verichains

SECURITY AUDIT OF

SPACESIP SMART CONTRACTS

Public Report

Oct 20, 2021

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Binance Smart Chain	This dual-chain architecture will empower its users to build their decentralized apps and digital assets on one blockchain and take advantage of the fast trading to exchange on the other: EVM Compatible, Proof of Staked Authority, Cross-Chain Transfer.
BNB	A cryptocurrency whose blockchain is generated by the Binance Smart Chain platform. Matic is used for payment of transactions and computing services in the Binance Smart Chain network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Oct 20, 2021. We would like to thank the SpaceSIP Team for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the SpaceSIP Smart Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the application, along with some recommendations. SpaceSIP Team has fixed all the reported issues.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About SpaceSIP Smart Contracts

Space SIP (SIP) is a Play to Earn NFT RPG developed on the Binance Smart Chain platform. The game revolves around the acquisition of a legendary Spaceship and powerful Weapon to wield them. Players can send Spaceship to mine $SIP tokens. Players may participate in combat using their assets to earn $SIP tokens. Assets are player owned NFTs minted in the ERC-721 standard which may be traded on the properietary marketplace.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Spacesip game. It was conducted on the source code provided by the SpaceSIP team.

The audit assumes that the variable token of the private sale contract is initializing to SIPToken of the same repository and same version given to Verichains, issues relating to deploying with other tokens or other versions of that token is not included here.

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy

- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The initial review was conducted on Oct 6, 2021 and a total effort of 14 working days was dedicated to identifying and documenting security issues in the code base of the SpaceSIP Smart Contracts.

The audit source code is on [redacted], commit process started on commit 03fc980efcf5a86e2b5548221189d153a9a99fdb.

2.2. Findings

The SpaceSIP Smart Contracts was written in Solidity language, with the required version to be ^0.8.2. The source code was written using OpenZeppelin's and Chainlink's library.

SpaceSIP Team fixed the code according to Verichain's draft report in commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

This section contains a detailed analysis of all the vulnerabilities that were discovered by the audit team during the audit process.

2.2.1. SIPTokenomicPrivateSale - Unexpected revert when whilelist price is invalid LOW

In addWhitelist function below, for loop condition packageId >= 0 can never be false:

[redacted]

As packageId is of type uint8, so it can never be negative, for solidity compiler with version lower than 0.8.0, the function will consume all remaining gas and finally revert due to out-of-gas if the inner if condition is not satisfied. In current contract, the declared solidity version is ^0.8.2 which will revert with when overflow occurs. The correct behavior is to revert with "PA>0" message. So, in the worst case this bug will cause unexpected revert (including waste of gas for lower solidity compiler), which can confuse unaware callers.

2.2.1.1. Recommend

A hacky change without assumptions is to change packageId >= 0 into packageId <= 4 as when the variable go below zero, it will become 255, this hack only work in this case (starting value is 4) but not in general, this hack also requires the code block to be nested within unchecked{ ... } for solidity 0.8.0+.

A possible change is to change the direction of for: for (uint8 packageId = 0; packageId <= 4; packageId++), note that this change will not work if 255 takes place of 4 instead.

The correct way to change should be changing uint8 to int8 so that the variable can be negative, but this require the referenced usages of packageId to be type-casted which reduce readablity (very minor), for example packagePrice[packageId] to packagePrice[uint8(packageId)].

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.2. SpaceshipPublicSale - Wrong white-listed user's boughtCount check LOW

While removing a white-listed user using removeWhitelists function, corresponding record in whitelist mapping's boughtCount attribute is checked with condition > 1. As from the function name, the if's body and the implementation of the singular operation function removeWhitelist, audit team thinks that the actual intent is to not to remove any user that has been paid before. In that case, the condition should be >= 1 or > 0. Current implementation will consider users that bought once the same as ones that have never bought and will remove them all.

[redacted]

RECOMMENDATION

We should update the boughtCount check as below (the negative condition was removed for simplification).

[redacted]

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.3. MysteryCapsule - Missing spaceshipMaxByMonth limit check LOW

In the _createSpaceship function of the MysteryCapsule contract, the spaceshipMaxByMonth variable is used to limit the number of spaceships creations per month. However, spaceships creations from the buy function of SpaceshipPublicSale contract do not have this limit check.

[redacted]

RECOMMENDATION

The requirement logic when adding a new spaceship should be reviewed carefully to check whether the spaceshipMaxByMonth variable should be used here or not.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.4. Spaceship - Burnt ships are not deallocated from the array LOW

The Spaceship contract manages spaceships by storing their information in the ships dynamic array. However, when a spaceship is burnt, its corresponding storage should be deallocated for both memory/gas saving and preventing of future relating security problems.

RECOMMENDATION

The code can be updated as below. By the way, using mapping instead of a dynamic array to store spaceships will be better in this case. Since it'll not leave a gap between items or require swap-and-delete implementation which consume more gas.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.5. MysteryCapsule - incorrect monthly limits update LOW

In the _createSpaceship function, when the block.timestamp reaches the next month, currentMonth should be updated to the new value before it is used for monthly limits update.

RECOMMENDATION

The code should be updated as below.

Or better, the if should be moved into _resetLimitByMonth.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.6. Unnecessary usage of upgradeable library contracts INFORMATIVE

Note: *This is not a security problem as long as the deployment is performed correctly. This issue is only provided as informational so readers can acknowledge the limitations of the chosen solution and relating possible attacks.*

SIPTokenomic is not planned to be upgradable but is based on AccessControlUpgradeable and PausableUpgradeable (upgradable variant of AccessControl and Pause contract of OpenZeppelin).

Read more: https://docs.openzeppelin.com/contracts/3.x/upgradeable.

[redacted]

Upgradeable variant contracts perform initial setup by calling initialize after construction without access control to the initialize function, that is, anyone that call initialize first will be the owner. So, the upgradeable contracts are subject to front-running-related attacks in term of initialize calling. Normal variant contracts initialize everything in constructor, so they are safe.

Upgradeable contracts are only useful when the main contract is upgradeable one, they introduce coding limitations that must be careful while coding new features too.

Read more: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.7. Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

Note: *This is not a security problem. This issue is only provided as informational so readers can acknowledge the limitations of the chosen solution.*

All SafeMath usage in the contract are for overflow checking, solidity 0.8.0+ already do that by default, the only usage of SafeMath now is to have a custom revert message which isn't the case in the auditing contracts. We suggest to use normal operators for readability and gas saving.

[redacted]

SIPToken uses the library too:

[redacted]

The given source code also make use of normal math operations (integer operations without SafeMath usage). Mixed usage of SafeMath and non-SafeMath in a single codebase will confuse unaware programmer/reader as normal operations are not reverted on overflow for old solidity compiler, the overflow may be intended or not, but they will all be reverted in solidity 0.8.0+.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.8. SIPTokenomicPrivateSale - Unchecked token deposit INFORMATIVE

The contract makes use of ERC20.transferFrom which may or may not revert if the token cannot be transferred. If the call silently returns false on error, as the calling contract is not handling return value it will not be aware of the error and still consider the call as success while the actual token amount has not been transferred.

The contract only use ERC20.transferFrom to transfer expected token amount without checking balance change, this will result in loss of token if the token to be transferred is fee-collecting token (token that transfer fee to some other place when use make transfers).

SIPTokenomicPrivateSale contract only use transferFrom when depositing USDT, USDC, BUSD and TUSD which will not be the case if these are the well-known corresponding token.

So, if only well-known values provided there will not be any error.

2.2.8.1. Recommend

- Make use of SafeERC20 and use safeTransferFrom if there will be any custom stable token that return false instead of reverting.
- Check the balance increasement if there will be any custom stable token that is fee-collecting token.

UPDATES

- *2021-10-20*: This has been acknowledged by SpaceSIP Team.

2.2.9. ChainlinkGlobalSeed - Base random seeds can be leaked by reading contract storage INFORMATIVE

Return value of requestRandomNumber(address user) function in the ChainlinkGlobalSeed contract will be used as the base seed for various random numbers within other contracts. The value is easy to predict by reading the contract storage and run before Chainlink updates the previous seed generation request.

[redacted]

These seeds will later be combined with currently trustable random sources like [redacted] so the final random system is still good.

UPDATES

- *2021-10-20*: This has been acknowledged by SpaceSIP Team.

2.2.10. MysteryCapsule - Unnecessary checks INFORMATIVE

The transfer and transferFrom functions of the ERC20 contract already check the balance of the sender before transfer, so the balance checks are unnecessary.

[redacted]

RECOMMENDATION

Remove all the unnecessary checks above.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.11. Unused functions and modifiers INFORMATIVE

There're some functions and modifiers which are not used anywhere as below.

[redacted]

RECOMMENDATION

Remove these functions and modifiers.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.12. SpaceshipPublicSale - Redundant initialized function calls INFORMATIVE

There're some redundant function calls in the initialize function as below.

[redacted]

RECOMMENDATION

Update the code as below.

[redacted]

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.13. SpaceshipPublicSale - Typo INFORMATIVE

There is a typo error in a storage variable name in the SpaceshipPublicSale contract.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.14. SIPToken - Unnecessary check of maxTransferAmount() INFORMATIVE

The maxTransferAmount() should be equal to totalSupply() in case we want to disable the anti-whale feature. So, the check below is unnecessary.

RECOMMENDATION

Remove the unnecessary check as below.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.15. SIPToken - Redundant code in the _transfer function INFORMATIVE

There are some redundant lines of code in the _transfer function as below.

RECOMMENDATION

Remove all the redundant lines of code above.

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

2.2.16. SIPToken - Inconsistent usages of msg.sender INFORMATIVE

The msg.sender should be replaced with _msgSender() to maintain the codebase's consistency.

[redacted]

RECOMMENDATION

[redacted]

UPDATES

- *2021-10-20*: SpaceSIP Team has fixed the issue at commit fcb004b4b2c1492849efaff21d13f03df15f94dc.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*2021-10-20*	Private Report	Verichains Lab
1.1	*2021-10-20*	Private Report	Verichains Lab
1.2	*2021-10-20*	Public Report	Verichains Lab

Table 2. Report versions history