SMART CONTRACT AUDIT REPORT

For

Infinity Token (INFINITY)

Prepared By: SFI Team

Prepared for: INFINITY team

Prepared on: 27/11/2021

Table of Content

- ## 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) 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.

• Overview of the audit

The project has 2 file. It contains approx 2939 lines of Solidity code. Most of the functions and state variables are well commented on using the Nat spec documentation, but that does not create any vulnerability.

• Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices automatically.

1. Unit tests passing.

2. Compilator warnings;

3. Race Conditions. Reentrancy. Cross-function Race Conditions. Pitfalls in Race Condition solutions;

4. Possible delays in data delivery;

5. Transaction-Ordering Dependence (front running);

6. Timestamp Dependence;

7. Integer Overflow and Underflow;

8. DoS with (unexpected) Revert;

9. DoS with Block Gas Limit;

10. Call Depth Attack. Not relevant in modern ethereum network

11. Methods execution permissions;

12. Oracles calls;

13. Economy model. It’s important to forecast scenarios when a user is provided with additional economic motivation or faced with limitations. If application logic is based on incorrect economy model, the application will not function correctly and participants will incur financial losses. This type of issue is most often found in bonus rewards systems.
14. The impact of the exchange rate on the logic;
15. Private user data leaks.

- ## Good things in smart contract

  Compiler version is static: -
  => In this file, you have put “pragma solidity 0.6.12;” which is a good way to define the compiler version.

```
pragma solidity 0.6.12;
```

- **SafeMath library: -**

INFINITY is using SafeMath library it is a good thing. It protect the contract from overflow and underflow.

```
library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, 'SafeMath: addition overflow');
        return c;
    }
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, 'SafeMath: subtraction overflow');
    }
```

- **Ownable library : -**
  - Here you INFINITY token using ownable library, Initializes the contract setting the deployer as the initial owner

```
        contract Ownable is Context {
    address private _owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed
newOwner);

    constructor() internal {
        address msgSender = _msgSender();
        _owner = msgSender;
        emit OwnershipTransferred(address(0), msgSender);
    }
```

  - Here you INFINITY token using interface IBEP20 which Returns the amount of tokens in existence, symbol, name , owner and etc. based on IBEP20 interface

```
interface IBEP20 {

    function totalSupply() external view returns (uint256);
    function decimals() external view returns (uint8);
    function symbol() external view returns (string memory);
    function name() external view returns (string memory);
    function getOwner() external view returns (address);
    function balanceOf(address account) external view returns (uint256);
```

- Here you INFINITY token using BEP20 contract which Inherit Context, IBEP20, and Ownable contracts.

```
contract BEP20 is Context, IBEP20, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;

    uint256 private _totalSupply;
    string private _name;
    string private _symbol;
  uint8 private _decimals;
```

- Here you INFINITY using address library which used for Collection of functions related to the address type.

```
library Address {

    function isContract(address account) internal view returns (bool) {
        bytes32 codehash;
        bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        assembly {
            codehash := extcodehash(account)
        }
        return (codehash != accountHash && codehash != 0x0);
    }
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, 'Address: insufficient balance');

        (bool success, ) = recipient.call{value: amount}('');
        require(success, 'Address: unable to send value, recipient may have
reverted');
    }

    function functionCall(address target, bytes memory data) internal returns
(bytes memory) {
        return functionCall(target, data, 'Address: low-level call failed');
    }
```

- **Critical vulnerabilities found in the contract**

  There not Critical severity vulnerabilities found

- **High vulnerabilities found in the contract**

  There not High severity vulnerabilities found

- **Medium vulnerabilities found in the contract**

  There not Medium severity vulnerabilities found

- **Low vulnerabilities found in the contract**

  There not Low severity vulnerabilities found

- **V. Low vulnerabilities found in the contract**

#Similar variable names:

```
  _name = name_;
  _symbol = symbol_;
```

In detail
BEPC20 .(string,string) : Variables have very similar names "_symbol" and "symbol_". Note: Modifiers are currently not considered by this static analysis.

 # Constant/View/Pure functions:

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

In detail
SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

- o **Notes**

#Guard conditions:

```
require(owner() == _msgSender(), "Ownable: caller is not the owner");
require(newOwner != address(0), "Ownable: new owner is the zero address");
```

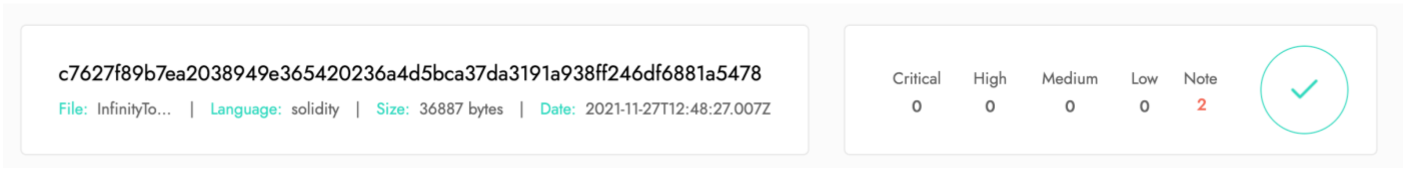
In detail
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code).
Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

#Gas Costs:
```
function getPriorVotes(address account, uint blockNumber)
        external
        view
        returns (uint256)
    {
        require(blockNumber < block.number, "INFINITY::getPriorVotes: not yet
determined");
        uint32 nCheckpoints = numCheckpoints[account];
        if (nCheckpoints == 0) {
            return 0;
        }if (checkpoints[account][nCheckpoints - 1].fromBlock <= blockNumber) {
            return checkpoints[account][nCheckpoints - 1].votes;
        }if (checkpoints[account][0].fromBlock > blockNumber) {
            return 0;
        }
        uint32 lower = 0;
        uint32 upper = nCheckpoints - 1;
        while (upper > lower) {
            uint32 center = upper - (upper - lower) / 2;
            Checkpoint memory cp = checkpoints[account][center];
            if (cp.fromBlock == blockNumber) {
                return cp.votes;
            } else if (cp.fromBlock < blockNumber) {
                lower = center;
            } else {
                upper = center - 1;}}
        return checkpoints[account][lower].votes;
    }
    function _delegate(address delegator, address delegatee)
        internal
    {
        address currentDelegate = _delegates[delegator];
        uint256 delegatorBalance = balanceOf(delegator);
        _delegates[delegator] = delegatee;
```

In detail
Gas requirement of function Infinity Token. get Prior Votes is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed Please avoid loops in your functions or actions that modify large areas of storage
(This includes clearing or copying arrays in storage)

Testing proves:

1- Check for security

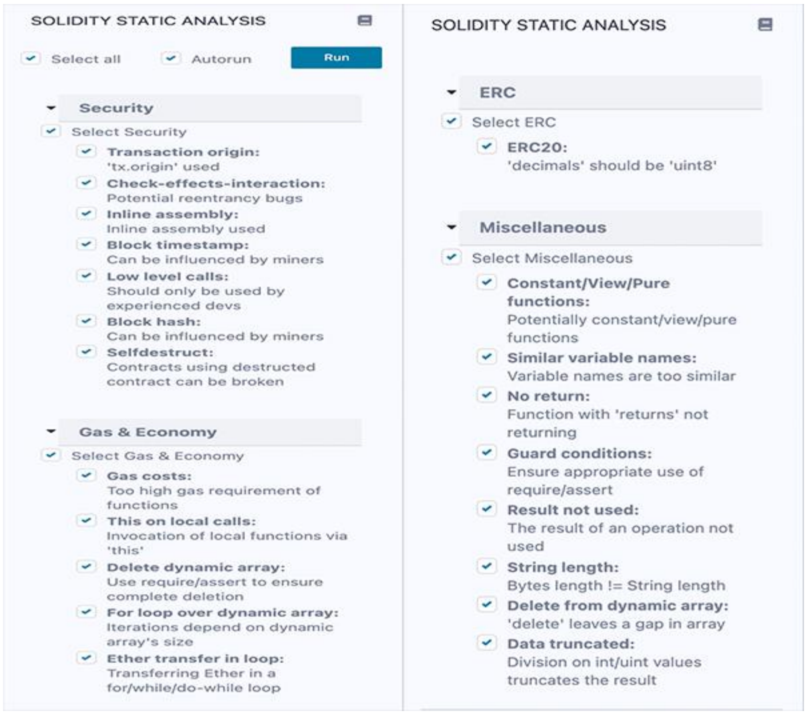


c7627f89b7ea2038949e365420236a4d5bca37da3191a938ff246df6881a5478

File: InfinityTo... | Language: solidity | Size: 36887 bytes | Date: 2021-11-27T12:48:27.007Z

Critical	High	Medium	Low	Note
0	0	0	0	2

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 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

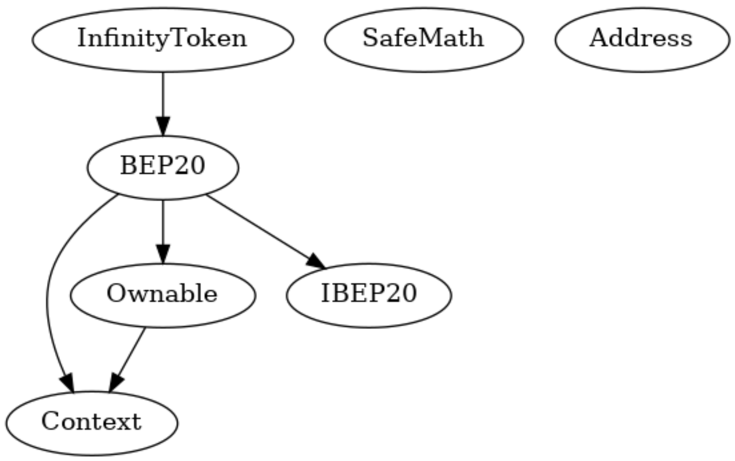
- ✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

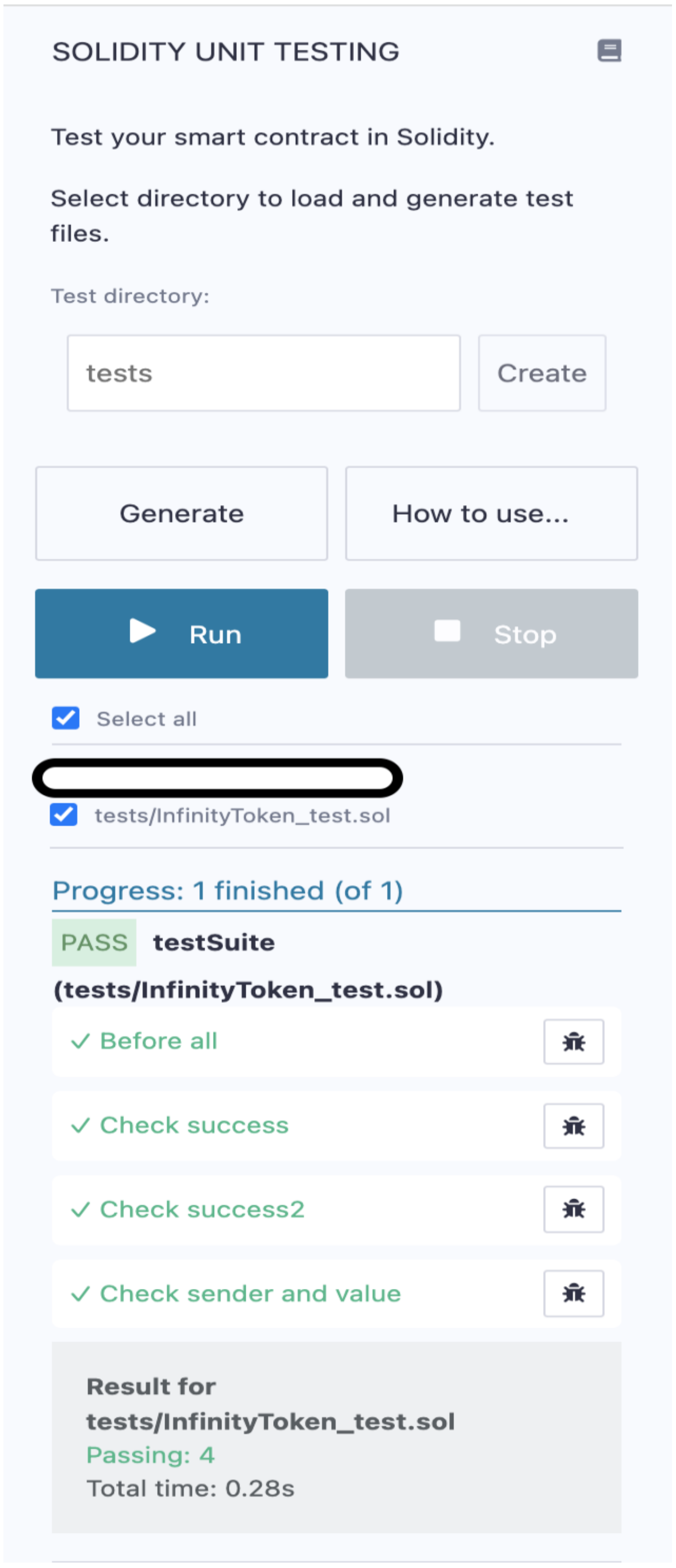
✓ Select Miscellaneous

- ✓ **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

3- Inheritance graph



4- 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

☑ tests/InfinityToken_test.sol

Progress: 1 finished (of 1)

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

✓ Before all

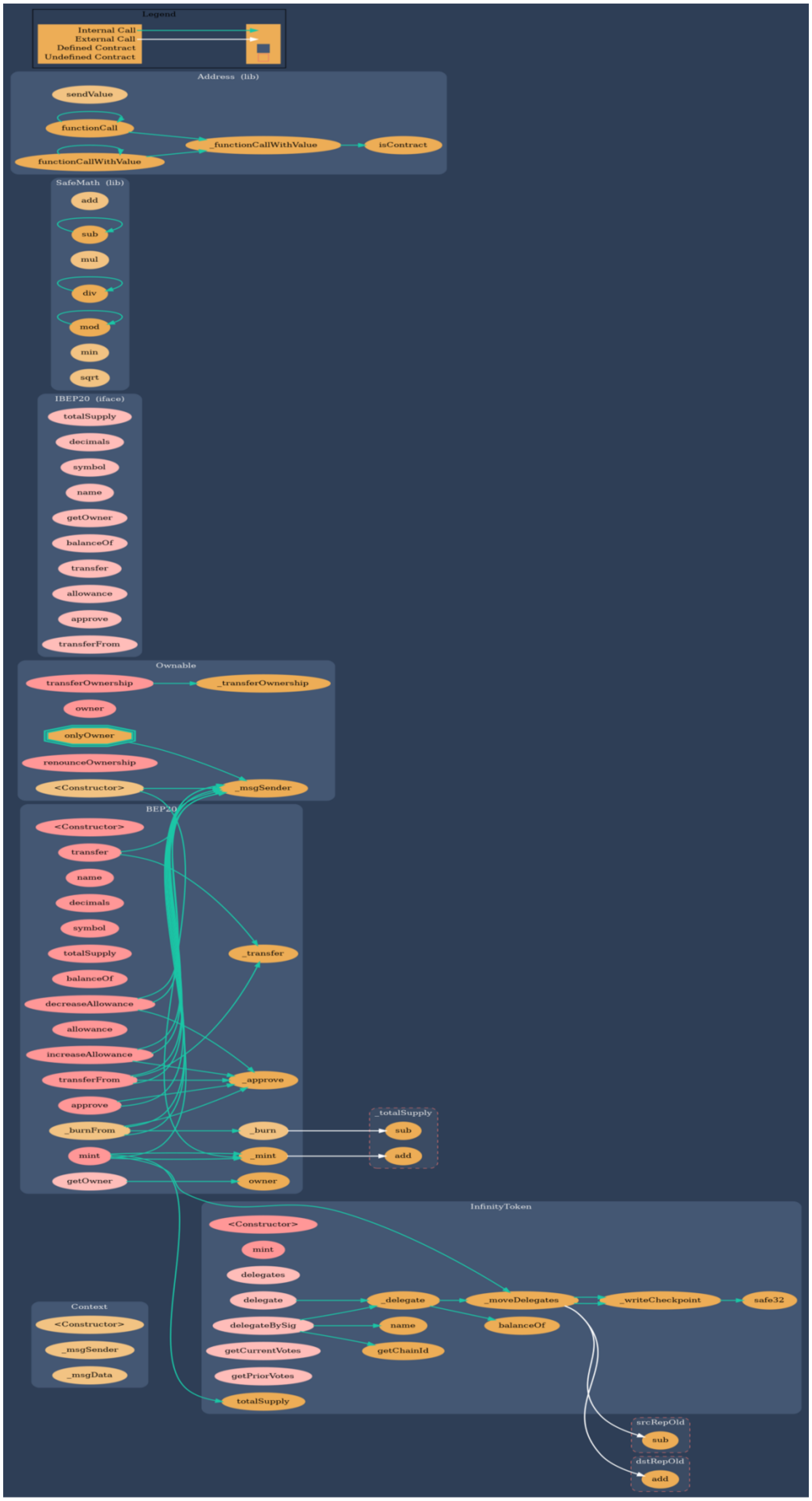
✓ Check success

✓ Check success2

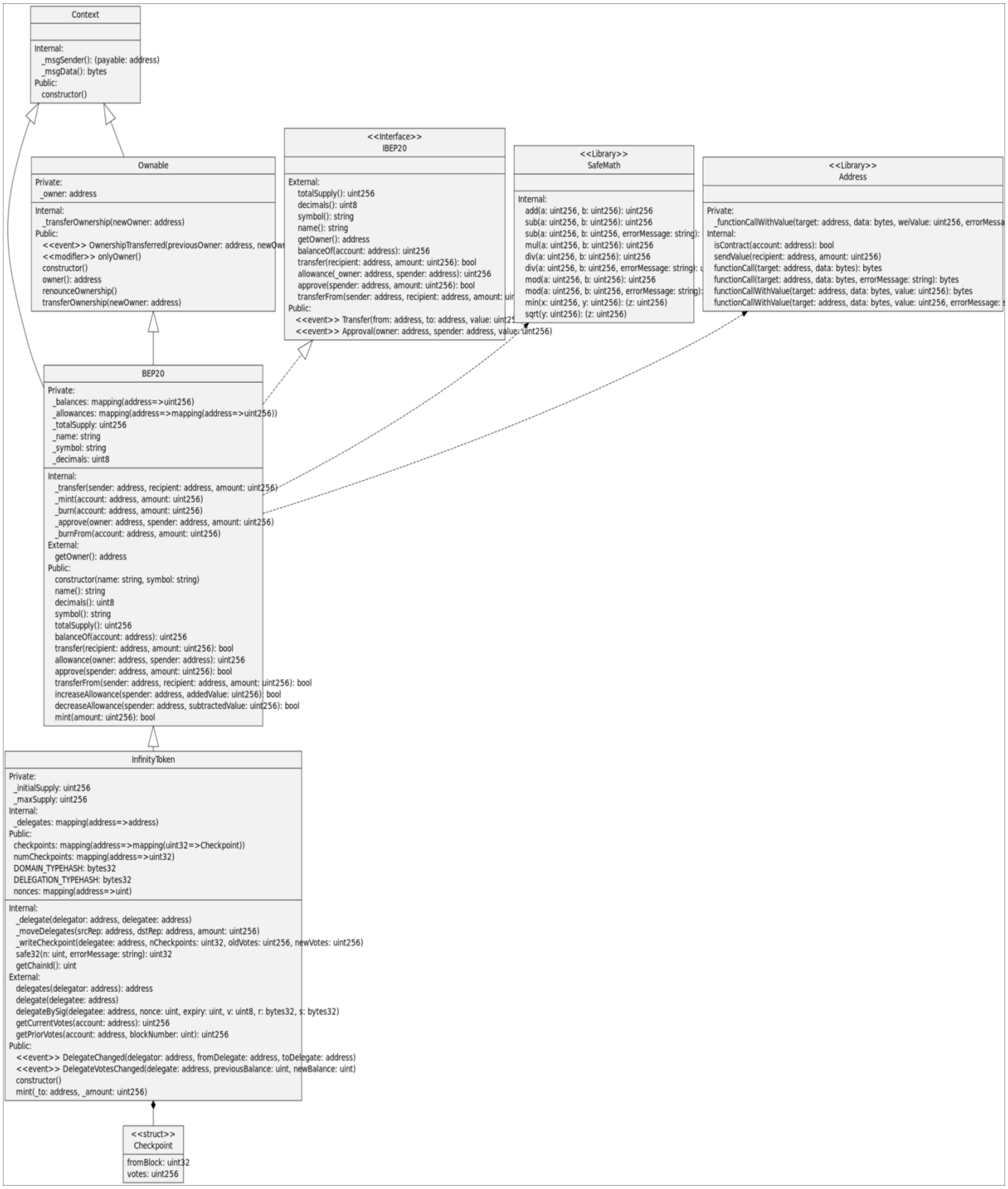
✓ Check sender and value

Result for tests/InfinityToken_test.sol
Passing: 4
Total time: 0.28s

5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Address (lib)
sendValue
functionCall
_functionCallWithValue
isContract
functionCallWithValue
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

- ## 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 |
```

• Summary of the Audit

According to automatically test, the customer`s solidity smart contract is **Secured**.

The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

The test found 0 critical, 0 high, 0 medium, 0 low, 2 Very low issues, and 2 notes.