



CHAIN AUDIT
PROFESSIONAL AUDIT

SMART CONTRACT AUDIT

SPACE BASE



SEPTEMBER 2ND, 2023.

Prepared by
Francis K.

Approved by
Chain Audit Team

INTRODUCTION

AUDITING FIRM	CHAIN AUDIT
CLIENT FIRM	SPACE BASE
METHODOLOGY	MANUAL CODE REVIEW AND AUTOMATED ANALYSIS
LANGUAGE	SOLIDITY
CONTRACT	0xA2493539e1035f75bf040288802d3BF2FcFD2Bf6
BLOCKCHAIN	ETHEREUM
CENTRALIZATION	YES
WEBSITE	https://spacebase.site/

You can check the authenticity of this audit on our website.

EXECUTIVE SUMMARY






Objective and Scope

The purpose of this audit was to evaluate and ensure the integrity, security, and functionality of the "SpaceBase" smart contract developed in Solidity for the Ethereum platform. A thorough review of the code was conducted, encompassing both manual and automated testing, with the aim of identifying and mitigating potential vulnerabilities and risks associated with the contract.

Methodology

A combined review approach was adopted:

- **Manual Review:** Our team of Solidity experts carried out a detailed and systematic review of the source code, paying particular attention to common vulnerabilities and insecure code patterns. Each function and modularity of the contract, its internal logic, and interactions amongst them were assessed.
- **Automated Review:** We utilized top-tier automated audit tools to scan the code for known vulnerabilities, compilation errors, and other potential security issues.

STATUS	CRITICAL 	MAJOR 	MEDIUM 	MINOR 	UNKNOWN 
OPEN	2	5	1	0	0
ACKNOWLEDGED	0	0	0	0	0
RESOLVED	0	0	0	0	0

IMPORTANCE OF SMART CONTRACT AUDITS

Smart contracts have ushered in a new era of trustless and decentralized operations on the blockchain. While they have the potential to revolutionize numerous sectors, from finance to supply chain, their immutable nature means that any vulnerability or flaw in their code is permanent once deployed. This underscores the crucial importance of smart contract audits.

Why are Audits Necessary?

1. **Immutability:** Once a contract is deployed, it cannot be changed. A bug or vulnerability can be exploited repeatedly unless it's fixed in a new version of the contract.
2. **Financial Implications:** Smart contracts often handle and manage valuable assets. Vulnerabilities can lead to substantial financial losses.
3. **Reputation:** A flawed contract can tarnish the reputation of a project, leading to a loss of trust and confidence among its users.
4. **Complexity:** Solidity, the primary language for Ethereum contracts, has its quirks. Even experienced developers might overlook subtleties that can become vulnerabilities.

Types of Smart Contract Attacks:

Smart contracts are vulnerable to a variety of attacks. Some of the most common include reentrancy attacks, overflow and underflow attacks, timestamp dependence attacks, and more. Each of these attacks exploits specific vulnerabilities in contract code, and a comprehensive audit aims to safeguard against all known vulnerabilities.

RISK CATEGORIES

Risk Type	Definition
Critical ●	A vulnerability that, if exploited, could have a catastrophic impact, potentially leading to substantial financial loss or irreversible damage to the contract's operations.
Major ●	A significant vulnerability that might not lead to total loss but can hamper the contract's functionality and compromise its objectives.
Medium ●	Issues that are of concern but might require specific conditions to be exploited. They can pose risks if combined with other vulnerabilities.
Minor ●	These vulnerabilities pose a limited threat and have a lower probability of being exploited. Often, they relate to best practices rather than direct exploitable flaws.
Unknown ●	Risks that haven't been fully understood or classified yet. They could be new or unique to the contract's specific design or context.

Status of Identified Risks

Status Type	Definition
Open	Vulnerabilities that have been identified but have not yet been addressed or rectified by the development team.
Acknowledged	The development team has recognized the issue but might be in the process of determining the best solution or mitigation strategy.
Resolved	The vulnerability has been effectively addressed and resolved by the development team, eliminating the risk it posed.

AUDITING IS AN ESSENTIAL STEP IN THE DEVELOPMENT AND DEPLOYMENT OF SMART CONTRACTS. IT ENSURES NOT ONLY THE SECURITY AND RELIABILITY OF THE CONTRACT BUT ALSO BUILDS TRUST AMONG ITS USERS AND STAKEHOLDERS. AS SMART CONTRACTS CONTINUE TO GROW IN COMPLEXITY AND IMPORTANCE, ROBUST AUDITING MECHANISMS WILL REMAIN A CORNERSTONE OF THE BLOCKCHAIN ECOSYSTEM.

IMPORTANT CONTRACT DETAILS

- **CHAIN:** ETHEREUM
- **TOKEN:** ETH
- **DAPP TYPE:** TOWER AND MAGE-BASED INVESTMENT GAME.
- **CONVERSION RATE:**
 - 1 MANA = 0,0001 ETH
 - 1 EMERALDS = 0,012 MANA
- **REFERRAL PERCENTAGE :**
 - STANDARD REFERRAL: 5% IN MANA
 - AFFILIATE REFERRAL: 7% IN EMERALDS FOR AFFILIATE
- **FEE :** 4%
- **MANAGER:** UNIQUE ADDRESS DEFINED IN THE CONSTRUCTOR, HAS LIMITED ADMINISTRATIVE POWERS.
- **REWARD ACCUMULATION:** BASED ON "YIELD", WHICH IS CALCULATED FROM THE TYPE AND LEVEL OF MAGES.

ANALYSIS OF CONTRACT FUNCTIONS

Function Name	Description	Usage
updateAffiliateStatus(address user, bool status)	Allows the manager to update the affiliate status of a user.	Used by the contract owner to mark or unmark a user as an affiliate.
addMana(address ref)	Allows users to add 'mana' to the contract by sending ETH. Also handles referral bonuses.	Users send ETH to this function to increase their 'mana' and, if they have a referrer, to give referral bonuses.
compound(uint256 emeralds)	Allows users to convert their 'emeralds' into 'mana' at a given rate.	Used by users to increase their 'mana' using their 'emeralds'.
withdrawMoney(uint256 emeralds)	Allows users to withdraw their 'emeralds' as ETH.	Used by users to withdraw their earnings.
collectMoney()	Syncs and collects accumulated 'emeralds' for the calling user.	Used by users to update and collect their 'emeralds' without withdrawing them.
upgradeTower(uint256 towerId)	Allows users to upgrade their tower, which increases the yield of 'emeralds'.	Users invoke this function to spend 'mana' and upgrade their tower.
upgradeTreasury()	Allows users to upgrade their treasury, which might increase capacity or yield.	Users use this function to spend 'mana' and upgrade their treasury.
getMages(address addr)	Queries and returns the 'mages' status for a given address.	To retrieve information about the 'mages' of a specific user.
syncTower(address user)	Syncs the user's tower, updating values based on yield and time.	This is an internal function used by other functions to ensure a user's tower data is up-to-date before any other operation.
getUpgradePrice(uint256 towerId, uint256 magId)	Returns the price to upgrade a tower to a specific 'magId'.	Used internally to calculate upgrade costs.
getYield(uint256 towerId, uint256 magId)	Returns the 'emeralds' yield for a specific 'magId'.	Used internally to calculate the yield of a tower after an upgrade.
getTreasure(uint256 treasureId)	Returns the price and value for a specific 'treasureId'.	Used internally to calculate costs and values related to treasures.

OWNER PRIVILEGES

THE SPACEBASE CONTRACT DISPLAYS THE FOLLOWING FUNCTION THAT GRANTS EXCLUSIVE RIGHTS TO THE CONTRACT MANAGER:

- **UPDATEAFFILIATESTATUS(ADDRESS USER, BOOL STATUS):**
 - **DESCRIPTION:** ALLOWS THE MANAGER TO UPDATE THE AFFILIATE STATUS OF A USER.
 - **RESTRICTION:** ONLY THE CONTRACT MANAGER CAN CALL THIS FUNCTION. THIS ENSURES THAT ONLY THE MANAGER HAS THE AUTHORITY TO MARK OR UNMARK A USER AS AN AFFILIATE.

IT'S PARAMOUNT TO UNDERSTAND THAT THIS FUNCTIONALITY CAN SIGNIFICANTLY INFLUENCE THE DYNAMICS OF THE CONTRACT, ESPECIALLY CONCERNING THE AFFILIATE SYSTEM. THEREFORE, IT SHOULD BE INVOKED JUDICIOUSLY.

CRITICAL VULNERABILITY FINDINGS

Vulnerability	Risk Type	Vulnerable	Reason	Affected Code
Overflow/Underflow	Critical ●	Yes	Lack of checks for overflow/underflow in mathematical operations.	Functions: addMana, compound, withdrawMoney, syncTower, upgradeTower, upgradeTreasury.
Short Address/Parameter Attack	Major ●	Yes	Transfers based on msg.value without additional verifications.	Function: addMana.
Timestamp Dependence	Major ●	Yes	Dependency on block.timestamp, which can be manipulated by miners.	Function: syncTower.
Outdated Compiler Version	Major ●	Yes	Use of an outdated Solidity version.	Directive pragma solidity 0.8.18.
Front Running	Major ●	Yes	Transactions can be anticipated or delayed by malicious actors or miners.	Functions: addMana and withdrawMoney.
Reordering Attack	Major ●	Yes	Transactions depend on the order of inclusion in a block, which can be manipulated by miners.	Functions relying on block.timestamp like syncTower.
Hardcoded Addresses	Medium ●	Yes	Hardcoded addresses can be problematic if control needs to be shifted.	Constructor and use of the manager variable.
Race Conditions	Critical ●	Yes	Multiple transactions can interact concurrently, leading to undesired conditions.	Functions: addMana and withdrawMoney.

FINDINGS AND RECOMMENDATIONS

BASED ON THE AUDIT OF THE SPACEBASE CONTRACT, THE FOLLOWING OBSERVATIONS AND SUGGESTIONS ARISE:

- **OVERFLOW/UNDERFLOW:**
 - OBSERVATION: THE CONTRACT DOES NOT HAVE CHECKS FOR OVERFLOW OR UNDERFLOW DURING ARITHMETIC OPERATIONS IN SEVERAL FUNCTIONS.
 - RECOMMENDATION: IMPLEMENT SAFEMATH OR USE SOLIDITY 0.8'S BUILT-IN OVERFLOW/UNDERFLOW CHECKS THROUGHOUT THE CONTRACT TO PREVENT THESE VULNERABILITIES.
- **SHORT ADDRESS/PARAMETER ATTACK:**
 - OBSERVATION: THE ADDMANA FUNCTION ACCEPTS TRANSFERS BASED ON MSG.VALUE WITHOUT ADDITIONAL VERIFICATIONS.
 - RECOMMENDATION: ENSURE THAT THE SENT DATA MATCHES EXPECTED LENGTHS. VALIDATE THE INPUT PARAMETERS THOROUGHLY.
- **TIMESTAMP DEPENDENCE:**
 - OBSERVATION: THE CONTRACT RELIES ON BLOCK.TIMESTAMP FOR ITS SYNCTOWER FUNCTION, WHICH CAN BE INFLUENCED TO A CERTAIN DEGREE BY MINERS.
 - RECOMMENDATION: CONSIDER MECHANISMS LESS PRONE TO MANIPULATION OR USE A DECENTRALIZED ORACLE FOR TIME CHECKS.
- **OUTDATED COMPILER VERSION:**
 - OBSERVATION: THE CONTRACT USES SOLIDITY VERSION 0.8.18, WHICH MIGHT NOT HAVE THE LATEST SECURITY PATCHES OR OPTIMIZATIONS.
 - RECOMMENDATION: UPGRADE TO THE LATEST STABLE VERSION OF SOLIDITY, ENSURING COMPATIBILITY AND THOROUGH TESTING.
- **FRONT RUNNING:**
 - OBSERVATION: TRANSACTIONS IN ADDMANA AND WITHDRAWMONEY FUNCTIONS CAN BE FRONT-RUN, LEADING TO POTENTIAL LOSS OR OTHER UNEXPECTED BEHAVIORS.
 - RECOMMENDATION: IMPLEMENT MECHANISMS TO DETER FRONT-RUNNING, SUCH AS USING COMMIT-REVEAL SCHEMES OR CONSIDERING OTHER ARCHITECTURAL CHANGES.
- **REORDERING ATTACK:**
 - OBSERVATION: TRANSACTIONS IN THE CONTRACT CAN BE REORDERED, ESPECIALLY THOSE RELYING ON BLOCK.TIMESTAMP, LEADING TO UNDESIRE OUTCOMES.
 - RECOMMENDATION: BE AWARE OF THIS RISK, AND DESIGN CONTRACT FUNCTIONS WITH THE ASSUMPTION THAT TRANSACTION ORDER CANNOT BE GUARANTEED.
- **HARDCODED ADDRESSES:**
 - OBSERVATION: THE MANAGER ADDRESS IS HARDCODED INTO THE CONTRACT AND CAN'T BE CHANGED.
 - RECOMMENDATION: PROVIDE MECHANISMS TO CHANGE HARDCODED ADDRESSES TO ENSURE FLEXIBILITY AND CONTROL IN UNFORESEEN SCENARIOS.
- **RACE CONDITIONS:**
 - OBSERVATION: FUNCTIONS ADDMANA AND WITHDRAWMONEY MIGHT BE SUSCEPTIBLE TO RACE CONDITIONS WHERE MULTIPLE TRANSACTIONS CAN LEAD TO UNEXPECTED RESULTS.
 - RECOMMENDATION: IMPLEMENT LOCKING MECHANISMS OR CHECKS TO PREVENT CONCURRENT INTERACTIONS THAT COULD LEAD TO RACE CONDITIONS.

CONCLUSION

UPON REVIEWING THE SPACEBASE SMART CONTRACT, WE HAVE IDENTIFIED SEVERAL VULNERABILITIES AND RISK POINTS THAT COULD ADVERSELY IMPACT ITS USERS AND THE CONTRACT'S INTEGRITY. THESE VULNERABILITIES, IF EXPLOITED, COULD LEAD TO FINANCIAL LOSSES, DATA MANIPULATION, AND OTHER UNWANTED BEHAVIORS. IT'S PARAMOUNT TO ADDRESS THESE ISSUES BEFORE THE CONTRACT IS USED IN A PRODUCTION ENVIRONMENT OR EXPOSED TO A SIGNIFICANT NUMBER OF USERS. THE IDENTIFIED VULNERABILITIES, SUCH AS RACE CONDITIONS AND TRANSACTION FRONT-RUNNING RISKS, ARE PARTICULARLY CONCERNING DUE TO THEIR POTENTIAL IMPACT AND THE COMPLEXITY OF THEIR MITIGATION.

SECURITY RATING 4/10

THE CONTRACT SCORES A 4 OUT OF 10 IN TERMS OF SECURITY. THIS RATING REFLECTS THE PRESENCE OF MULTIPLE MEDIUM TO HIGH-RISK VULNERABILITIES AND THE NEED FOR SUBSTANTIAL CODE REVISION AND MODIFICATION BEFORE DEPLOYING ON A MAINNET. IT'S HIGHLY ADVISABLE TO CONDUCT FURTHER REVIEWS AND COMPREHENSIVE TESTING AFTER ADDRESSING THE PROVIDED RECOMMENDATIONS.



CHAINAUDIT
FRANCIS K.
AUDITOR FOR CHAINAUDIT