



SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



PROJECT: WEALTH MOUNTAIN

DATE: July 18, 2022



www.safuaudit.com

INTRODUCTION

Client	Wealth Mountain
Language	Solidity
Contract address	0x9B9C918FAC2DFCCaFff3B95b4f46FD9A5D9D701b
Owner	0xC79C0260bd38276060765337Be66f998a34cb19b
Deployer	0xC79C0260bd38276060765337Be66f998a34cb19b
SHA1-Hash	cecf96a08569da0548d5eb4a1ca406218fc507aa
Platform	Binance Smart Chain
Compiler	v0.8.13+commit.abaa5c0e
Optimization	No with 200 runs
Website	https://wealthmountain.app/
Telegram	https://t.me/+SpoLt1tttkU0ZTI5



TABLE OF CONTENTS

01 INTRODUCTION

Introduction

Approach

Risk classification

02 CONTRACT INSPECTION

Contract Inspection

Inheritance Tree

Owner privileges

03 MANUAL ANALYSIS

Manual analysis

04 FINDINGS

Vulnerabilities Test

Findings list

Issues description

05 WEBSITE

Website Audit

06 CONCLUSIONS

Disclaimer

Rating

Conclusion



APPROACH



Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.



Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.



Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
 - Back-doors
 - Vulnerability
 - Accuracy
 - Readability
-



Tools

- Remix IDE
- Mythril
- Open Zeppelin Code Analyzer
- Solidity Code Compiler
- Hardhat



RISK CLASSIFICATION

CRITICAL

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

MEDIUM

Issues on this level could potentially bring problems and should eventually be fixed.

MINOR

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

INFORMATIONAL

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.



OVERVIEW

Fees

- Fixed fee of 10% on every stake.
- No collecting fees.
- Unstaking fees are gradually decreasing:
 - 10% in the first 10 days
 - 8% between 10 and 20 days
 - 5% between 20 and 50 days
 - 2% after 50 days

Fees privileges

- Owner can't modify fees

Ownership

- Owned, but no privileged functions

Pause function

- Can't pause staking

Blacklist

- Can't blacklist



CONTRACT INSPECTION 🔍

Imported contracts or frameworks used:

```
| **IERC20** | Interface |   |||  
| **Address** | Library |   |||  
| **SafeERC20** | Library |   |||  
| **SafeMath** | Library |   |||  
| **WealthMountainBSC** | Implementation |   |||
```

Tested Contract File:

```
| File Name | SHA-1 Hash |  
|-----|-----|  
| WealthMountain.sol | cecf96a08569da0548d5eb4a1ca406218fc507aa |
```

```
| **WealthMountainBSC** | Implementation |   |||  
| L | <Constructor> | Public ! | ● | NO ! |  
| L | stakeStablecoins | Public ! | 💰 | NO ! |  
| L | userInfo | External ! | | NO ! |  
| L | withdrawDivs | Public ! | ● | NO ! |  
| L | withdrawInitial | Public ! | ● | NO ! |  
| L | withdrawRefBonus | Public ! | ● | NO ! |  
| L | stakeRefBonus | Public ! | ● | NO ! |  
| L | calcddiv | Public ! | | NO ! |  
| L | compound | Public ! | ● | NO ! |
```

Symbol	Meaning
●	Function can modify state
💰	Function is payable
🔒	Private function
🔓	Internal function
NO !	Function has no modifier



INHERITANCE TREE



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.



MANUAL FUNCTIONS ANALYSIS

The contract is verified to check if functions do and work as they should and malicious code is not inserted.

	Tested	Result
Transfer	Yes	N/A
Total Supply	Yes	N/A
Buy Back	Yes	N/A
Burn	Yes	N/A
Mint	Yes	N/A
Rebase	Yes	N/A
Pause	Yes	N/A
Blacklist	Yes	N/A
Lock	Yes	N/A
Max Transaction	Yes	N/A
Transfer Ownership	Yes	N/A
Renounce Ownership	Yes	N/A



VULNERABILITIES TEST

ID	Description	
V-01	Function Default Visibility	Passed
V-02	Integer Overflow and Underflow	Passed
V-03	Outdated Compiler Version	Passed
V-04	FloatingPragma	Minor
V-05	Unchecked Call Return Value	Passed
V-06	Unprotected Ether Withdrawal	Passed
V-07	Unprotected SELF-DESTRUCT Instruction	Passed
V-08	Re-entrancy	Passed
V-09	State Variable Default Visibility	Minor
V-10	Uninitialized Storage Pointer	Passed
V-11	Assert Violation	Passed
V-12	Use of Deprecated Solidity Functions	Passed
V-13	Delegate Call to Untrusted Callee	Passed
V-14	DoS with Failed Call	Passed
V-15	Transaction Order Dependence	Passed
V-16	Authorization through tx.origin	Passed
V-17	Block values as a proxy for time	Passed



V-18	Signature Malleability	Passed
V-19	Incorrect Constructor Name	Passed
V-20	Shadowing State Variables	Passed
V-21	Weak Sources of Randomness from Chain Attributes	Passed
V-22	Missing Protection against Signature Replay Attacks	Passed
V-23	Lack of Proper Signature Verification	Passed
V-24	Requirement Violation	Passed
V-25	Write to Arbitrary Storage Location	Passed
V-26	Incorrect Inheritance Order	Passed
V-27	Insufficient Gas Griefing	Passed
V-28	Arbitrary Jump with Function Type Variable	Passed
V-29	DoS With Block Gas Limit	Passed
V-30	Typographical Error	Passed
V-31	Right-To-Left-Override control character (U+202E)	Passed
V-32	Presence of unused variables	Passed
V-33	Unexpected Ether balance	Passed
V-34	Hash Collisions With Multiple Variable Length Arguments	Passed
V-35	Message call with the hardcoded gas amount	Passed
V-36	Code With No Effects (Irrelevant/Dead Code)	Informational
V-37	Unencrypted Private Data On-Chain	Passed



FINDINGS

ID	Category	Issue	Severity
V-01	Vulnerabilities	Unlocked Compiler	Minor
GO-01	Gas Optimization	State Variables that could be declared constant	Informational
CS-02	Coding Standards	Dead Code	Informational



GO-01: State variables that could be declared constant

Line #189-190

```
uint256 refPercentage = 30;  
uint256 devPercentage = 100;
```

Description

refPercentage and devPercentage do not change anywhere and should be declared constant to save gas.

Recommendation

Add the constant attributes to state variables that never change.



CS-01: Dead Code

Line #351 - 379

```
} else if (currDays >= FeesKey[40].daysInSeconds && currDays < FeesKey[50].daysInSeconds){  
} else if (currDays >= FeesKey[50].daysInSeconds){
```

Description

FeesKey[50] value key is not defined in constructor. Therefore, the last **if..else** statements in **withdrawInitial()** function is never used and makes code review more difficult.

Recommendation

- Delete unused code / replace with valid statement.



Website	https://wealthmountain.app/
Domain Registry	https://www.namecheap.com/
Domain Expiry Date	2023-07-02
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Passed
Sitemap Test	Informative
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	0.45 seconds - Passed
HTTP2 Test	Passed
Safe Browsing Test	Passed



DISCLAIMER

SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice, or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

Accuracy of Information

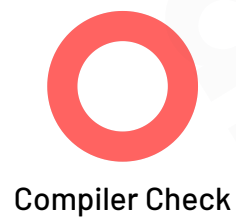
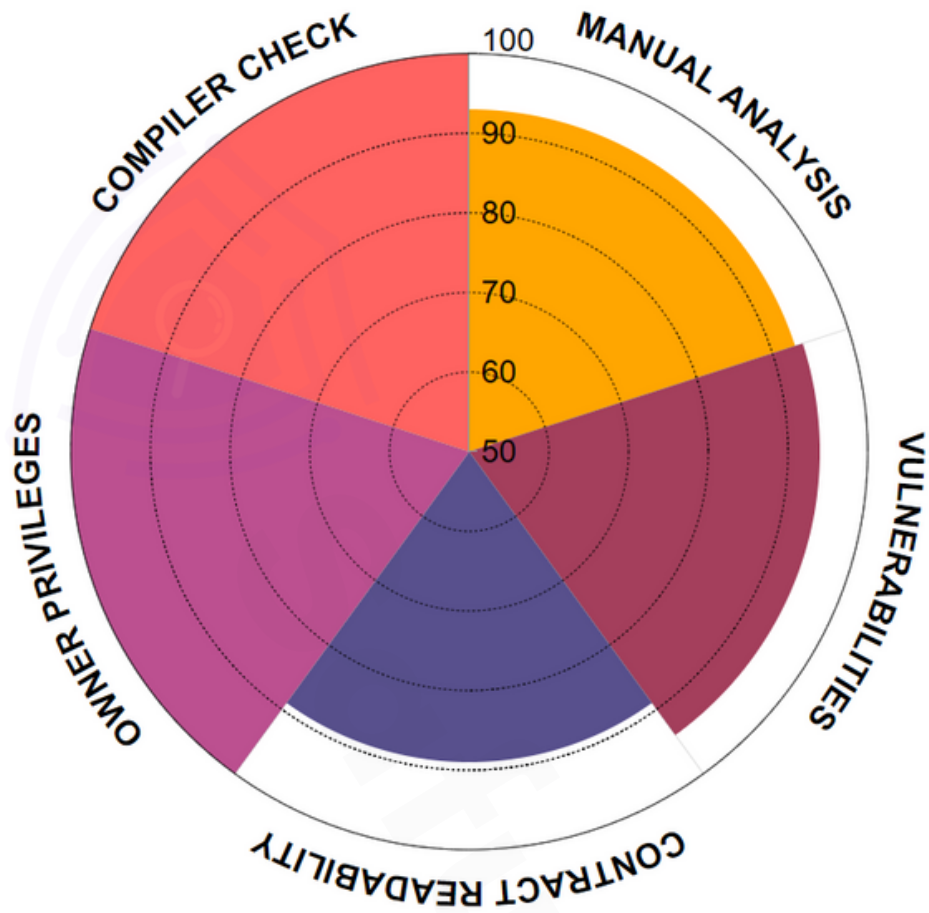
SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only – we recommend proceeding with several independent audits. Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on the smart contract safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.



RATING



Final Score: **95.2**



CONCLUSION

Project Wealth Mountain does not contain any severe issues or risk characteristics.

There is no token, no mining mechanisms, and no tokenomics. Users can directly stake BUSD stablecoin and earn daily dividends.

SafuAudit has tested the security based on manual and automated tests. Please note that we don't offer any warranties for business model.





SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



"Only in growth, reform, and change, paradoxically enough, is true security to be found."



www.safuaudit.com

