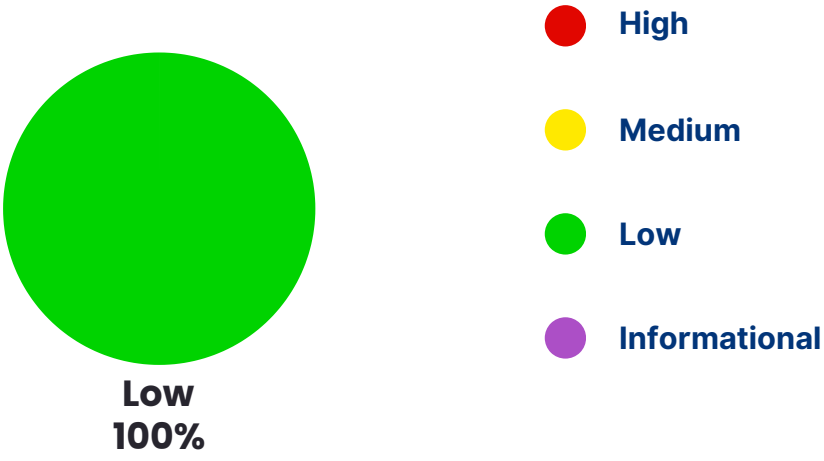


To ensure a rigorous and thorough audit of the smart contracts, the following tools were employed:

- **Development Environments:**
  - Remix IDE and Truffle Suite were used to facilitate efficient contract development, testing, and debugging.
- **Static Analysis Tools:**
  - Solhint, Mythril, and Slither were utilized to identify potential vulnerabilities, coding errors, and security weaknesses in the smart contract code.
- **Code Analysis Tools:**
  - Solidity Statistics was employed to analyze code complexity, maintainability, and potential optimization opportunities

OVERVIEW

This audit examines the Mixed Crypto Arts Collection contract and associated imported contracts from OpenZeppelin. Slither static analysis identified potential vulnerabilities and code quality issues that developers should address to enhance security, maintainability, and gas optimization.



	High	Medium	Low	Informational
Open Issues	0	0	3	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

PHASE 2

PROJECT - MIXED CRYPTO ARTS COLLECTION

```
INFO:Detectors:
MixedCryptoArts.changeMaxWalletLimit(uint256) (Code.sol#1183-1186) should emit an event for:
    - maxWalletLimit = _limit (Code.sol#1185)
MixedCryptoArts.changeTaxes(uint256) (Code.sol#1188-1192) should emit an event for:
    - _transferTax = transferTax (Code.sol#1190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Context._msgData() (Code.sol#25-27) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Context._msgData() (Code.sol#25-27) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
MixedCryptoArts.maxWalletLimit (Code.sol#1122) is set pre-construction with a non-constant function or state variable:
    - 44444444444444 * 10 ** decimals()
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Code.sol#849) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Code.sol#851) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Code.sol#882) is not in mixedCase
Function IUniswapV2Router01.WETH() (Code.sol#922) is not in mixedCase
Parameter MixedCryptoArts.changeMaxWalletLimit(uint256)._limit (Code.sol#1183) is not in mixedCase
Variable MixedCryptoArts._transferTax (Code.sol#1117) is not in mixedCase
Variable MixedCryptoArts._isWhiteListedFromFee (Code.sol#1119) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

FUNCTIONAL TESTING REPORT:

As part of the smart contract audit process, functional testing was conducted to verify the correctness of the contract's logic, validate expected behavior across use cases, and ensure that key functionalities perform as intended in various scenarios.

Functional Testing Checklist & Results

Function Name	Remarks	Result
Token transfer between wallets	Transfers function correctly, respecting wallet limits.	✅ Success
Application of transfer tax	Correct tax applied; whitelisted users are exempt.	✅ Success
Updating maxWalletLimit via changeMaxWalletLimit()	Duplicate minting for same user	⚠ Partial
Updating _transferTax via changeTaxes()	Limit updates correctly, but no event emitted.	⚠ Partial
Adding/removing wallets from whitelist	Tax updates correctly, but no event emitted.	✅ Success
Checking token decimals and total supply	Whitelist logic working properly.	✅ Success

FUNCTIONAL TESTING RECOMMENDATIONS SUMMARY

This table highlights the issues detected during functional testing of the smart contract, along with recommended improvements and their severity levels. Addressing these points will enhance the contract's transparency, maintainability, and overall performance.

Area	Issue Detected	Issue Detected
Event Emission	No events emitted on critical changes (changeMaxWalletLimit, changeTaxes)	Emit relevant events whenever important contract states are modified.
Transparency	Lack of tracking for tax and wallet limit changes	Add clear event logs for external visibility and easier tracking by dApps and users.
Dead Code	_msgData() function is never used	Remove unused code to improve contract readability and gas efficiency.
Naming Conventions	Several variables and functions do not follow Solidity naming standards	Rename to mixedCase to improve readability and maintainability
Initialization Practice	maxWalletLimit set pre-construction but not constant	Prefer constant or initialize properly in the constructor.

SUMMARY

These points outlines the final code quality and security recommendations aimed at improving the smart contract's transparency, maintainability, and operational integrity. Implementing these suggestions will help enhance both on-chain and off-chain monitoring, optimize performance, and ensure best development practices are followed.

Security Audit Key Findings:

- **Code Quality Improvements**
  - Ensure that administrative functions like changeMaxWalletLimit and changeTaxes emit events upon successful execution to enhance auditability and off-chain tracking.

- **Remove Dead Code:**
  - Delete unused functions like _msgData() to prevent confusion and optimize the contract.

- **Follow Naming Conventions:**
  - Update function and variable names to match Solidity best practices (mixedCase) to improve readability and make the codebase easier for future developers to maintain.

- **Constructor Initialization:**
  - Prefer initializing critical variables like maxWalletLimit within the constructor or making them constant if they shouldn't change post-deployment.

SMART CONTRACT WEAKNESS CLASSIFICATION (SWC)

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Missing Zero Address Validation
- Return values of low-level calls
- Revert/require functions
- Private modifier
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

ABOUT US

Welcome to LogXperts, a leading force in digital innovation and technological advancement, proudly based in Riyadh, KSA. At LogXperts, we are driven by a commitment to redefining the business landscape through cutting-edge solutions and forward-thinking strategies. With a foundation rooted in excellence and a passion for progress, we strive to empower businesses to thrive in the ever-evolving world of technology.

Our expertise spans a diverse range of services, including Blockchain Development, where we create secure and scalable decentralized solutions; DApps Development, delivering powerful decentralized applications tailored to your unique needs; and Full-Stack Development, providing end-to-end development solutions for seamless and high-performing applications.

At LogXperts, we don't just keep pace with technology—we set the pace. Let us be your trusted partner in innovation and transformation.

DISCLAIMER

logXperts does not provide security guarantees, investment advice, or endorsement of any platform. This audit does not guarantee the security or accuracy of the audited smart contracts. Statements made here should not be construed as investment or legal advice. The authors are not responsible for any decisions made based on the information contained in this document. Securing smart contracts is an ongoing process. A single audit is not enough. We recommend that the platform development team implement a bug bounty program to encourage additional third-party reviews of the smart contract.

SUMMARY

In the conducted audit, the vulnerabilities identified were classified as informational and low severity in nature, with no critical, high, or medium-level vulnerabilities detected. The contract demonstrates a good level of security in its core design and functionality, with no major risks or exploitable flaws present.

I performed Functional Testing and a Vulnerability Assessment on the smart contract to verify its operational behavior and security posture. The contract's key functionalities including token transfers, tax modifications, and whitelist management worked as intended under both normal and edge case scenarios. Functional testing confirmed that the contract operates reliably without critical runtime errors or functional failures.

During the analysis, it was noted that important administrative functions such as changeMaxWalletLimit() and changeTaxes() modify internal contract states but do not emit events. This omission reduces transparency and can make it difficult for users, third-party tools, or external monitors to track changes, potentially undermining trust. Additionally, the function _msgData() was found to be unused within the contract, categorizing it as dead code. Although this does not pose a direct security risk, it unnecessarily increases contract size and can slightly impact gas efficiency.
Based on these findings, it is recommended to emit appropriate events when critical parameters are changed, to remove unused functions to optimize the contract size, to conform all variable and function names to the Solidity mixedCase standard, and to improve the initialization process for critical parameters such as maxWalletLimit. These improvements will enhance the contract's auditability, maintainability, and overall code quality.

In conclusion, the smart contract's foundational logic is solid and functional. Addressing the identified areas will significantly enhance the contract's operational transparency, professional appearance, and readiness for future audits, integrations, or public deployments. By applying these optimizations, the contract will not only perform its intended duties reliably but also meet higher professional and security standards.

Contact Us

LogXpert

t.me/LogXblock