Heesho Wavefront

Security Audit Report

June 13, 2024

Contents

1 Introduction

1.1 About Wavefront

`Wavefront` is a `meme` token issuance and presale platform that incorporates a decentralized exchange (DEX) feature, providing a one-stop solution for the issuance and trading of `meme` tokens. This platform facilitates seamless interactions between token creators and traders, enhancing the accessibility and efficiency of `meme` token markets.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/Heesho/wavefront/tree/AlternativeFeeStructure
- CommitID: 9cac7ad

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/Heesho/wavefront/tree/AlternativeFeeStructure
- CommitID: 7e9d5f7

Note that this audit only covers the `MemeFactory.sol`, `WaveFrontFactory.sol`, `WaveFrontTreasury.sol`, and `WaveFrontRouter.sol` contracts.

2 | Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the `Wavefront` project Throughout this audit, we identified a total of 3 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	1	1	-	-
Informational	1	-	-	1
Undetermined	1	-	-	1

3 | Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Description
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

[U-1] Suggested Access Control for Meme::donate()/burn()

Target	Category	IMPACT	LIKELIHOOD	STATUS
MemeFactory.sol	Security	Undetermined	Low	Addressed

The donate() function allows anyone to donate `base` token to the pool, which can subsequently increase the price of `meme` token. This enables malicious actors to manipulate the price of meme token at will, potentially introducing unforeseen risks. Considering that ordinary users do not have the intention to donate, it is recommended to implement necessary access control for this function to prevent potential misuse by malicious actors.

Meme::burn()/donate()

```
411 function burn(uint256 amount)
412     public
413     notZeroInput(amount)
414 {
415     uint256 savedMaxSupply = maxSupply;
416     uint256 savedReserveMeme = reserveMeme;
417     if (savedMaxSupply > savedReserveMeme) {
418         uint256 reserveBurn = savedReserveMeme.mulWadDown(amount).divWadDown(
                savedMaxSupply - savedReserveMeme);
419         reserveMeme -= reserveBurn;
420         maxSupply -= (amount + reserveBurn);
421         emit Meme__ReserveMemeBurn(reserveBurn);
422     } else {
423         maxSupply -= amount;
424     }
425     _burn(msg.sender, amount);
426     emit Meme__Burn(msg.sender, amount);
427 }

429 function donate(uint256 amount)
430     external
431     nonReentrant
432 {
433     emit Meme__Donated(msg.sender, amount);
434     IERC20(base).safeTransferFrom(msg.sender, address(this), amount);
435     _add(amount);
436 }
```

Remediation Apply necessary access control for the `donate()/burn()` functions.

[L-1] Potential Risks Associated with Centralization

Target	Category	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Security	Low	Low	Acknowledged

In the `Wavefront` implementation, the existence of a privileged `owner` account introduces centralization risks, as it holds significant control and authority over critical operations governing the protocol. In the following, we show the representative function potentially affected by the privileges associated with the privileged account.

MemeFactory::setWaveFrontFactory()

```
635 /**
636   * @dev Allows the WaveFrontFactory contract to update the address of the
        WaveFrontFactory contract.
637   * @param _waveFrontFactory The new address of the WaveFrontFactory contract.
638   */
639 function setWaveFrontFactory(address _waveFrontFactory)
640     external
641     onlyOwner
642 {
643     waveFrontFactory = _waveFrontFactory;
644 }
```

Remediation To mitigate the identified issue, it is recommended to introduce multi-sig mechanism to undertake the role of the privileged account. Moreover, it is advisable to implement timelocks to govern all modifications to the privileged operations.

Response By Team This issue has been confirmed by the team.

[I-1] Removal of Redundant Code

Target	Category	IMPACT	LIKELIHOOD	STATUS
WaveFrontRouter.sol	Coding Practices	N/A	N/A	Addressed

The `WaveFrontRouter::redeem()` function enables users to withdraw `meme` tokens bought during the presale phase. It calls `PreMeme::redeem()` (line 135) to distribute these tokens directly to users. However, it redundantly tries to transfer tokens again (line 137) from the `WaveFrontRouter` contract, despite `PreMeme::redeem()` having already completed the transfer. This redundancy in code could lead to inefficiencies and potential confusion regarding the token flow within the contract.

WaveFrontRouter::redeem()

```
function redeem(address meme) external {
    address preMeme = IMeme(meme).preMeme();
    if (block.timestamp > IPreMeme(preMeme).endTimestamp() && !IPreMeme(preMeme)
        .ended()) {
        IPreMeme(preMeme).openMarket();
        emit WaveFrontRouter__MarketOpened(meme, IPreMeme(preMeme).
            totalBaseContributed(), IPreMeme(preMeme).totalMemeBalance());
    }
    IPreMeme(preMeme).redeem(msg.sender);
    uint256 memeBalance = IERC20(meme).balanceOf(address(this));
    IERC20(meme).transfer(msg.sender, memeBalance);
    emit WaveFrontRouter__Redeemed(meme, msg.sender, memeBalance);
}
```

PreMeme::redeem()

```
function redeem(address account) external nonReentrant {
    if (!ended) revert PreMeme__InProgress();
    uint256 contribution = account_BaseContributed[account];
    if (contribution == 0) revert PreMeme__NotEligible();
    account_BaseContributed[account] = 0;
    uint256 memeAmount = totalMemeBalance.mulWadDown(contribution).divWadDown(
        totalBaseContributed);
    IERC20(meme).safeTransfer(account, memeAmount);
    emit PreMeme__Redeemed(meme, account, memeAmount);
}
```

Remediation Remove redundant code in the buy(), sell(), createMeme(), and redeem() functions.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Name	AstraSec Team
Phone	+86 176 2267 4194
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI