AUDITS

MH SWIFTSCAN REVIEW

FIGHT OF THE AGES

JUNE 23 RD 2022

AUDITS

TABLE OF CONTENTS

LEGAL DISCLAIMER

MH Audits are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts MH Audits to perform a security review.

MH Audits does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

MH Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

The report is provided only for the contract(s) mentioned in the report and does not include any other potential additions and/or contracts deployed by Owner. The report does not provide a review for contract(s), applications and/or operations, that are out of this report scope.

MH Audits’ goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

MH Audits represents an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. MH Audits’ position is that each company and individual are responsible for their own due diligence and continuous security.

The security audit is not meant to replace functional testing done before a software release. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits and a public bug bounty program to ensure the security of the smart contracts.

MH AUDITS
INTRODUCTION

MH Audits is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

Secure your project with MH Audits
We offer field-proven audits with in-depth reporting and a range of suggestions to improve and avoid contract vulnerabilities.

Industry-leading comprehensive and transparent smart contract auditing on all public and private blockchains.

Vunerability checking
A crucial manual inspection carried out to eliminate any code flaws and security loopholes. This is vital to avoid vulnerabilities and exposures incurring costly errors at a later stage.

Contract verification
A thorough and comprehensive review in order to verify the safety of a smart contract and ensure it is ready for launch and built to protect the end-user.

Risk assessment
Analyse the architecture of the blockchain system to evaluate, assess and eliminate probable security breaches. This includes a full assessment of risk and a list of expert suggestions.

In-depth reporting
A truly custom exhaustive report that is transparent and depicts details of any identified threats and vulnerabilities and classifies those by severity.

Fast turnaround
We know that your time is valuable and therefore provide you with the fastest turnaround times in the industry to ensure that both your project and community are at ease.

Best-of-class blockchain engineers
Our engineers combine both experience and knowledge stemming from a large pool of developers at our disposal. We work with some of the brightest minds that have audited countless smart contracts over the last 4 years.

PROJECT SUMMARY

PROJECT INTRODUCTION

FOTA is Triple-A MOBA Gaming Project that prioritizes player experience in the Metaverse. FOTA incorporates Microsoft Mesh - a Microsoft Corporation platform pioneering in creating the most immersive Metaverse experience.

The World of FOTA is more than a fantasy-themed adventure. Races from various universes battle to become the Emperor of all realms. In FOTA, investors/players own Heroes as Non-fungible Tokens (NFTs). They can interact with the NFTs in the real world and have absolute ownership over their digital assets.

Project Name *Fight Of The Ages*

Contract Name *FOTA Token*

Contract Address *0x0A4E1BdFA75292A98C15870AeF24bd94BFFe0Bd4*

Contract Chain *Mainnet*

Contract Type *Smart Contract*

Platform *EVM*

Language *Solidity*

Codebase *https://bscscan.com/address/0x0A4E1BdFA75292A98C15870AeF24bd94BFFe0Bd4#code*

INFO & SOCIALS

Network *BNB Smart Chain (BEP20)*

Max Token Supply *700.000.000*

Website *https://www.fota.io/*

Twitter *https://twitter.com/fightoftheages*

Telegram Chat *https://t.me/fota_groupchat*

https://t.me/fota_groupchat2

Telegram Ann *https://t.me/fota_channel*

Discord *https://discord.gg/fota*

Facebook *https://www.facebook.com/fightoftheages*

Youtube *https://www.youtube.com/c/FOTAFightOfTheAges*

BSCScan *https://bscscan.com/token/0x0A4E1BdFA75292A98C15870AeF24bd94BFFe0Bd4*

AUDIT SCORES

87*

PASS

Issues	7
◆ Critical	0
◆ Major	0
◆ Medium	2
◆ Minor	2
◆ Informational	3
◆ Discussion	0

All issues are described in further detail on the following pages.

AUDIT SCOPE

FILE	LOCATION
FOTAToken.sol	*BSC Deployment: /address/0x0A4E1BdFA75292A98C15870AeF24bd94BFFe0Bd4#code*
TokenAuth.sol	*BSC Deployment: /address/0x0A4E1BdFA75292A98C15870AeF24bd94BFFe0Bd4#code*

REVIEW
METHODOLOGY

TECHNIQUES

This report has been prepared for Fight Of The Ages to discover issues and vulnerabilities in the source code of the Fight Of The Ages project as well as any contract dependencies that were not part of an officially recognized library. An examination has been performed, utilizing Static Analysis and MH SwiftScan review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts producedby industry leaders.

The security assessment resulted in findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective in the comments below.

TIMESTAMP

Version	v1.0
Date	2022/06/23
Description	Layout project Automated / Static security testing Summary

KEY FINDINGS

TITLE	SEVERITY	STATUS
Impercise Arithmetic Operations Order	◆ Medium	*Pending*
BEP20 Approve Front-Running Attack	◆ Medium	*Pending*
Cheaper Inequalities In Require()	◆ Minor	*Pending*
Cheaper Inequalities In If()	◆ Minor	*Pending*
Block Values As A Proxy For Time	◆ Informational	*Pending*
Private Modifier Does Not Hide Data	◆ Informational	*Pending*
Presence Of Overpowered Role	◆ Informational	*Pending*

VULNERABILITIES

Description:

There are no floating points in Solidity, and the compiler only interacts with integers. This creates a constraint when the code is doing division operations before multiplication and introduces a possibility for rounding errors.

Consider an example - console.log((30 / 13) * 100 * 13)

This will yield 2999.9999999999995 *when done in JavaScript, but in Solidity, the same expression will produce* 2999*. This can lead to loss of funds.*

***Location:** contracts/FOTAToken.sol L143*

***Issue:** Impercise Arithmetic Operations Order*

***Level:** Medium*

***Recommendation:** Keeping the right precision in your smart contracts is very important, especially when dealing with ratios and rates which reflect economic decisions.*

It should be ensured that any ratios or rates being used allow for large numerators inside fractions.

Another tactic to keep in mind is to be careful of the order of operations.

Notice that the division occurs before the multiplication. This example would have achieved a greater precision if the calculation performed the multiplication first and then the division.

Alleviation:

IN-DEPTH VULNERABILITIES

Description:

The approve() *method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account.*

This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.

Meanwhile, if the sender decides to change the amount and sends another approve *transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the* BEP20 Approve *function.*

Location: *contracts/FOTAToken.sol L200-L203*

Issue: *BEP20 Approve Front-Running Attack*

Level: *Medium*

Recommendation: *Only use the approve function of the BEP-20 standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).*

*Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Bscscan.io] (*https://bscscan.io/*)*

Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

Alleviation:

IN-DEPTH VULNERABILITIES

Description:

The contract was found to be doing comparisons using inequalities inside the require statement. When inside the require statements, non-strict inequalities (>=, <=) are usually costlier than the strict equalities (>, <).

Location: *contracts/TokenAuth.sol L107; L120*
contracts/FOTAToken.sol L77; L84; L91; L98; L276

Issue: *Cheaper Inequalities In* Require()

Level: *Minor*

Recommendation: *It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save ~3 gas as long as the logic of the code is not affected.*

Alleviation:

Description:

The contract was found to be doing comparisons using inequalities inside the if statement. When inside the if statements, non-strict inequalities (>=, <=) are usually cheaper than the strict equalities (>, <).

Location: *contracts/FOTAToken.sol L120; L136; L156; L176*

Issue: *Cheaper Inequalities In* If()

Level: *Minor*

Recommendation: *It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save ~3 gas as long as the logic of the code is not affected.*

Alleviation:

Description:

block.timestamp *and* block.number *can be used to determine the current time or the time delta. However, they are not recommended for most use cases.*

For block.number*, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.*

Due to variable block times, block.number *should not be relied on for precise calculations of time.*

Location: *contracts/FOTAToken.sol L55; L61; L68; L73; L116; L120-L122; L132; L136-138; L149; L156-L158; L169; L176-L178*

Issue: *Block Values As A Proxy For Time*

Level: *Informational*

Recommendation: *Smart contracts should be written with the idea that block values are not precise, and their use can have unexpected results. Alternatively, oracles can be used.*

Alleviation:

IN-DEPTH
VULNERABILITIES

Description:

Everything that is inside a contract is visible to all observers external to the blockchain. Making something private only prevents other contracts from reading or modifying the information, but it will still be visible to the whole world and observers of the blockchain.

Miners have access to all contracts' code and data. Developers must account for the lack of privacy in Ethereum (EVM / BNB Smart Chain).

Location: contracts/FOTAToken.sol L37-L40; L47;

Issue: Private Modifier Does Not Hide Data

Level: Informational

Recommendation: Keep in mind that the private modifier does not make a variable invisible and should not keep sensitive contents within the modifier.

It is a best practice to use private when you really want to protect your state variables and functions because you hide them behind logic executed through internal or public functions.

Alleviation:

VULNERABILITIES

Description:

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.

Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Location: *contracts/TokenAuth.sol L85-L110; L118-L123*

Issue: *Presence Of Overpowered Role*

Level: *Informational*

Recommendation: *We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address.*

For systems that are provisioned for a single user, you can use ***[Ownable.sol](***https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/ownership/Ownable.sol***).***

For systems that require provisioning users in a group, you can use ***[@openzeppelin/Roles.sol](***https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/access/Roles.sol***)*** *or* ***[@hq20/Whitelist.sol](***https://github.com/HQ20/contracts/blob/v0.0.2/contracts/access/Whitelist.sol***).***

Alleviation:

SOURCE CODE

https://bscscan.com/address/0x0A4E1BdFA75292A98C15870AeF24bd94BFFe0Bd4#code

APPENDIX

FINDING CATEGORIES

The assessment process will utilize a mixture of static analysis, swift scan and other security techniques.

This report has been prepared for Fight Of The Ages project using MH SwiftScan to examine and discover vulnerabilities and safe coding practices in Supernova's smart contract including the libraries used by the contract that are not officially recognized.

The scan runs a comprehensive static analysis on the solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (110+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to gas optimizations that help in reducing the overall gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

AUDIT SCORES

MH Audits AuditScores is not a live dynamic score. It is a fixed value determined at the time of the report issuance date.

*****Note that if no manual in-depth expert review has been performed a score multiplier of .9 will apply to the final result.

MH Audits AuditScores are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports and scores are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts MH Audits to perform a security review.

AUDITS

WEBSITE
MHAUDITS.IO

TWITTER
@MHAUDITS