SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 14. January, 2022

Audit

Security Assessment
16. January, 2022

For

Disclaimer 3
Description 5
Project Engagement 5
Logo 5
Contract Link 5
Methodology 7
Used Code from other Frameworks/Smart Contracts (direct imports) 8
Tested Contract Files 9
Source Lines 10
Risk Level 10
Capabilities 11
Scope of Work 13
Inheritance Graph 13
Verify Claims 14
Modifiers 20
CallGraph 22
Source Units in Scope 23
Critical issues 24
High issues 24
Medium issues 24
Low issues 24
Informational issues 24
Audit Comments 26
SWC Attacks 27

Disclaimer

SolidProof.io reports 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. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof 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.

SolidProof.io Reports represent 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. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	14. January 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	16. January 2022	• Reaudit

Network
Binance Smart Chain (BEP20)

Website
https://spacebattleship.com/

Telegram
https://t.me/SpaceBattleShip

Description

New platform for all in one. Staking, farming, swap, DApp, and much more...
With our system (Tokenomics) we create a stable liquidity pool that enables investors to invest securely without suffering losses

Project Engagement

During the 12th of January 2022, **Spacebattleship Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- https://bscscan.com/address/0x873651ca77ab5f740bb61f36a5c499b6aace928c#code

v1.1

- https://bscscan.com/address/0xa5411d1e0924d64c2e32590f670b08f54b5f147b#code

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

V1.0

```
Context
Ownable
IERC20
IERC20Metadata
IERC721
IERC721Enumerable
IDexFactory
IDexRouter
IUniswapV2Pair
DividendPayingTokenOptionalInterface
DividendPayingTokenInterface
SafeMath
SignedSafeMath
SafeCast
IterableMapping
ERC20
SafeToken
DividendPayingToken
SpaceBattleShipDividendTracker
```

V1.1

```
Context
Ownable
ReentrancyGuard
IERC20
IERC20Metadata
IERC721
IERC721Enumerable
IDexFactory
IDexRouter
IUniswapV2Pair
DividendPayingTokenOptionalInterface
DividendPayingTokenInterface
SignedSafeMath
SafeCast
IterableMapping
ERC20
SafeToken
DividendPayingToken
SpaceBattleShipDividendTracker
```

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

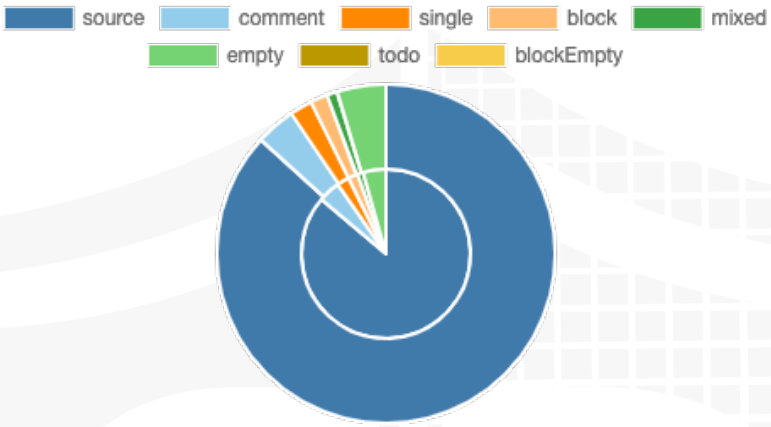
File Name	SHA-1 Hash
contracts/spacebattleship.sol	a9376a7062399eac2d478563bd183dd8ad639e19

v1.1

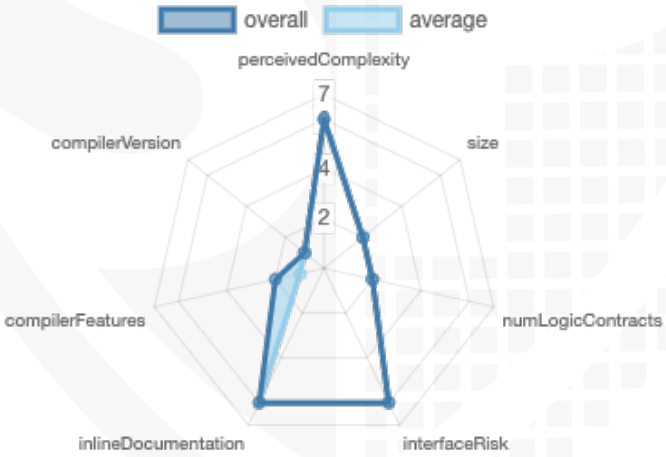
File Name	SHA-1 Hash
contracts/spacebattleship.sol	549cf6174d921df82483b3ec49b8f54b14354a5b

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	7	4	9	2
1.1	7	3	9	3

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	170	9
1.1	168	9

Version	External	Internal	Private	Pure	View
1.0	119	168	3	43	79
1.1	117	152	3	30	79

State Variables

Version	Total	Public
1.0	71	26
1.1	71	34

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.10`		`yes`		
1.1	`0.8.10`		`yes`		

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0	yes					yes → NewContract:SpaceBattleShipDividendTracker

Scope of Work

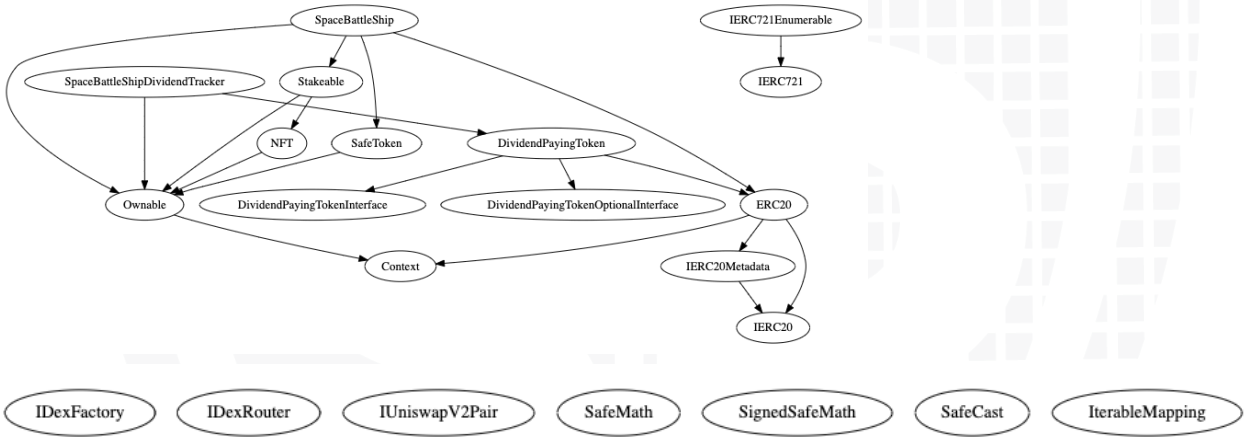
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

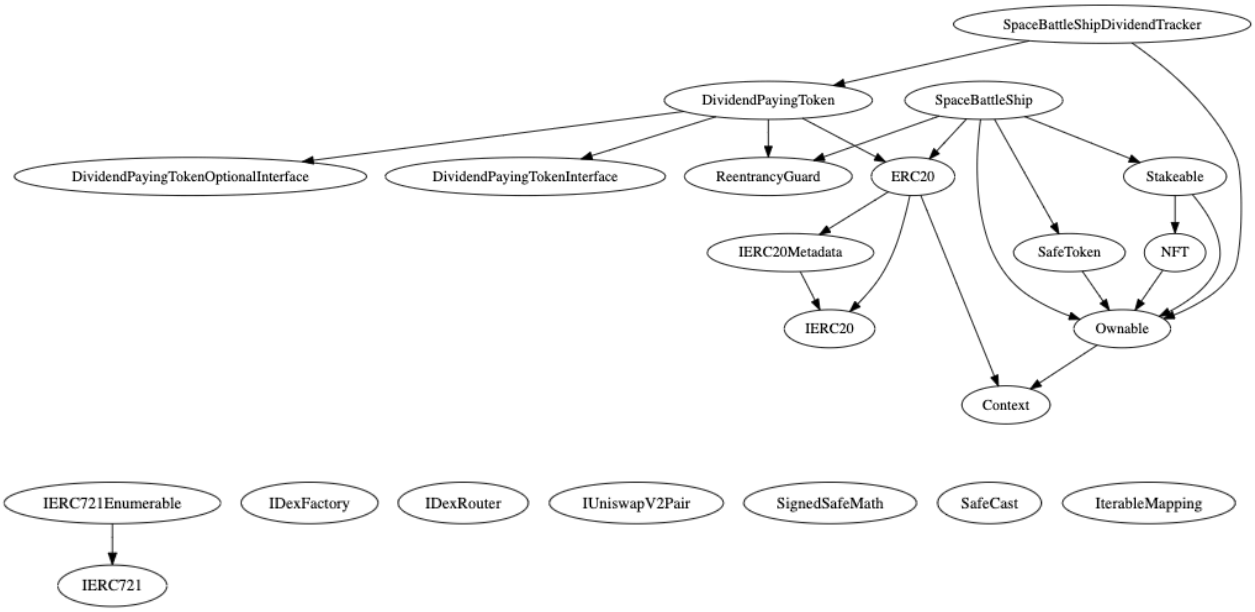
1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph

v1.0



v1.1



Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract

1. addNFT
2. addPair
3. approve
4. claim
5. decreaseAllowance
6. excludeFromDividends
7. increaseAllowance
8. processDividendTracker
9. removeLastPair
10. rescueAllBNB
11. rescueAllTokens
12. rescueBNB
13. rescueToken
14. setApy
15. setBeneficiarySettings
16. setExcludeFromAll
17. setExtraFeeOnSell
18. setFees
19. setIsFeeExempt
20. setIsTxLimitExempt
21. setMarketingWallet
22. setMaxBuyAndWallet
23. setMaxSellTx
24. setNFTContractAddress
25. setNFTContractAdmin
26. setSafeManager
27. setStakingWallet
28. setSwapSettings
29. stake
30. transfer
31. transferFrom
32. transferOwnership
33. updateClaimWait
34. updateGasForProcessing
35. websiteSwapBnbForTokens
36. websiteSwapTokensForBnb
37. withdrawStake

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified
Deployer cannot mint	✓	✓	✓

Max / Total Supply: 100.000.000

Comments:

v1.0

- Tokens will be burned/minted in dividend tracker contract while using setBalance function in tracker

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Tokens will be burned/minted in dividend tracker contract while using setBalance function in tracker
- Deployer can lock by
  - Setting _maxTxAmountBuy to 0
  - Setting too high fees
- Deployer can set _maxWalletAmount to minimum _calculatedTotalSupply/100

v1.1

- Team fixed
  - Setting _maxTxAmountBuy to 0
    - Minimum _maxTxAmountBuy must be higher equal to _calculatedTotalSupply / 100
- Team has added new events for critical variable changes
- Deployer can still lock user funds by setting fees to high
- Require statement added to takeFee function to take fee from time 1642449600

Format	Seconds
GMT	Mon Jan 17 2022 20:00:00 GMT+0000
Your Time Zone	Mon Jan 17 2022 21:00:00 GMT+0100 (Central European Standard Time)
Relative	in a day

-

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	-	-	-

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	–

Modifiers

SpaceBattleShip

```
setFees
onlyOwner
setExtraFeeOnSell
onlyOwner
setSwapSettings
onlyOwner
setMaxSellTx
onlyOwner
setMaxBuyAndWallet
onlyOwner
setMarketingWallet
onlyOwner
setStakingWallet
onlyOwner
setBeneficiarySettings
onlyOwner
addPair
onlyOwner
removeLastPair
onlyOwner
setExcludeFromAll
onlyOwner
setIsFeeExempt
onlyOwner
setIsTxLimitExempt
onlyOwner
excludeFromDividends
onlyOwner
updateClaimWait
onlyOwner
updateGasForProcessing
onlyOwner
```

Stakeable

```
setApy
onlyOwner
```

NFT

```
setNFTContractAdmin
onlyOwner
setNFTContractAddress
onlyOwner
```

SpaceBattleShipDividendTracker

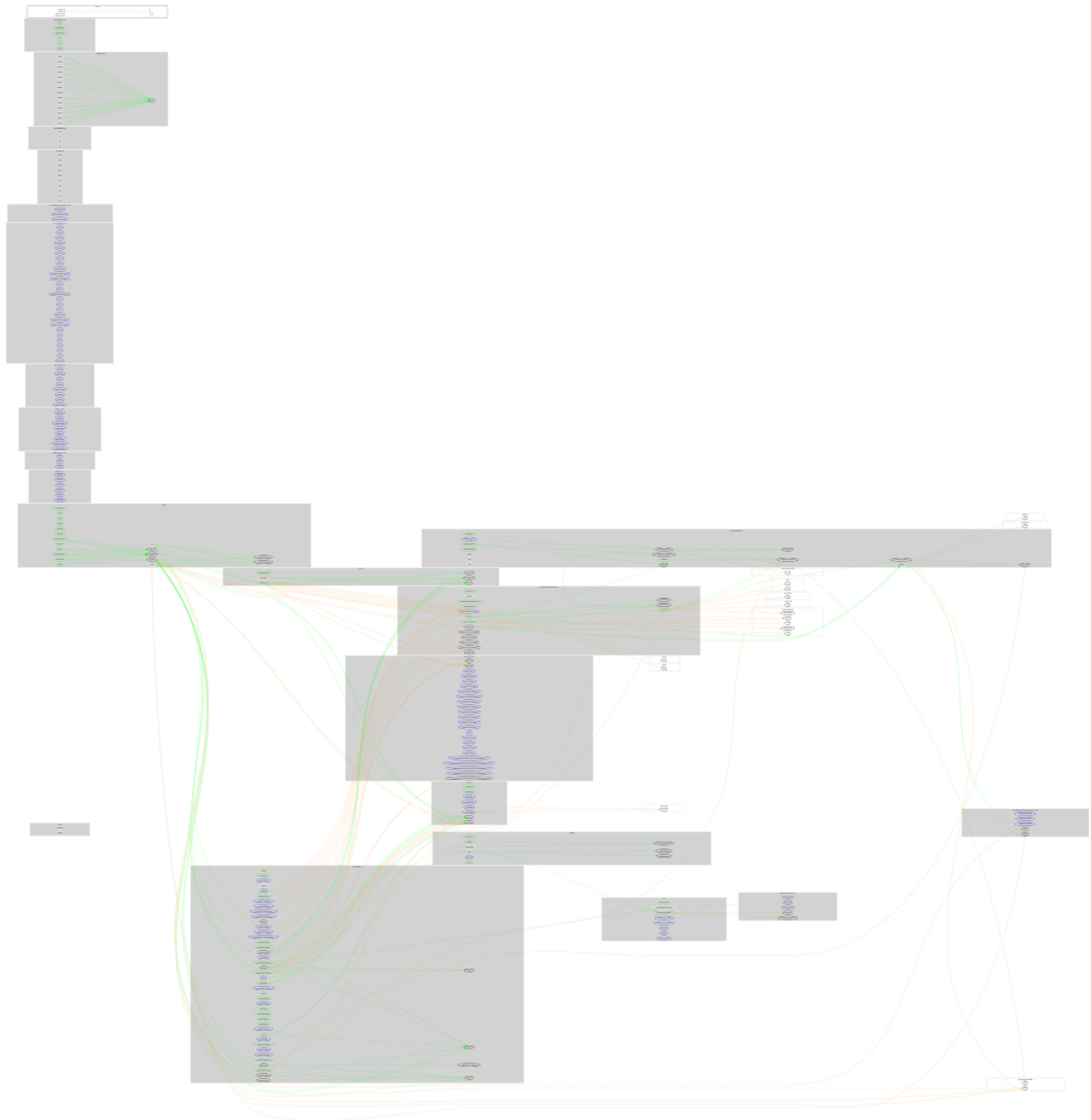
```
_minimumTokenBalanceForReward
onlyOwner
excludeFromDividends
onlyOwner
updateClaimWait
onlyOwner
setBalance
onlyOwner
process
processAccount
onlyOwner
```

Comments

- Deployer can set following state variables without any limitations
  - Fees can be set to high value
    - Deployer has to set a higher feeDenominator

- Deployer can enable/disable following state variables
  - swapEnabled
  - isTxLimitExempt
  - isFeeExempt
  - excludedFromDividends

CallGraph



Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/spacebattleship.sol	13	9	1145	1025	943	46	1066	💰💠🌀🔆♻️Σ
📝📚🔍🎨	**Totals**	**13**	**9**	**1145**	**1025**	**943**	**46**	**1066**	💰💠🌀🔆♻️Σ

v1.1

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/spacebattleship.sol	13	9	1116	996	915	46	1044	💰💠🌀🔆♻️Σ
📝📚🔍🎨	**Totals**	**13**	**9**	**1116**	**996**	**915**	**46**	**1044**	💰💠🌀🔆♻️Σ

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Main	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	Main	Missing Events Arithmetic	1055	Emit an event for critical parameter (all used state variables inside function) changes

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)	552, 553, 715	Add the `constant` attributes to state variables that never change

#2	Main	Unused return values	922, 988	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	Main	Source code formatting	-	Format the source code properly for better readability
#4	Main	Error message is missing	379, 383	Provide require statement error message We recommend to start every message with the contract name followed by double point and starts with an uppercase letter e.g. “SafeToken: Error message”

#5	Main	Unnecessary code	Between line 429 and 430	Revert the removing of _transfer function
				
				Don’t remove whole function, remove only the red part from function below require(false); of the following function
				
				function _transfer(address from, address to, uint256 value) internal virtual override {
				require(false);
				int256 _magCorrection = magnifiedDividendPerShare.mul(value).toInt256();
				
				magnifiedDividendCorrections[from] = magnifiedDividendCorrections[from].add(_magCorrection);
				
				magnifiedDividendCorrections[to] = magnifiedDividendCorrections[to].sub(_magCorrection);
				}

Audit Comments

14. January 2022:

- There are several issues which must be fixed
- We recommend you to read whole report for more information

16. January 2022:

- Reaudit
  - Several issues were fixed by the team of SpaceBattleShip

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	**PASSED**
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	**PASSED**
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	**PASSED**
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	**PASSED**
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	**PASSED**
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	**PASSED**
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	**PASSED**
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	**PASSED**

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY