Smart Contract
Security Audit
V1

BSP Smart Contract

6/10/2022

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

Table of Contents

Table of Contents

Background

Project Information

- Smart Contract Information
- Executive Summary

File and Function Level Report

- **File in Scope:**

Issues Checking Status

- Severity Definitions
- Audit Findings

Automatic testing

- Testing proves
- Inheritance graph
- Call graph

Unified Modeling Language (UML)

Functions signature

Automatic general report

Conclusion

Disclaimer

Background

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Project Information

- **Platform**: Tron
- **Contract Address**: TLzCuwHjyg9qQyuMFPHs1x6iwzCVQnpT2Z
- **Code:**

https://tronscan.io/#/contract/TLzCuwHjyg9qQyuMFPHs1x6iwzCVQnpT2Z/code

Smart Contract Information

- Name: BSP
- More Info: BSP is a staking smart contract in which users earn amazing rewards by staking USDT-Token and referring it to other users. The staked USDTs are safe and cannot be withdrawn even by its deployer.

Contracts address deployed to test net (Tron)

BSP smart contract on Tron test net to test every function by the auditor.

https://nile.tronscan.org/#/contract/TTcUmnwdg9hJQJfZz1ue7E8qxt9zKK885R

Executive Summary

According to our assessment, the customer`s solidity smart contract is **“SECURED”**.

Well Secured	
Secured	✓
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 2 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

BSP.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
BSP.sol	**d76fd2e6f6b7ddd3a88630466d1ce623fe0c1d881915bf2eb73e08065c21bd71**	TLzCuwHjyg9qQyuMFPHs1x6iwzCVQnpT2Z

- Contract: BSP
- Inherit: context
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
balStatus	✓	Read / public	**Passed**
dayLuckUsers	✓	Read / public	**Passed**
dayLuckUsersDeposit	✓	Read / public	**Passed**
dayTopUsers	✓	Read / public	**Passed**
defaultRefer	✓	Read / public	**Passed**
depositors	✓	Read / public	**Passed**
feeReceivers	✓	Read / public	**Passed**
getCurDay	✓	Read / public	**Passed**
getCurSplit	✓	Read / public	**Passed**
getDayLuckLength	✓	Read / public	**Passed**
getDepositorsLength	✓	Read / public	**Passed**
getMaxFreezing	✓	Read / public	**Passed**

getOrderLength	✓	Read / public	**Passed**
getTeamDeposit	✓	Read / public	**Passed**
getTeamUsersLength	✓	Read / public	**Passed**
isFreezeReward	✓	Read / public	**Passed**
lastDistribute	✓	Read / public	**Passed**
level4Users	✓	Read / public	**Passed**
luckPool	✓	Read / public	**Passed**
orderInfos	✓	Read / public	**Passed**
rewardInfo	✓	Read / public	**Passed**
starPool	✓	Read / public	**Passed**
startTime	✓	Read / public	**Passed**
teamUsers	✓	Read / public	**Passed**
topPool	✓	Read / public	**Passed**
totalUser	✓	Read / public	**Passed**
usdt	✓	Read / public	**Passed**
userInfo	✓	Read / public	**Passed**
userLayer1DayDeposit	✓	Read / public	**Passed**
deposit	✓	Write / public	**Passed**
depositBySplit	✓	Write / public	**Passed**
distributePoolRewards	✓	Write / public	**Passed**
withdraw	✓	Write / public	**Passed**
register	✓	Write / public	**Passed**
transferBySplit	✓	Write / public	**Passed**

Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	**Passed**
2	Race conditions and Reentrancy. Cross-function race conditions.	**Passed**
3	Possible delays in data delivery.	**Passed**
4	Oracle calls.	**Passed**
5	Design Logic.	**Passed**
6	Timestamp dependence.	**Passed**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed with Notes**
10	Methods execution permissions.	**Passed**
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	**Passed**
12	The impact of the exchange rate on the logic.	**Passed**
13	Private user data leaks.	**Passed**
14	Malicious Event log.	**Passed**
15	Scoping and Declarations.	**Passed**
16	Uninitialized storage pointers.	**Passed**
17	Arithmetic accuracy.	**Passed**

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Note	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

Audit Findings

Critical:

No Critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found.

Medium:

No Medium severity vulnerabilities were found

Low:

#Missing zero address validation

Description

When the owner deploys the smart contract, he has to check for the zero address to make, he didn't add the zero address for USDT contract or default refer. Otherwise, the contract will not work correctly. And in the register function too.

```
constructor(address _usdtAddr, address _defaultRefer, address[2] memory
_feeReceivers) public {

        usdt = IERC20(_usdtAddr);

        feeReceivers = _feeReceivers;

        startTime = 0;

        lastDistribute = block.timestamp;

        defaultRefer = _defaultRefer;

    }
function register(address _referral) external {

        require(userInfo[_referral].totalDeposit > 0 || _referral == defaultRefer,
"invalid refer");

        UserInfo storage user = userInfo[msg.sender];

        require(user.referrer == address(0), "referrer bonded");

        user.referrer = _referral;

        user.start = block.timestamp;
```

```
        _updateTeamNum(msg.sender);

        totalUser = totalUser.add(1);

        emit Register(msg.sender, _referral);

    }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version 2.

#Use of block.timestamp for comparisons

Description

The value of block.timestamp can be manipulated by the miner.
And conditions with strict equality is difficult to achieve - block.timestamp

Remediation

Avoid use of block.timestamp

Status: Acknowledged

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Compiler version is old

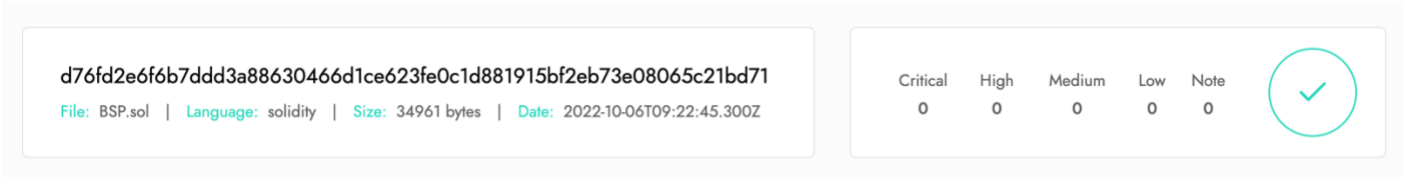
Description

The compiler being used was released 3 years – 3years and half ago. It's recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

Status: Acknowledged.

Automatic Testing

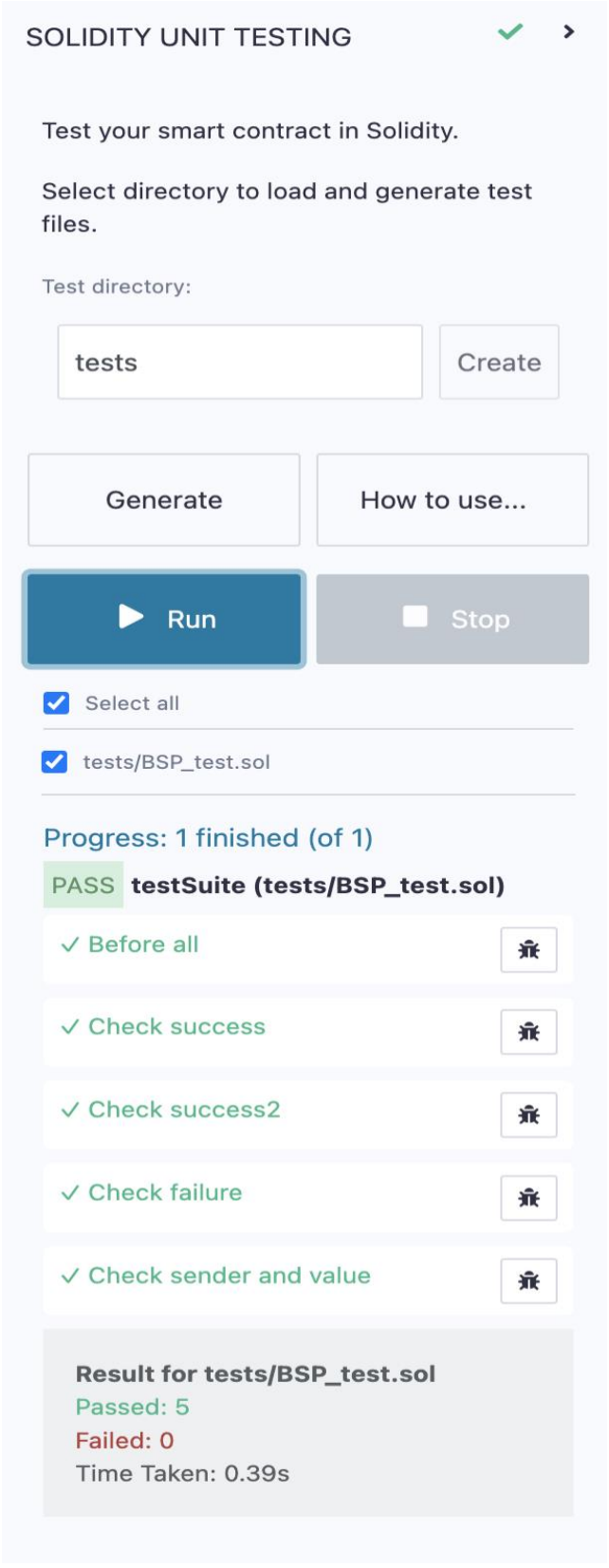
1- Check for security



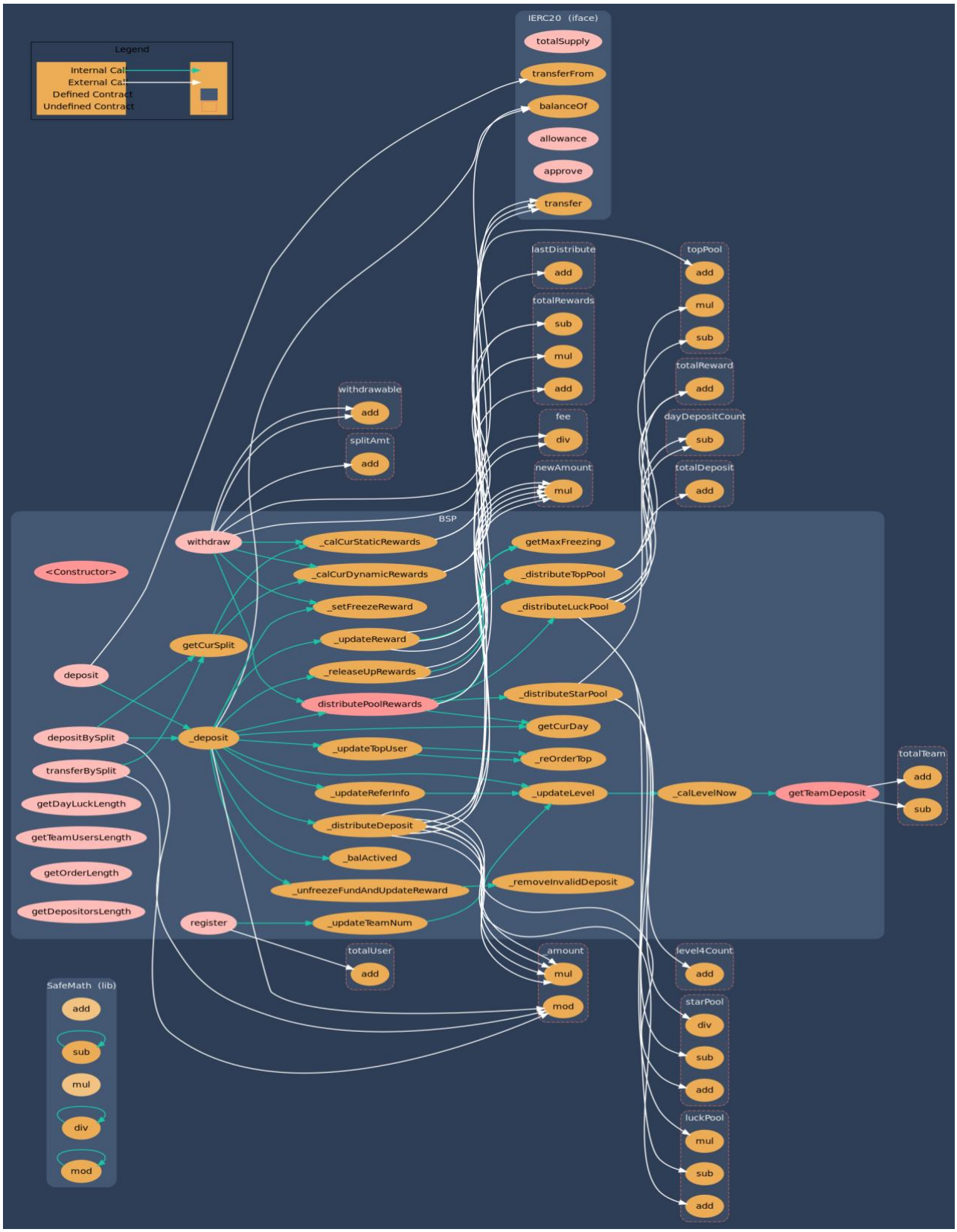
2- SOLIDITY STATIC ANALYSIS

3- Inheritance graph

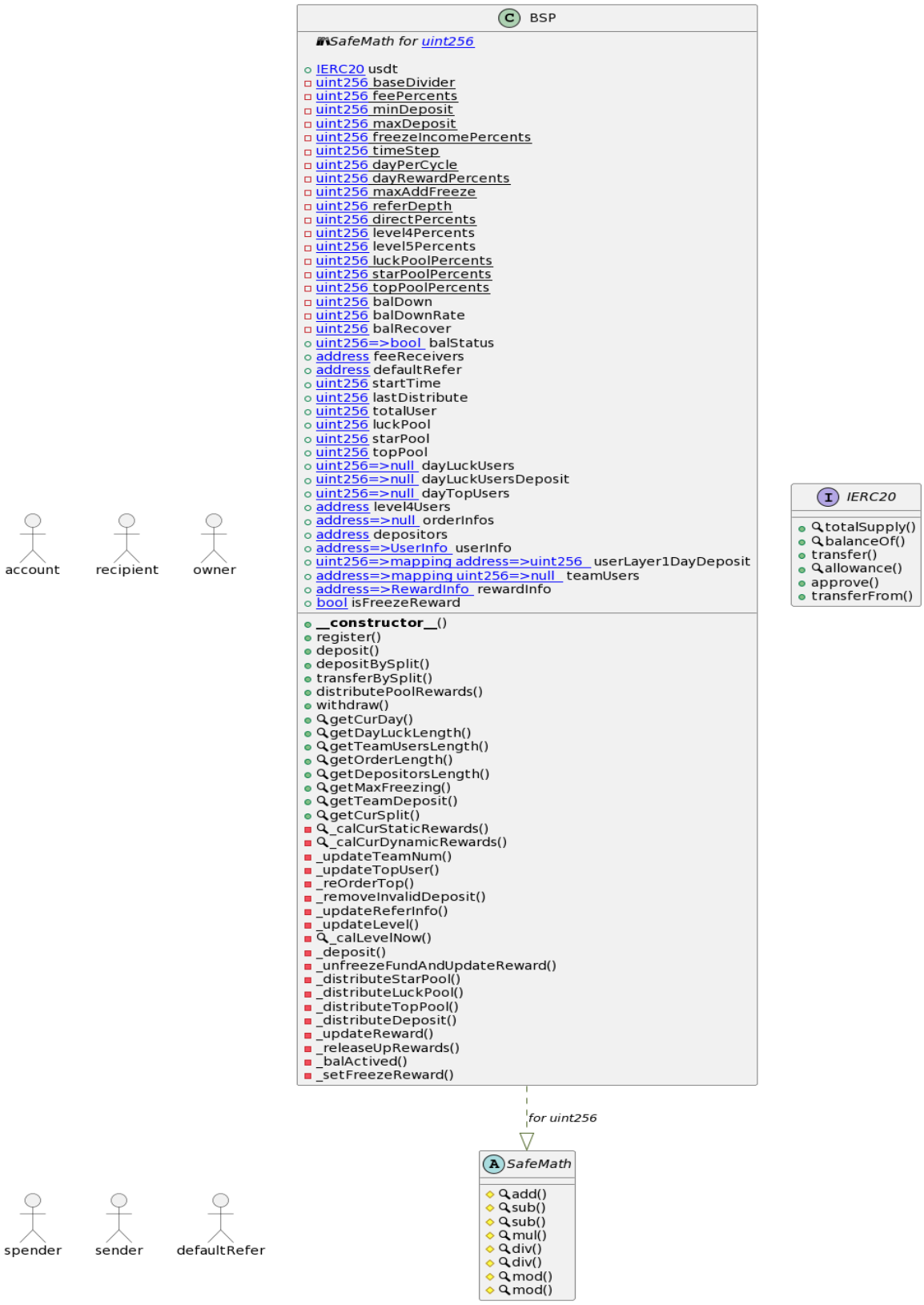
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
transferFrom
balanceOf
allowance
approve
transfer
lastDistribute
add
topPool
add
mul
sub
totalRewards
sub
mul
add
totalReward
add
withdrawable
add
fee
div
dayDepositCount
sub
splitAmt
add
newAmount
mul
totalDeposit
add
BSP
withdraw
_calCurStaticRewards
_calCurDynamicRewards
_setFreezeReward
_updateReward
_releaseUpRewards
distributePoolRewards
getMaxFreezing
_distributeTopPool
_distributeLuckPool
_distributeStarPool
getCurDay
_reOrderTop
_updateLevel
_calLevelNow
getTeamDeposit
totalTeam
add
sub
<Constructor>
getCurSplit
deposit
depositBySplit
transferBySplit
getDayLuckLength
getTeamUsersLength
getOrderLength
getDepositorsLength
_deposit
_updateTopUser
_updateReferInfo
_distributeDeposit
_balActived
_unfreezeFundAndUpdateReward
_removeInvalidDeposit
register
_updateTeamNum
totalUser
add
amount
mul
mod
level4Count
add
starPool
div
sub
add
luckPool
mul
sub
add
SafeMath (lib)
add
sub
mul
div
mod

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
4420e486  =>  register(address)
b6b55f25  =>  deposit(uint256)
c511b345  =>  depositBySplit(uint256)
71a6b69d  =>  transferBySplit(address,uint256)
70abe5fe  =>  distributePoolRewards()
3ccfd60b  =>  withdraw()
00a7a56e  =>  getCurDay()
f7689907  =>  getDayLuckLength(uint256)
a5b5038c  =>  getTeamUsersLength(address,uint256)
de6b8a2e  =>  getOrderLength(address)
2fea20f6  =>  getDepositorsLength()
e402a071  =>  getMaxFreezing(address)
7647e0ff  =>  getTeamDeposit(address)
4c809c66  =>  getCurSplit(address)
3390af2a  =>  _calCurStaticRewards(address)
73586f11  =>  _calCurDynamicRewards(address)
1e1d7b2a  =>  _updateTeamNum(address)
abad60cf  =>  _updateTopUser(address,uint256,uint256)
0fdefd00  =>  _reOrderTop(uint256)
9f913d98  =>  _removeInvalidDeposit(address,uint256)
659b33a2  =>  _updateReferInfo(address,uint256)
c46e6af4  =>  _updateLevel(address)
6d7c9f9a  =>  _calLevelNow(address)
6da1339c  =>  _deposit(address,uint256)
0817d351  =>  _unfreezeFundAndUpdateReward(address,uint256)
34e973c9  =>  _distributeStarPool()
af4b13e8  =>  _distributeLuckPool(uint256)
450e49ac  =>  _distributeTopPool(uint256)
67defb85  =>  _distributeDeposit(uint256)
3538f54e  =>  _updateReward(address,uint256)
963b90f3  =>  _releaseUpRewards(address,uint256)
86ccca6b  =>  _balActived(uint256)
23afd631  =>  _setFreezeReward(uint256)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/BSP.sol	bb353bd18f40a4e685411cd144e871350244a5cc

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
BSP	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	register	External ❗	🛑	NO❗
└	deposit	External ❗	🛑	NO❗
└	depositBySplit	External ❗	🛑	NO❗
└	transferBySplit	External ❗	🛑	NO❗
└	distributePoolRewards	Public ❗	🛑	NO❗
└	withdraw	External ❗	🛑	NO❗
└	getCurDay	Public ❗		NO❗
└	getDayLuckLength	External ❗		NO❗
└	getTeamUsersLength	External ❗		NO❗
└	getOrderLength	External ❗		NO❗
└	getDepositorsLength	External ❗		NO❗
└	getMaxFreezing	Public ❗		NO❗
└	getTeamDeposit	Public ❗		NO❗
└	getCurSplit	Public ❗		NO❗
└	_calCurStaticRewards	Private 🔐		
└	_calCurDynamicRewards	Private 🔐		

└	_updateTeamNum	Private 🔓	🛑	
└	_updateTopUser	Private 🔓	🛑	
└	_reOrderTop	Private 🔓	🛑	
└	_removeInvalidDeposit	Private 🔓	🛑	
└	_updateReferInfo	Private 🔓	🛑	
└	_updateLevel	Private 🔓	🛑	
└	_calLevelNow	Private 🔓		
└	_deposit	Private 🔓	🛑	
└	_unfreezeFundAndUpdateReward	Private 🔓	🛑	
└	_distributeStarPool	Private 🔓	🛑	
└	_distributeLuckPool	Private 🔓	🛑	
└	_distributeTopPool	Private 🔓	🛑	
└	_distributeDeposit	Private 🔓	🛑	
└	_updateReward	Private 🔓	🛑	
└	_releaseUpRewards	Private 🔓	🛑	
└	_balActived	Private 🔓	🛑	
└	_setFreezeReward	Private 🔓	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

Security state of the reviewed contract is “ Secured”.

✓ No volatile code.

✓ No high severity issues were found.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. 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.