InterFi NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT OF

SMART CONTRACT AUDIT | TEAM KYC | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA

Summary

Auditing Firm	InterFi Network
Architecture	InterFi "Echelon" Auditing Standard
Smart Contract Audit Approved By	Chris | Blockchain Specialist at InterFi Network
Project Overview Approved By	Albert | Marketing Specialist at InterFi Network
Platform	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Consultation Request Date	November 10, 2021
Report Date	November 15, 2021

Audit Summary

InterFi team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- **Spaceship War's smart contract source code has LOW RISK SEVERITY.**
- **Spaceship War has PASSED the smart contract audit.**

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on InterFi's GitHub: **https://github.com/interfinetwork**

Table Of Contents

Project Overview

InterFi was consulted by Spaceship War to conduct the smart contract security audit of their solidity source code.

What Is Spaceship War (Game Story)

The spaceship war game is based on the popular "Star Wars" film, with the events described taking place in a fantasy galaxy. The fantasy story takes place in the year 2080, when human science and technology are at their most advanced.

People have discovered that, in addition to Earth, there are many other creatures and planets with life similar to Earth. Mercury (Mercury), Venus (Venus), and Mars are the closest to Earth. After a long period of peaceful in the galaxy, all four planets face a dangerous threat from monsters that appear unexpectedly from four other planets nearby:

- Jupiter
- Uranus
- Neptune
- Saturn

When faced with the possibility of mass killing. All four planets (Earth-Mercury-Venus-Mars) partnered up to form the "PEACEFUL" alliance in the galaxy. And since then, the "Spaceship War" against alien monsters has begun to make sure true peace for the entire Solar System.

Project	Spaceship War
Blockchain	Binance Smart Chain
Language	Solidity
Contract	0x1b23340f5221FBD2e14f36e5b3E5d833D4D215b5
Telegram	https://t.me/spaceshipwar_community
Twitter	https://twitter.com/SpaceshipWar
Announcements	https://t.me/spaceshipwar_channel

Public logo

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x1b23340f5221FBD2e14f36e5b3E5d833D4D215b5#code

Contract Name: SpaceshipToken

Symbol: SPW

Compiler Version: v0.8.4

Optimization Enabled: Yes with 200 runs

Solidity Source Code On InterFi GitHub

https://github.com/interfinetwork/audited-codes/blob/main/SpaceshipWar.sol

SHA-1 Hash

Solidity source code is audited at hash #0732902a369b6a9b62296db96a8e9f40e80e5d60

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of SpaceshipWar. InterFi has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category	
Smart Contract Vulnerabilities	❖ Re-entrancy ❖ Unhandled Exceptions ❖ Transaction Order Dependency ❖ Integer Overflow ❖ Unrestricted Action ❖ Incorrect Inheritance Order ❖ Typographical Errors ❖ Requirement Violation
Source Code Review	❖ Ownership Takeover ❖ Gas Limit and Loops ❖ Deployment Consistency ❖ Repository Consistency ❖ Data Consistency ❖ Token Supply Manipulation
Functional Assessment	❖ Access Control and Authorization ❖ Operations Trail and Event Generation ❖ Assets Manipulation ❖ Liquidity Access

InterFi's Echelon Audit Standard

The aim of InterFi's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
   - ❖ Review of the specifications, sources, and instructions provided to InterFi to make sure we understand the size, scope, and functionality of the smart contract.
   - ❖ Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.
2. Static, Manual, and Software analysis:
   - ❖ Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - ❖ 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

Automated 3P frameworks used to assess the smart contract vulnerabilities

- ❖ Slither
- ❖ Consensys MythX
- ❖ Consensys Surya
- ❖ Open Zeppelin Code Analyzer
- ❖ Solidity Code Complier

InterFi's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level vulnerabilities are should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level vulnerabilities can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Smart Contract – Static Analysis

Symbol	Meaning
🛑	Function can be modified
💵	Function is payable
🔒	Function is locked
🔐	Function can be accessed
!	Important functionality

SpaceshipToken	Implementation	BEP20		
└	<Constructor>	Public !	🛑	BEP20
└	setP2eAddress	External !	🛑	onlyOwner
└	setMktAddress	External !	🛑	onlyOwner
└	setOperationFee	External !	🛑	onlyOwner
└	addBlacklist	External !	🛑	onlyOwner
└	addExcludeFee	External !	🛑	onlyOwner
└	swapTokensForEthAndSend	Private 🔐	🛑	
└	_transfer	Internal 🔒	🛑	
BEP20	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public !	🛑	NO!
└	getOwner	External !		NO!
└	name	Public !		NO!
└	decimals	Public !		NO!
└	symbol	Public !		NO!
└	totalSupply	Public !		NO!
└	balanceOf	Public !		NO!
└	transfer	Public !	🛑	NO!
└	allowance	Public !		NO!
└	approve	Public !	🛑	NO!
└	transferFrom	Public !	🛑	NO!
└	increaseAllowance	Public !	🛑	NO!
└	decreaseAllowance	Public !	🛑	NO!
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_burnFrom	Internal 🔒	🛑	
IUniswapV2Router02	Interface	IUniswapV2Router01		

```
| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External ❗ | 🛑 |NO❗ |
| └ | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ❗ | 🛑 |NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🛑 |NO❗ |
| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 |NO❗ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🛑 |NO❗ |
||||||
| **IUniswapV2Factory** | Interface |  |||
| └ | feeTo | External ❗ |  |NO❗ |
| └ | feeToSetter | External ❗ |  |NO❗ |
| └ | getPair | External ❗ |  |NO❗ |
| └ | allPairs | External ❗ |  |NO❗ |
| └ | allPairsLength | External ❗ |  |NO❗ |
| └ | createPair | External ❗ | 🛑 |NO❗ |
| └ | setFeeTo | External ❗ | 🛑 |NO❗ |
| └ | setFeeToSetter | External ❗ | 🛑 |NO❗ |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |  | |
| └ | _msgData | Internal 🔒 |  | |
||||||
| **SafeMath** | Library |  |||
| └ | tryAdd | Internal 🔒 |  | |
| └ | trySub | Internal 🔒 |  | |
| └ | tryMul | Internal 🔒 |  | |
| └ | tryDiv | Internal 🔒 |  | |
| └ | tryMod | Internal 🔒 |  | |
| └ | add | Internal 🔒 |  | |
| └ | sub | Internal 🔒 |  | |
| └ | mul | Internal 🔒 |  | |
| └ | div | Internal 🔒 |  | |
| └ | mod | Internal 🔒 |  | |
| └ | sub | Internal 🔒 |  | |
| └ | div | Internal 🔒 |  | |
| └ | mod | Internal 🔒 |  | |
||||||
| **Address** | Library |  |||
| └ | isContract | Internal 🔒 |  | |
| └ | sendValue | Internal 🔒 | 🛑 | |
| └ | functionCall | Internal 🔒 | 🛑 | |
| └ | functionCall | Internal 🔒 | 🛑 | |
| └ | functionCallWithValue | Internal 🔒 | 🛑 | |
| └ | functionCallWithValue | Internal 🔒 | 🛑 | |
| └ | functionStaticCall | Internal 🔒 |  | |
| └ | functionStaticCall | Internal 🔒 |  | |
| └ | functionDelegateCall | Internal 🔒 | 🛑 | |
| └ | functionDelegateCall | Internal 🔒 | 🛑 | |
| └ | verifyCallResult | Internal 🔒 |  | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
```


└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	getOwner	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IUniswapV2Router01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗

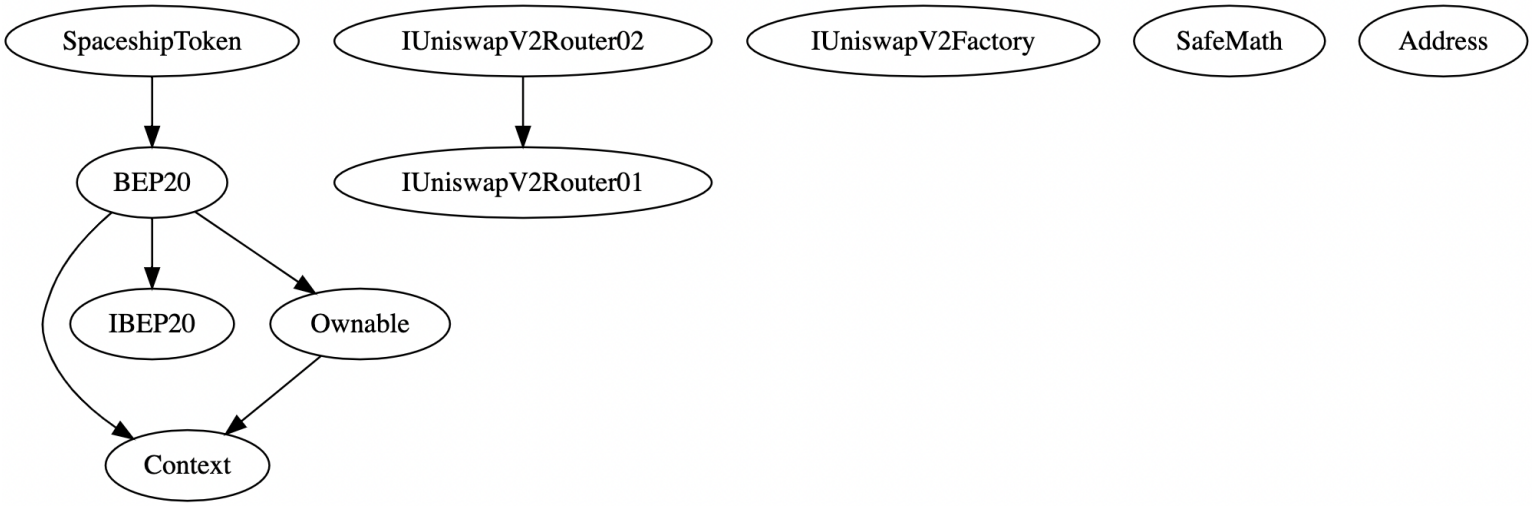
Smart Contract – Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
632eeb23  =>  setP2eAddress(address,uint256)
7f5e3d43  =>  setMktAddress(address,uint256)
9419c8ef  =>  setOperationFee(uint256)
17ca2e41  =>  addBlacklist(address,bool)
97d1ca7e  =>  addExcludeFee(address,bool)
b856ca6c  =>  swapTokensForEthAndSend(uint256,address)
30e0789e  =>  _transfer(address,address,uint256)
893d20e8  =>  getOwner()
06fdde03  =>  name()
313ce567  =>  decimals()
95d89b41  =>  symbol()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
a457c2d7  =>  decreaseAllowance(address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
a22b35ce  =>  _burnFrom(address,uint256)
af2979eb  =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984  =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95  =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
884557bf  =>  tryAdd(uint256,uint256)
```

```
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
c21d36f3  =>  functionStaticCall(address,bytes)
dbc40fb9  =>  functionStaticCall(address,bytes,string)
ee33b7e2  =>  functionDelegateCall(address,bytes)
57387df0  =>  functionDelegateCall(address,bytes,string)
946b5793  =>  verifyCallResult(bool,bytes,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
fc201122  =>  _setOwner(address)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
```


Inheritance Graph



Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
Total Supply	provides information about the total token supply	Yes	**Passed**
Balance Of	provides account balance of the owner's account	Yes	**Passed**
Transfer	executes transfers of a specified number of tokens to a specified address	Yes	**Passed**
Approve	allow a spender to withdraw a set number of tokens from a specified account	Yes	**Passed**
Allowance	returns a set number of tokens from a spender to the owner	Yes	**Passed**
Buy Back	is an action in which the project buys back its tokens from the existing holders usually at a market price	NA	NA
Burn	executes transfers of a specified number of tokens to a burn address	NA	NA
Mint	executes creation of a specified number of tokens and adds it to the total supply	NA	NA
Rebase	circulating token supply adjusts (increases or decreases) automatically according to a token's price fluctuations	NA	NA
Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	**Passed**
Lock	stops or locks all function modules of the smart contract	NA	NA

Review

- Active smart contract owner: **0x4d65dd9673bc4d6183575802a68a1e33e466102d**
- **Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.**
- Smart contract owner can blacklist certain wallets from interacting with the contract function modules.
- Owner can-not lock or burn user assets.
- Owner can-not stop or pause the smart contract.
- Owner can-not mint tokens after launch.
- The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```
string private _name = "SpaceshipToken";

string private _symbol = "SPW";

library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");

function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

    return c;

function div(uint256 a, uint256 b) internal pure returns (uint256) {
    return div(a, b, "SafeMath: division by zero");

function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, "SafeMath: modulo by zero");
```

- The smart contract has 1 low severity issue which may or may not create any functional vulnerability.

{

"resource": " /SpaceshipWar.sol",

"owner": "_generated_diagnostic_collection_name_#0",

"severity": 8, (! Low Severity)

" Expected identifier, got 'LParen”,

"source": "solc",

}

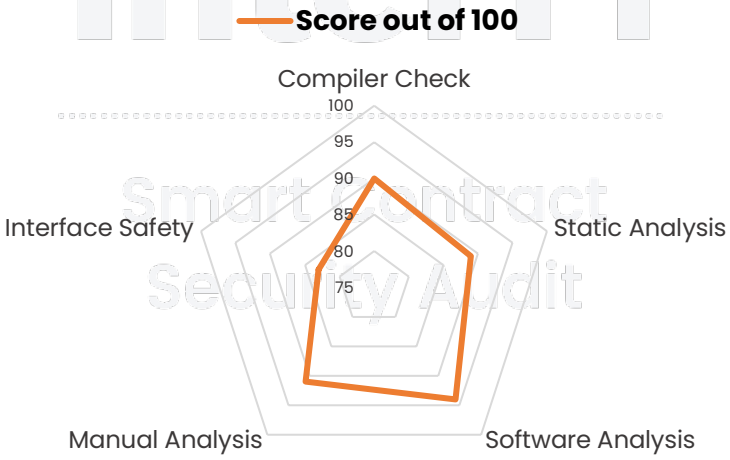
Smart Contract – SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**! Low**
SWC-103	Floating Pragma	**Passed**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELFDESTRUCT Instruction	**Passed**
SWC-107	Re-entrancy	**Passed**
SWC-108	State Variable Default Visibility	**Passed**
SWC-109	Uninitialized Storage Pointer	**Passed**
SWC-110	Assert Violation	**Passed**
SWC-111	Use of Deprecated Solidity Functions	**Passed**
SWC-112	Delegate Call to Untrusted Callee	**Passed**
SWC-113	DoS with Failed Call	**Passed**
SWC-114	Transaction Order Dependence	**Passed**
SWC-115	Authorization through tx.origin	**Passed**
SWC-116	Block values as a proxy for time	**Passed**
SWC-117	Signature Malleability	**Passed**
SWC-118	Incorrect Constructor Name	**Passed**

SWC-119	Shadowing State Variables	**Passed**
SWC-120	Weak Sources of Randomness from Chain Attributes	**Passed**
SWC-121	Missing Protection against Signature Replay Attacks	**Passed**
SWC-122	Lack of Proper Signature Verification	**Passed**
SWC-123	Requirement Violation	**Passed**
SWC-124	Write to Arbitrary Storage Location	**Passed**
SWC-125	Incorrect Inheritance Order	**Passed**
SWC-126	Insufficient Gas Griefing	**Passed**
SWC-127	Arbitrary Jump with Function Type Variable	**Passed**
SWC-128	DoS With Block Gas Limit	**Passed**
SWC-129	Typographical Error	**Passed**
SWC-130	Right-To-Left-Override control character (U+202E)	**Passed**
SWC-131	Presence of unused variables	**Passed**
SWC-132	Unexpected Ether balance	**Passed**
SWC-133	Hash Collisions With Multiple Variable Length Arguments	**Passed**
SWC-134	Message call with hardcoded gas amount	**Passed**
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Passed**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	**1 low severity issue identified**
Passed	**41 functions and instances verified and passed**



Compiler Check	90
Static Analysis	89
Software Analysis	94
Manual Analysis	91
Interface Safety	83

Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

Spaceship War's smart contract source code has LOW RISK SEVERITY.

Spaceship War has PASSED the smart contract audit.

Note for stakeholders

- Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.
- Make sure that the project team's KYC/identity is verified by an independent firm, e.g., InterFi.
- Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in project's longevity. It is recommended to have multiple liquidity providers.
- Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.
- Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Important Disclaimer

InterFi Network provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without InterFi's prior written consent.**

InterFi provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, InterFi 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.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided on this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

About InterFi Network

InterFi Network provides intelligent blockchain solutions. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **InterFi's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.**

InterFi is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. **InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit https://interfi.network

To view our audit portfolio, visit https://github.com/interfinetwork

To book an audit, message https://t.me/interfiaudits

@INTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA