

# Smart Contract Security Audit V1

## FastBNB Yield Smart Contract

<https://fastbnb.io/>

16/1/2022



<https://saferico.com/>

[business@saferico.com](mailto:business@saferico.com)

[https://t.me/SFI\\_ANN](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

- **Website:** <https://fastbnb.io/>
- **Telegram group:** <https://t.me/fastbnbio>
- **Platform:** Binance Smart Chain
- **Contract Address:** 0x8C705C8814D5A1E4b7b12ca4c6c745FFf5C42469
- **FastBNB Yield Smart Contract:** FastBNB Binance smart chain-based Binance verified audited contract enables the use of Binance Coin BEP-20 for use in yield investment industry. The Longevity and stability of FastBNB will ultimately bring crypto users, miners, investors under single platform to stake, refer and earn seed income with unlimited opportunities making use of decentralization becomes a perfect yield platform which operates effortlessly 24x7 all the days of year!

## Contracts address deployed to test net (BSC)

FastBNB contract on testnet.bsc (BSC Test Net)

<https://testnet.bscscan.com/address/0xc8d6d74e23586f524e847a65448a57276912c985>

## Executive Summary

According to our assessment, the customer's solidity smart contract is **InSecure**. The owner can take all investors funds using liquidity function.

Well Secured	
<b>Secured</b>	
Poor Secured	
<b>Insecure</b>	✓

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

The files:

FASTBNB.sol

# File and Function Level Report

## File in Scope:

Contract Name	SHA 256 hash	Contract Address
FASTBNB.sol	698d5a50f83be2f0c7f829bc 9d1aa63428490065d7fbc28 dba2cd068fdaf10d2	0x8C705C8814D5A1E4b7b12ca4c6c745FFf5C 42469

- Contract: FASTBNB
- Inherit: Context, SafeMath
- Observation: NOT passed including security check
- Test Report: NOT passed
- Score: NOT passed
- Conclusion: NOT passed

Function	Test Result	Type / Return Type	Score
getContractBalance	✓	Read / public	Passed
commissionWallet	✓	Read / public	Passed
getcurrentseedincome	✓	Read / public	Passed
getDownlineRef	✓	Read / public	Passed
getPlanInfo	✓	Read / public	Passed
getSiteInfo	✓	Read / public	Passed
getUserAmountOfDeposits	✓	Read / public	Passed
getUserAvailable	✓	Read / public	Passed
getUserCheckpoint	✓	Read / public	Passed
getUserDepositInfo	✓	Read / public	Passed
getUserDividends	✓	Read / public	Passed
getUserDownlineCount	✓	Read / public	Passed

getUserInfo	✓	Read / public	Passed
getUserReferralBonus	✓	Read / public	Passed
getUserReferralTotalBonu s	✓	Read / public	Passed
getUserReferralWithdraw n	✓	Read / public	Passed
getUserReferrer	✓	Read / public	Passed
getUserSeedIncome	✓	Read / public	Passed
getUserTotalDeposits	✓	Read / public	Passed
getUserTotalReferrals	✓	Read / public	Passed
getUserTotalSeedWithdra wn	✓	Read / public	Passed
getUserTotalWithdrawn	✓	Read / public	Passed
INVEST_MIN_AMOUN T	✓	Read / public	Passed
PERCENT_STEP	✓	Read / public	Passed
PERCENTS_DIVIDER	✓	Read / public	Passed
PLANPER_DIVIDER	✓	Read / public	Passed
PROJECT_FEE	✓	Read /public	Passed
REFERRAL_PERCENTS	✓	Read /public	Passed
referralCount_	✓	Read / public	Passed
RefUser	✓	Read / public	Passed
SEED_PERCENTS	✓	Read / public	Passed
started	✓	Read / public	Passed
TIME_STEP	✓	Read / public	Passed
totalInvested	✓	Read / public	Passed
totalRefBonus	✓	Read / public	Passed
invest	✓	Write / payable	Passed
withdraw	✓	Write / public	Passed
liquidity	✓	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	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
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
18	Design Logic.	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:**

### #The owner can take all investors funds

#### Description

In the contract there is liquidity function which allow to the owner to withdraw all investors funds using this function.

You can check every liquidity function here you will see the owner withdraw BNB from contract to his wallet

<https://www.bscscan.com/address/0xf7a457999cf09a9da7463ff3105fe5d1760a9b57>

#### Remediation

Send back all investors' funds and delete this function from contract and redeploy it again.

**Status:** OPEN

The screenshot displays a transaction on the BscScan platform. The transaction is successful, with a status of 'Success' and 242 block confirmations. It occurred 12 minutes ago on January 16, 2022, at 09:28:49 AM UTC. The transaction details are as follows:

Field	Value								
Transaction Hash	0xbe5c5a16a19866da025dc4a2a0af9f88dde798b5aed311807131a0e619a8018c								
Status	Success								
Block	14412129 (242 Block Confirmations)								
Timestamp	12 mins ago (Jan-16-2022 09:28:49 AM +UTC)								
From	0xf7a457999cf09a9da7463ff3105fe5d1760a9b57								
To	Contract 0x8c705c8814d5a1e4b7b12ca4c6c745ff5c42469								
Value	0 BNB (\$0.00)								
Transaction Fee	0.000178105 BNB (\$0.09)								
Gas Limit	56,809								
Gas Used by Transaction	35,621 (62.7%)								
Gas Price	0.000000005 BNB (5 Gwei)								
Nonce	603								
Input Data	<table border="1"><thead><tr><th>#</th><th>Name</th><th>Type</th><th>Data</th></tr></thead><tbody><tr><td>0</td><td>amount</td><td>uint256</td><td>18000000000000000000</td></tr></tbody></table>	#	Name	Type	Data	0	amount	uint256	18000000000000000000
#	Name	Type	Data						
0	amount	uint256	18000000000000000000						

The transaction is a transfer of 18 BNB from the contract to the owner's wallet. The input data shows a single parameter 'amount' of type 'uint256' with a value of 18000000000000000000.

#### Check here some of the transactions:

<https://www.bscscan.com/tx/0xbe5c5a16a19866da025dc4a2a0af9f88dde798b5aed311807131a0e619a8018c>

<https://www.bscscan.com/tx/0x7417b58a9f0c5035db0f72ab86858aab353dbd741718d02f8156040abacf0df3>

<https://www.bscscan.com/tx/0xad2a8a1a7dae5164b3cb2ae73ec23c2c52d2749770f975242d1d22be2b295385>  
<https://www.bscscan.com/tx/0xd898015a32fc5bc6c0479470751820c37f3c58a2e9781491fd3f746df767642e>  
<https://www.bscscan.com/tx/0x86b7845bb2bf06015a53b903399d8e0e303e38096bb76d3680bcebb48f373fc1>  
<https://www.bscscan.com/tx/0xf8387cfe3a2ce99c7592f909af7c8de46d6af46fc91eb56e2f298882a9698f46>  
<https://www.bscscan.com/tx/0x66ef057c2915d3fe6f5a6fa5db04d73b6e7200f643a46b2812f53a9baae80f38>  
<https://www.bscscan.com/tx/0x475fc0ed67c78490371f20996665661f2b6cbace405eeb2132b13523e3d0c5c5>  
<https://www.bscscan.com/tx/0x08b563f891d45d97cb468c732ccb4044dda5463c5fc5379d04424ee4876089a8>  
<https://www.bscscan.com/tx/0xd6b97b93d93a6081bb74aaa4ac2e7ea9b5d056be980c4881842fb899de298a7f>

you can see the total amount send and transfer from the owner wallet



## #The principal deposit cannot be withdrawn.

### Description

The project's business logic is different from most DeFi projects in the space, as users' principal deposits cannot be withdrawn by any means. The only return users can get are dividend and referral rewards. Inexperienced and reckless users might miss the aforementioned fact when depositing their principal into the contracts and expect to get their principal back.

### Remediation

If the current implementation is the intended design, we advise the team to present the facts and potential risks to the community in a clear and notable fashion.

## **Alleviation**

[FASTBNB Team]: FASTBNB is a high-risk project in which users can only receive dividends and referral rewards, and the main deposit cannot be withdrawn in any way. We inform all users of this , the site contains information about potential risks.

## **# Unable to withdraw all bonuses when the contract balance is insufficient**

### **Description**

in this contract, users cannot withdraw their reward bonus when there is not enough balance in the contract. The maximum withdrawal amount is equal to the remaining part. User may lose their funds due to the flawed design. This leads to a potential attack method: The early investors with first-hand information deposit a certain amount of BNB in the contract before the project is widely known. By the time the project is well recognized and investors start to deposit BNB, the reward early investors earned during this period might be larger than the current contract balance. They can call the "withdraw" function immediately and drain all of the BNB deposited in the contract. Later investors can no longer get any rewards and their principal will be lost as well.

### **Remediation**

We advise the dev team to modify the current code logic and make sure users can fully receive their rewards. At the same time, we advise the team to present the fact and potential risks to the community in a clear and notable fashion.

## **Alleviation**

[FASTBNB Team]: We cannot change the logic, as mentioned earlier and how we notify users about it - the project carries a high risk and is experimental.

### **High:**

No High severity vulnerabilities were found

### **Medium:**

No Medium severity vulnerabilities were found.

### **Low:**

## **#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**

## # Inaccurate value of user.totalbonus

### Description

User.bonus is wrongly added to user.totalbonus In this line, user.bonus represents the leftover reward that has not been issued to users because of the insufficient balance. This part of the referral bonus is included in the totalBonus , and should not be added again

```
if (contractBalance < totalAmount) {  
    user.bonus = totalAmount.sub(contractBalance);  
    user.totalBonus = user.totalBonus.add(user.bonus);  
    totalAmount = contractBalance;  
}  
  
users[[uplineupline]]..totalBonus totalBonus == users  
users[[uplineupline]]..totalBonustotalBonus..addadd( (amountamount))
```

### Remediation

We advise the dev team to remove the aforementioned line from the codebase.

Status: **Acknowledged**

## Very Low:

No Very Low severity vulnerabilities were found.

### Notes:

## # Redundant code

### Description

The variable totalRefBonus is declared but not used. It does not affect the functionality of the codebase.

### Recommendation

We advise totalRefBonus related code to be removed from the codebase

Status: **Acknowledged**

# Automatic Testing

## 1- Check for security

698d5a50f83be2f0c7f829bc9d1aa63428490065d7fbc28dba2cd068fdaf10d2

File: FASTB... | Language: solidity | Size: 59454 bytes | Date: 2022-01-15T05:54:48.905Z

Critical	High	Medium	Low	Note
0	0	0	0	0



## 2- SOLIDITY STATIC ANALYSIS

### SOLIDITY STATIC ANALYSIS

☒ Select all

☒ Autorun

Run

▼ Security

☒ Select Security

- ☒ **Transaction origin:**  
'tx.origin' used
- ☒ **Check-effects-interaction:**  
Potential reentrancy bugs
- ☒ **Inline assembly:**  
Inline assembly used
- ☒ **Block timestamp:**  
Can be influenced by miners
- ☒ **Low level calls:**  
Should only be used by experienced devs
- ☒ **Block hash:**  
Can be influenced by miners
- ☒ **Selfdestruct:**  
Contracts using destructured contract can be broken

▼ Gas & Economy

☒ Select Gas & Economy

- ☒ **Gas costs:**  
Too high gas requirement of functions
- ☒ **This on local calls:**  
Invocation of local functions via 'this'
- ☒ **Delete dynamic array:**  
Use require/assert to ensure complete deletion
- ☒ **For loop over dynamic array:**  
Iterations depend on dynamic array's size
- ☒ **Ether transfer in loop:**  
Transferring Ether in a for/while/do-while loop

### SOLIDITY STATIC ANALYSIS

▼ ERC

☒ Select ERC

- ☒ **ERC20:**  
'decimals' should be 'uint8'

▼ Miscellaneous

☒ Select Miscellaneous

- ☒ **Constant/View/Pure functions:**  
Potentially constant/view/pure functions
- ☒ **Similar variable names:**  
Variable names are too similar
- ☒ **No return:**  
Function with 'returns' not returning
- ☒ **Guard conditions:**  
Ensure appropriate use of require/assert
- ☒ **Result not used:**  
The result of an operation not used
- ☒ **String length:**  
Bytes length != String length
- ☒ **Delete from dynamic array:**  
'delete' leaves a gap in array
- ☒ **Data truncated:**  
Division on int/uint values truncates the result

### 3- SOLIDITY UNIT TESTING

#### SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

☒ Select all

☒ tests/FASTBNB\_test.sol

Progress: 1 finished (of 1)

PASS

**testSuite**  
(tests/FASTBNB\_test.sol)

✓ Before all

✓ Check success

✓ Check success2

✓ Check failure

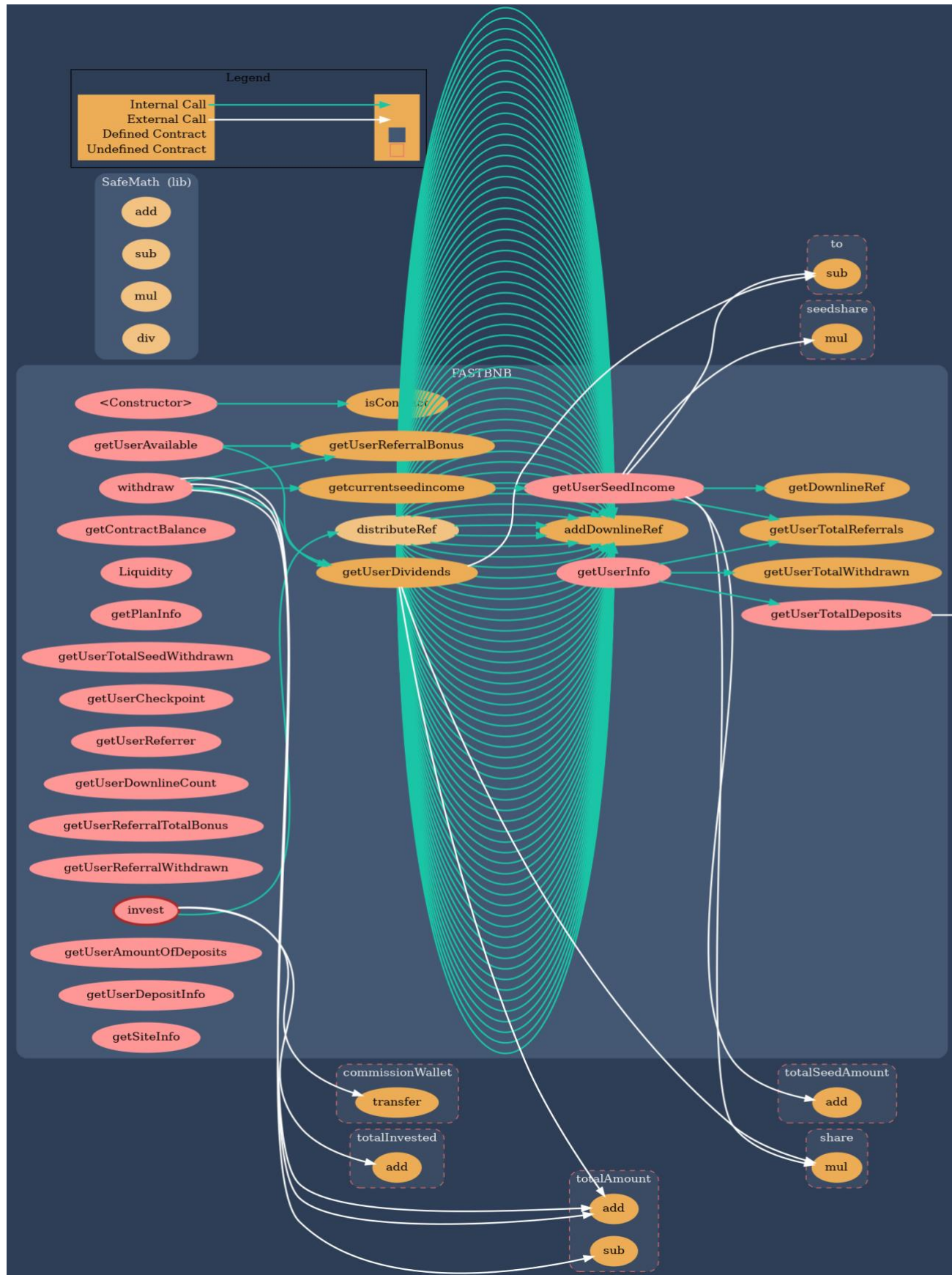
✓ Check sender and value

**Result for tests/FASTBNB\_test.sol**

Passing: 5

Total time: 7.14s

#### 4- Call graph





[illegible]



## Functions signature

```
16279055 => isContract(address)
2b152e40 => getDownlineRef(address,uint256)
0c6821f7 => addDownlineRef(address,address,uint256)
22bc2959 => distributeRef(address,address,bool)
581c5ae6 => invest(address,uint8)
3ccfd60b => withdraw()
6f9fb98a => getContractBalance()
198b2eae => Liquidity(uint256)
aecaa634 => getPlanInfo(uint8)
040a772e => getUserDividends(address)
ded37a0b => getUserSeedIncome(address)
f54b6a21 => getcurrentseedincome(address)
4fae0d76 => getUserTotalSeedWithdrawn(address)
fb4cb32b => getUserTotalWithdrawn(address)
d7ffca91 => getUserCheckpoint(address)
36144c9a => getUserReferrer(address)
03a93c0c => getUserDownlineCount(address)
fbfcb279 => getUserTotalReferrals(address)
e85abe09 => getUserReferralBonus(address)
48c37203 => getUserReferralTotalBonus(address)
6bb18556 => getUserReferralWithdrawn(address)
153ab9df => getUserAvailable(address)
a8aeb6c2 => getUserAmountOfDeposits(address)
7e3abeea => getUserTotalDeposits(address)
c0806b03 => getUserDepositInfo(address,uint256)
4ce87053 => getSiteInfo()
6386c1c7 => getUserInfo(address)
771602f7 => add(uint256,uint256)
b67d77c5 => sub(uint256,uint256)
c8a4ac9c => mul(uint256,uint256)
a391c15b => div(uint256,uint256)
```

# Automatic general report

## Files Description Table



File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/FASTBNB.sol	1384693866df13536c36d7dcf1b108380da311d4

## Contracts Description Table

Contract	Type	Bases		
:-----: :-----: :-----: :-----: :-----				
-----:				
L	**Function Name**	**Visibility**	**Mutability**	
**Modifiers**				
**FASTBNB**	Implementation			
L	<Constructor>	Public	!	NO
L	getDownlineRef	Public	!	NO
L	addDownlineRef	Internal	!	
L	distributeRef	Internal	!	
L	invest	Public	!	NO
L	withdraw	Public	!	NO
L	getContractBalance	Public	!	NO
L	Liquidity	Public	!	NO
L	getPlanInfo	Public	!	NO
L	getUserDividends	Public	!	NO
L	getUserSeedIncome	Public	!	NO
L	getcurrentseedincome	Public	!	NO
L	getUserTotalSeedWithdrawn	Public	!	NO
L	getUserTotalWithdrawn	Public	!	NO
L	getUserCheckpoint	Public	!	NO
L	getUserReferrer	Public	!	NO
L	getUserDownlineCount	Public	!	NO
L	getUserTotalReferrals	Public	!	NO
L	getUserReferralBonus	Public	!	NO
L	getUserReferralTotalBonus	Public	!	NO
L	getUserReferralWithdrawn	Public	!	NO
L	getUserAvailable	Public	!	NO
L	getUserAmountOfDeposits	Public	!	NO
L	getUserTotalDeposits	Public	!	NO
L	getUserDepositInfo	Public	!	NO
L	getSiteInfo	Public	!	NO
L	getUserInfo	Public	!	NO
L	isContract	Internal	!	
**SafeMath**	Library			
L	add	Internal	!	
L	sub	Internal	!	
L	mul	Internal	!	
L	div	Internal	!	

## 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 and developer level documentation (code flow diagram with function level description) not provided, 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 “Insecure”.

- ✓ volatile code.
- ✓ many high severity issues were found.
- ✓ No Contract Ownership Renounced.

# 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.