Smart Contract Security Audit V1

Infinity Token

27/12/2021

https://saferico.com/

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

–

Table of Contents

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**: Binance Smart Chain
- **Contract Address**: 0xb6cc0bb448adD423dcfBAf9242844fA723D8875A

Token Information

- Name: INFINITY
- Total Supply: 3000 Max supply:15000
- Holders: address
- Total transactions:

Contracts address deployed to test net (BSC)

Infinity token contract on testnet.bsc (BSC Test Net)
https://testnet.bscscan.com/address/0x95d70fe2a4c012871549ae36c3aa04f6d451b0de

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**. Because all medium and low issues fixed in version 2.

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, 1 medium, 2 low, 0 very low-level issues and 1note in all solidity files of the contract in version 1 and fixed in version 2.

The files:

Infinity Token.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Infinity Token.sol	**907f5c235848519712a1413265caa3c2b36ca6ddafbe933f29a9cb180aed7de6**	0xb6cc0bb448adD423dcfBAf9242844fA723D8875A

- Contract: Infinity Token
- Inherit: BEP20
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
decimals	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
allowance	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
getOwner	✓	Read / public	**Passed**
owner	✓	Read / public	**Passed**
checkpoints	✓	Read / public	**Passed**
delegates	✓	Read / public	**Passed**
DELEGATION_TYPEHASH	✓	Read / public	**Passed**
DOMAIN_TYPEHASH	✓	Read / public	**Passed**
getCurrentVotes	✓	Read / public	**Passed**
getPriorVotes	✓	Read / public	**Passed**

nonces	✓	Read / public	Passed
numCheckpoints	✓	Read / public	Passed
approve	✓	Write / public	Passed
transferFrom	✓	Write / public	Passed
transfer	✓	Write / public	Passed
renounceOwnership	✓	Write / public	Passed
mint	✓	Write / public	Passed
mint	✓	Write / public	Passed
delegate	✓	Write / public	Passed
delegateBySig	✓	Write / public	Passed
decreaseAllowance	✓	Write / public	Passed
increaseAllowance	✓	Write / public	Passed
transferOwnership	✓	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:

No critical severity vulnerabilities were found.

High:

No High severity vulnerabilities were found

Medium:

# The owner can mint a new token anytime

Description

The owner has the ability to mint more token which can effect on the price of the token; this represents a risk for the users because in that case their funds
will be more less in price.
P.S in the contract there 2 mint function one can’t mine more than max supply, these issues we take about the other one which the owner has the ability to mint tokens as he wants.

Remediation

Make mint() function internal so no one can mint more tokens.

Status: Closed, fixed in version 2

Low:

# Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
uint256 private constant _initialSupply = 3000*10**18;
uint256 private constant _maxSupply = 15000*10**18;
```

Recommendation

Replace the initialization as

```
uint256 private constant _initialSupply = 3000000000000000000000;
uint256 private constant _maxSupply = 15000000000000000000000;
```

Status: Closed, fixed in version 2

#Unused Functions

Description

The following function can be omitted as it's not used anywhere:

_burn ()

Reason

This function is internal and is not called inside any external or public functions.

Status: Closed, fixed in version 2

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Compiler version is old

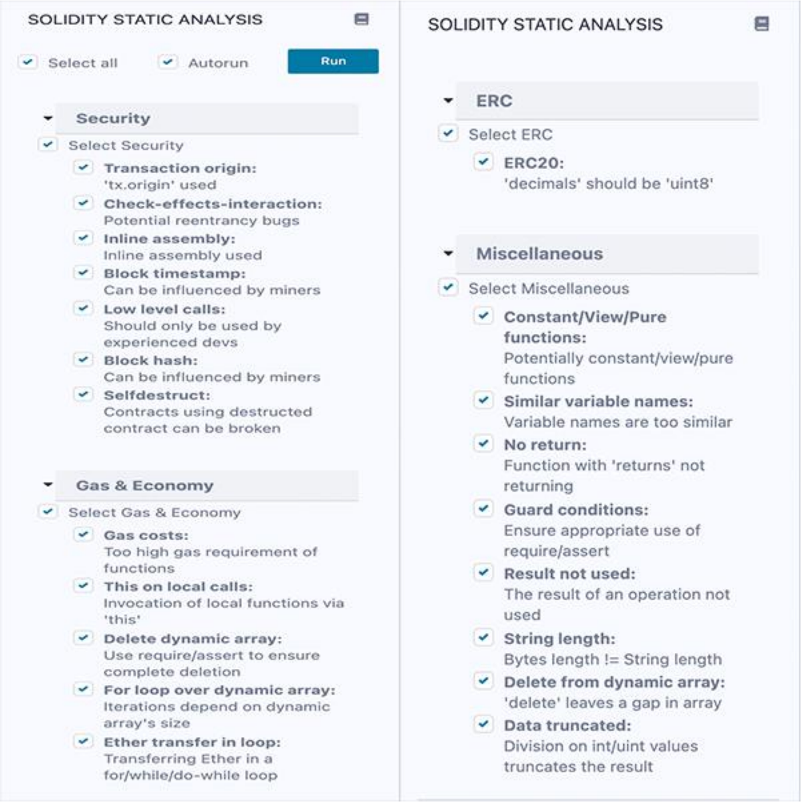
Description

The compiler being used was released a 3 years 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- 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 destructed 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

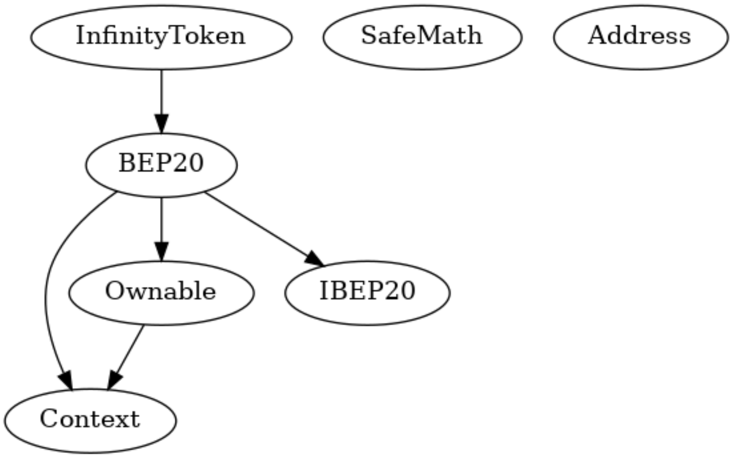
- ☑ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

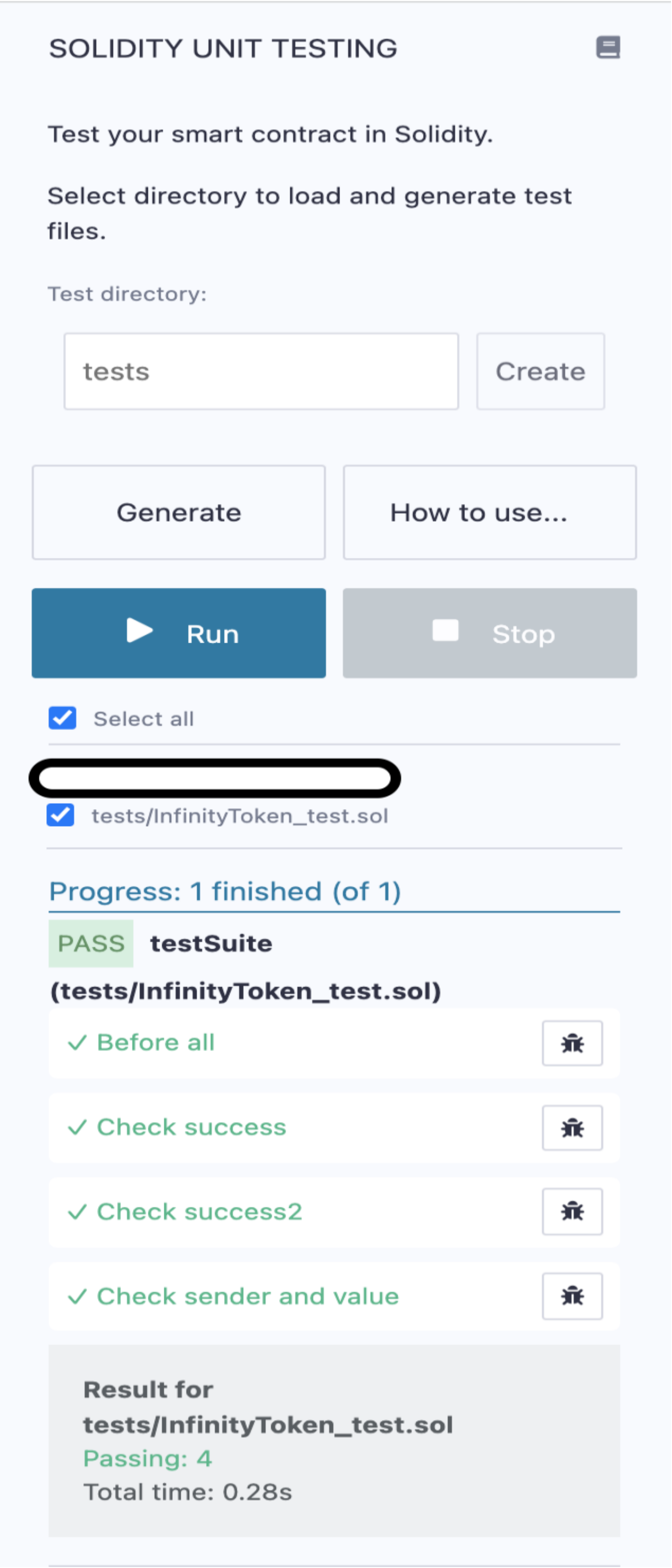
☑ Select Miscellaneous

- ☑ **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

2- Inheritance graph



3- SOLIDITY UNIT TESTING



SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory: tests

☑ Select all

[redacted]

☑ tests/InfinityToken_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/InfinityToken_test.sol)**

✓ Before all

✓ Check success

✓ Check success2

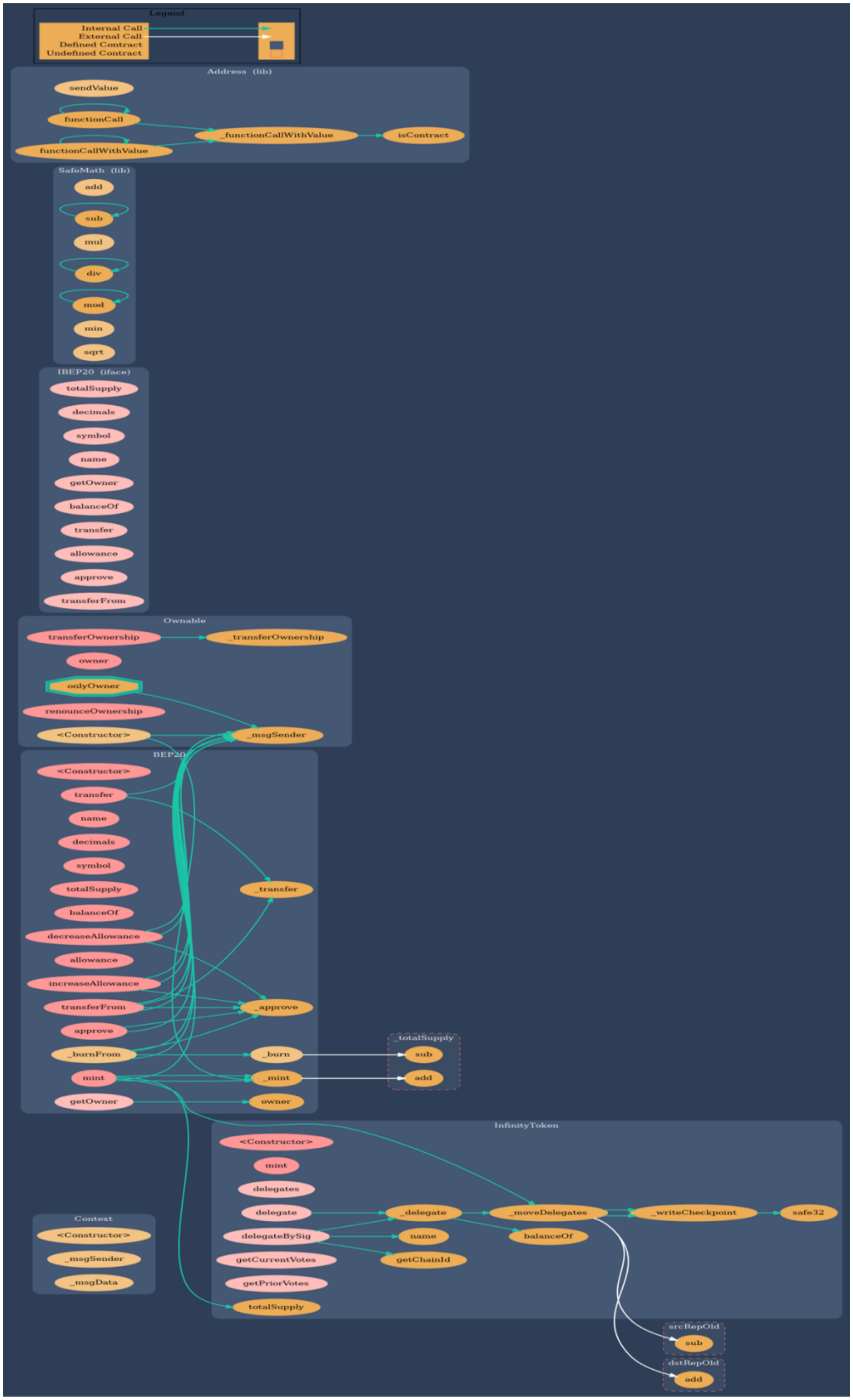
✓ Check sender and value

Result for tests/InfinityToken_test.sol

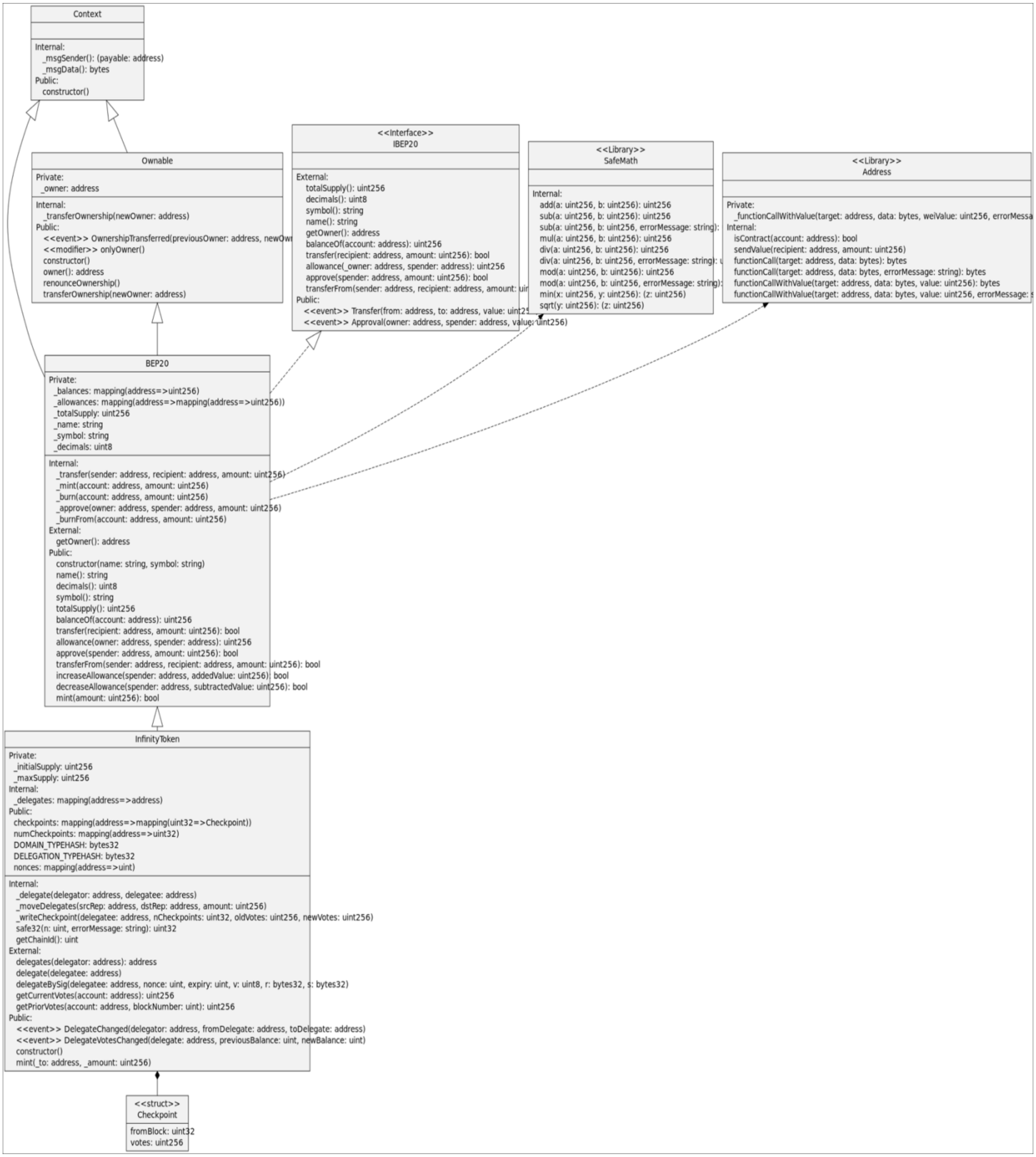
Passing: 4

Total time: 0.28s

4- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Address (lib)
sendValue
functionCall
functionCallWithValue
_functionCallWithValue
isContract
SafeMath (lib)
add
sub
mul
div
mod
min
sqrt
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
Ownable
transferOwnership
_transferOwnership
owner
onlyOwner
renounceOwnership
<Constructor>
_msgSender
BEP20
<Constructor>
transfer
name
decimals
symbol
totalSupply
balanceOf
decreaseAllowance
allowance
increaseAllowance
transferFrom
approve
_burnFrom
mint
getOwner
_transfer
_approve
_burn
_mint
owner
_totalSupply
sub
add
InfinityToken
<Constructor>
mint
delegates
delegate
delegateBySig
getCurrentVotes
getPriorVotes
totalSupply
_delegate
name
getChainId
_moveDelegates
balanceOf
_writeCheckpoint
safe32
Context
<Constructor>
_msgSender
_msgData
srcRepOld
sub
dstRepOld
add

Unified Modeling Language (UML)



Context
Internal:
_msgSender(): (payable: address)
_msgData(): bytes
Public:
constructor()
Ownable
Private:
_owner: address
Internal:
_transferOwnership(newOwner: address)
Public:
<<event>> OwnershipTransferred(previousOwner: address, newOwr
<<modifier>> onlyOwner()
constructor()
owner(): address
renounceOwnership()
transferOwnership(newOwner: address)
<<Interface>>
IBEP20
External:
totalSupply(): uint256
decimals(): uint8
symbol(): string
name(): string
getOwner(): address
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(_owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uir
Public:
<<event>> Transfer(from: address, to: address, value: uint25
<<event>> Approval(owner: address, spender: address, value: uint256)
<<Library>>
SafeMath
Internal:
add(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256, errorMessage: string):
mul(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256, errorMessage: string): u
mod(a: uint256, b: uint256): uint256
mod(a: uint256, b: uint256, errorMessage: string):
min(x: uint256, y: uint256): (z: uint256)
sqrt(y: uint256): (z: uint256)
<<Library>>
Address
Private:
_functionCallWithValue(target: address, data: bytes, weiValue: uint256, errorMessa
Internal:
isContract(account: address): bool
sendValue(recipient: address, amount: uint256)
functionCall(target: address, data: bytes): bytes
functionCall(target: address, data: bytes, errorMessage: string): bytes
functionCallWithValue(target: address, data: bytes, value: uint256): bytes
functionCallWithValue(target: address, data: bytes, value: uint256, errorMessage: s
BEP20
Private:
_balances: mapping(address=>uint256)
_allowances: mapping(address=>mapping(address=>uint256))
_totalSupply: uint256
_name: string
_symbol: string
_decimals: uint8
Internal:
_transfer(sender: address, recipient: address, amount: uint256)
_mint(account: address, amount: uint256)
_burn(account: address, amount: uint256)
_approve(owner: address, spender: address, amount: uint256)
_burnFrom(account: address, amount: uint256)
External:
getOwner(): address
Public:
constructor(name: string, symbol: string)
name(): string
decimals(): uint8
symbol(): string
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
increaseAllowance(spender: address, addedValue: uint256): bool
decreaseAllowance(spender: address, subtractedValue: uint256): bool
mint(amount: uint256): bool
InfinityToken
Private:
_initialSupply: uint256
_maxSupply: uint256
Internal:
_delegates: mapping(address=>address)
Public:
checkpoints: mapping(address=>mapping(uint32=>Checkpoint))
numCheckpoints: mapping(address=>uint32)
DOMAIN_TYPEHASH: bytes32
DELEGATION_TYPEHASH: bytes32
nonces: mapping(address=>uint)
Internal:
_delegate(delegator: address, delegatee: address)
_moveDelegates(srcRep: address, dstRep: address, amount: uint256)
_writeCheckpoint(delegatee: address, nCheckpoints: uint32, oldVotes: uint256, newVotes: uint256)
safe32(n: uint, errorMessage: string): uint32
getChainId(): uint
External:
delegates(delegator: address): address
delegate(delegatee: address)
delegateBySig(delegatee: address, nonce: uint, expiry: uint, v: uint8, r: bytes32, s: bytes32)
getCurrentVotes(account: address): uint256
getPriorVotes(account: address, blockNumber: uint): uint256
Public:
<<event>> DelegateChanged(delegator: address, fromDelegate: address, toDelegate: address)
<<event>> DelegateVotesChanged(delegate: address, previousBalance: uint, newBalance: uint)
constructor()
mint(_to: address, _amount: uint256)
<<struct>>
Checkpoint
fromBlock: uint32
votes: uint256

Functions signature

```
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
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)
7ae2b5c7  =>  min(uint256,uint256)
677342ce  =>  sqrt(uint256)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
36455e42  =>  _functionCallWithValue(address,bytes,uint256,string)
a457c2d7  =>  decreaseAllowance(address,uint256)
a0712d68  =>  mint(uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
a22b35ce  =>  _burnFrom(address,uint256)
40c10f19  =>  mint(address,uint256)
587cde1e  =>  delegates(address)
5c19a95c  =>  delegate(address)
c3cda520  =>  delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32)
b4b5ea57  =>  getCurrentVotes(address)
782d6fe1  =>  getPriorVotes(address,uint256)
a28a42b3  =>  _delegate(address,address)
955f9fd8  =>  _moveDelegates(address,address,uint256)
ee59e77f  =>  _writeCheckpoint(address,uint32,uint256,uint256)
869d1f83  =>  safe32(uint256,string)
3408e470  =>  getChainId()
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/InfinityToken.sol |
546f155ae3edb4dee320ac07a0b4e5f696996ee1 |

 Contracts Description Table

|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **Context** | Implementation |  |||
| └ | <Constructor> | Internal 🔒 | 🛑  | |
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Internal 🔒 | 🛑  | |
| └ | owner | Public ❗️ |   |NO❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **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❗️ |
||||||
| **SafeMath** | Library |  |||
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | mod | Internal 🔒 |   | |
| └ | min | Internal 🔒 |   | |
| └ | sqrt | Internal 🔒 |   | |
||||||
| **Address** | Library |  |||
| └ | isContract | Internal 🔒 |   | |
| └ | sendValue | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | _functionCallWithValue | Private 🔐 | 🛑  | |
||||||
| **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❗️ |
| └ | mint | Public ❗️ | 🛑  | onlyOwner |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _mint | Internal 🔒 | 🛑  | |
| └ | _burn | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _burnFrom | Internal 🔒 | 🛑  | |
||||||
| **InfinityToken** | Implementation | BEP20 |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | mint | Public ❗️ | 🛑  | onlyOwner |
| └ | delegates | External ❗️ |   |NO❗️ |
| └ | delegate | External ❗️ | 🛑  |NO❗️ |
| └ | delegateBySig | External ❗️ | 🛑  |NO❗️ |
| └ | getCurrentVotes | External ❗️ |   |NO❗️ |
| └ | getPriorVotes | External ❗️ |   |NO❗️ |
| └ | _delegate | Internal 🔒 | 🛑  | |
| └ | _moveDelegates | Internal 🔒 | 🛑  | |
| └ | _writeCheckpoint | Internal 🔒 | 🛑  | |
| └ | safe32 | Internal 🔒 |   | |
| └ | getChainId | 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 “secured”.

✓ No mint function.
✓ No volatile code.
✓ Not many high severity issues were found.
✓ 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.