QuillAudits

Audit Report
May, 2024

For

0X

Table of Content

Executive Summary

Project Name

OXME

Overview

The OXME smart contract is a capped ERC20 token contract that splits its supply among the Miners, Referrals (which together contribute towards the Royalty) and Investors. Each of the allocated tiers are succinctly separated from each other with limits imposed on their supply and roles assigned to allow specific users to mint tokens.

Only 1% of the supply is minted at deployment time to the provided investorAddress. There is no cliff or vesting period for the other tokens to get minted by.

Timeline

28th April, 2024 - 1st May, 2024

Updated Code Received

2nd May 2024

Second Review

2nd May 2024

Method

Manual Review, Functional Testing, Automated Testing, etc.
All the raised flags were manually reviewed and re-tested to identify any false positives.

Audit Scope

The scope of this audit was to analyse the OXME codebase for quality, security, and correctness.

Source Code

https://github.com/helius-work/OXME/blob/development/smart-contract/contracts/OXME.sol

Contracts In-Scope

smart-contract/

- contracts/OXME.sol

Branch

develop

Executive Summary

Contracts Out of Scope

In-scope contract has been audited by QuillAudits. However, these contracts inherit functionality from out-of-scope Smart contracts that were not audited. Vulnerabilities in unaudited contracts could impact in-scope Smart Contracts functionality. QuillAudits is not responsible for such vulnerabilities.

Below are Out of Scope Contracts:

- OpenZeppelin contracts: ERC20.sol, ERC20Capped.sol, AccessControl.sol

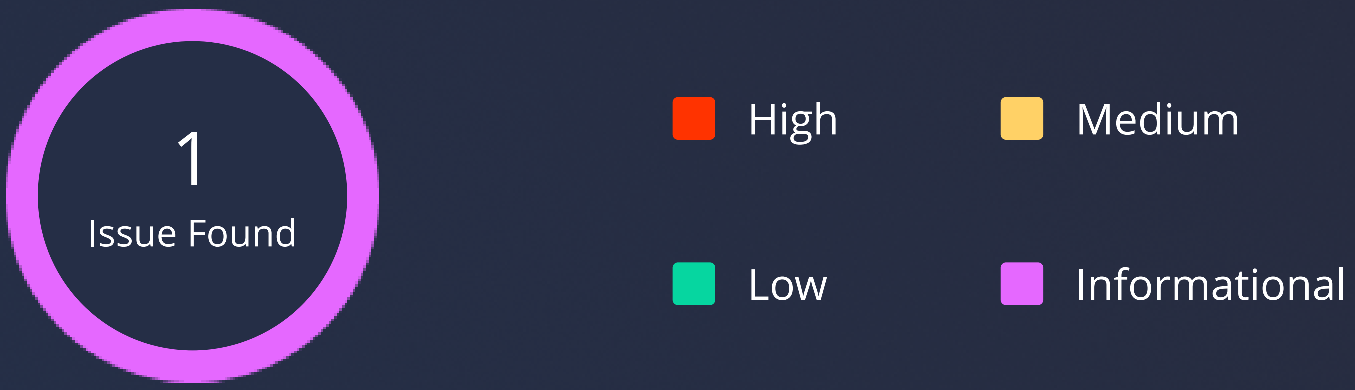
Fixed In

https://github.com/helius-work/OXME/commit/b78638e064e392a426753f1ae19d1144b24d42dd

Mainnet Address

https://arbiscan.io/address/0x614cD634269f5133D10da79C35B36dD9C0f4Fe47

Number of Security Issues per Severity



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	1

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ DoS with Block Gas Limit
- ✓ Transaction-Ordering Dependence
- ✓ Use of tx.origin
- ✓ Exception disorder
- ✓ Gasless send
- ✓ Balance equality
- ✓ Byte array
- ✓ Transfer forwards all gas
- ✓ ERC20 API violation
- ✓ Compiler version not fixed
- ✓ Redundant fallback function
- ✓ Send instead of transfer
- ✓ Style guide violation
- ✓ Unchecked external call
- ✓ Unchecked math
- ✓ Unsafe type inference
- ✓ Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- 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's standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Manual Review, Slither, Hardhat.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Informational Issues

1. Gas Optimization: Variables can be marked immutable

Path

OXME.sol

Function

constructor()

Description

In Solidity, variables take up storage slots that are relatively expensive to access and read from. Variables that can be set before or during deployment, and cannot be changed are preferably marked as immutable to save the variables in the contract bytecode thereby making it less expensive to access and read from.

Recommendation

The following variables set during deployment in the constructor can be marked as immutable: **miningMintCap**, **referralMintCap**, **royaltyMintCap**, **investorsMintCap**, **investorAddress**.

Status

Resolved

Functional Tests Cases

- ✓ Mint within the specified token caps
- ✓ Mint all tokens at any time

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the OXME codebase. We performed our audit according to the procedure described above.

An issue of Informational severity was found, which OXME Team Fixed, and suggestions and best practices were also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in OXME smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of OXME smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the OXME to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we’ve secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project’s security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M
Lines of Code Audited

Follow Our Journey

Audit Report
May, 2024

For

0X

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com