Smart Contract Security Audit V1

Honeyman Token Smart Contract

23/8/2022

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**: Ethereum
- **Contract Address**: 0xcaba8c0f1ff5c58a6cb79f51872876531f994db6
- **Code Source:**

https://rinkeby.etherscan.io/address/0xcaba8c0f1ff5c58a6cb79f51872876531f994db6#code

Token Information

- Name: honeyman1
- Total Supply: 1,000,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

honeyman1 Token smart contract on Eth test net by the auditor to test every function (ETH Test Net)

https://rinkeby.etherscan.io/address/0xcaba8c0f1ff5c58a6cb79f51872876531f994db6

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Well 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, 1 high, 0 medium, 2 low, 0 very low-level issues and 4 notes in all solidity files of the contract

The files:

honeyman1.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
honeyman1.sol	**20ebd92a085b2b9596888b4e0291a66220b290e0bf8752d0c40d8aeb26c92f90**	0xcaba8c0f1ff5c58a6cb79f51872876531f994db6

- Contract: Token
- Inherit: Context, IERC20, IERC20Metadata
- 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**
Owner	✓	Read / public	**Passed**
_allowances	✓	Read / public	**Passed**
_balances	✓	Read / public	**Passed**
_name	✓	Read / public	**Passed**
_ symbol	✓	Read / public	**Passed**
_ totalSupply	✓	Read / public	**Passed**

burnAddress	✓	Read / public	**Passed**
burnAmount	✓	Read / public	**Passed**
burnPercent	✓	Read / public	**Passed**
charityAddress	✓	Read / public	**Passed**
charityPercent	✓	Read / public	**Passed**
marketingAmount	✓	Read / public	**Passed**
approve	✓	Write / public	**Passed**
TransferFrom	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
burn	✓	Write / public	**Passed**
Prize_Fund	✓	Write / public	**Passed**
Reflections	✓	Write / public	**Passed**
RenounceOwnership	✓	Write / public	**Passed**
SetBurnPercent	✓	Write / public	**Passed**
OwnershipRenounce	✓	Write / public	**Passed**
changeOwner	✓	Write / public	**Passed**
SetCharityAddress	✓	Write / public	**Passed**
SetCharityPercent	✓	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:

#Logic errors

Description

According to the smart contract functionality, the owner can burn any amount of any address tokens, but the auditor had found when he uses the burn function act like the mint function and the total supply increase.

```
function  burn(address account, uint256 amount) onlyOwner  public virtual {
        require(account != address(0), "ERC20: burn to the zero address");
        _beforeTokenTransfer(address(0), account, amount);
        _totalSupply += amount;
        _balances[account] += amount;
        emit Transfer(address(0), account, amount);
    }
```

You can check these transactions:

https://rinkeby.etherscan.io/tx/0x96e6d052a5b2e7f596e90cc14c7790bb302213550a2f368414ed2f1c69133821
https://rinkeby.etherscan.io/tx/0xbfa3ae8c613c8d38a8544c4c5cd09438dad4165e9159832d08c653897d295dc1
https://rinkeby.etherscan.io/tx/0x0bbcba9abf36ca14e3ba22dd873d84cd669a23bc6ab2e58099db8516153f6654

The second error there is 2 Renounce Ownership functions none of them has transferred the ownership to zero address the owner can add any address means these two functions act like transfer the ownership not Renounce Ownership to zero address.

```
function RenounceOwnership(address _DEAD, bool _boo) onlyOwner public returns
(address _dead) {
        safeTransfer = _boo;
        _dead = _DEAD;
    }
function OwnershipRenounce(address _owner) onlyOwner public {
        owner = _owner;
    }
```

You can check these transactions:

https://rinkeby.etherscan.io/tx/0xdb46d55654aae3a782c65750053cf3eb9e025d07b05d4fa10aa5c9c54642d83d
https://rinkeby.etherscan.io/tx/0x51e1577a330dbbebbbcc041b2e074ee593ca4eddd0e5b85978317a6037ed9f38

Remediation
The team should redesign these functions again and test it again.

Status: Closed. Fixed In version 2

Medium:

No Medium severity vulnerabilities were found.

Low:

#Missing zero address validation

Description
When the owner wants to change charity wallet, he has to check for the zero address to make, he didn’t add the zero address. Otherwise, he will lose the fees.

```
function SetCharityAddress(address payable  _charityAddress) onlyOwner public {
        charityAddress = _charityAddress;
    }
```

Remediation
Use the require statement to check for zero addresses.

Status: Closed. Fixed in version 2.

#Owner privileges (In the period when the owner isn’t renounced)

Description

The owner can change the Fees.

```
function SetCharityPercent(uint256 _charityPercent) onlyOwner public {
        charityPercent = _charityPercent;
    }

    function SetBurnPercent(uint256 _burnPercent) onlyOwner public {
        burnPercent = _burnPercent;
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change the fees and give them time if they want to use the old fees.
P.S: This issue is common to the majority of rewards smart contracts.

Status: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Naming Conventions

Description

The contract follows a consistent naming convention where we are private variables with leading"_" and public variables without it. But we have missed to comply to the condition for certain variable names "__totalSupply" which is public.

Remediation

Remove "_" from external variable names and add it to private variable names.

Status: Status Closed. Fixed in version 2.

# Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
    _totalSupply = 1000000000 *10**18;
```

Recommendation

Replace the initialization as

```
    _totalSupply = 1000000000000000000000000000;
```

Status Closed. Fixed in version 2.

#Missing SPDX-License-Identifier:

Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information .

Remediation
Add License Identifier
// SPDX-License-Identifier: MIT

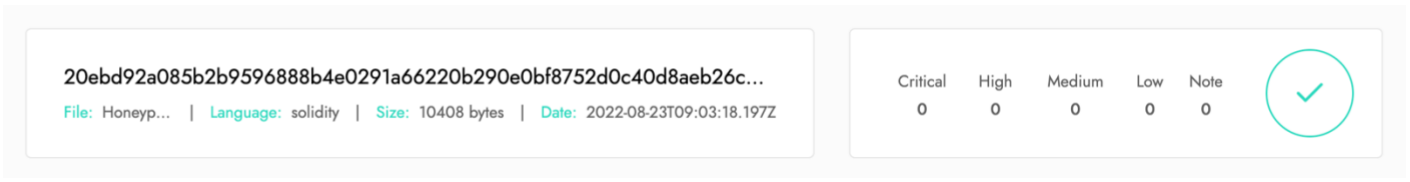
Status: Closed. Fixed In version 2.

#Compiler version is old
Description
The compiler being used was released a year – a year and half ago. It's recommended
to use more recent compiler version, there can be benefits like reduction
in bytecode size etc.

Status: Closed. Fixed In version 2.

Automatic Testing

1- Check for security

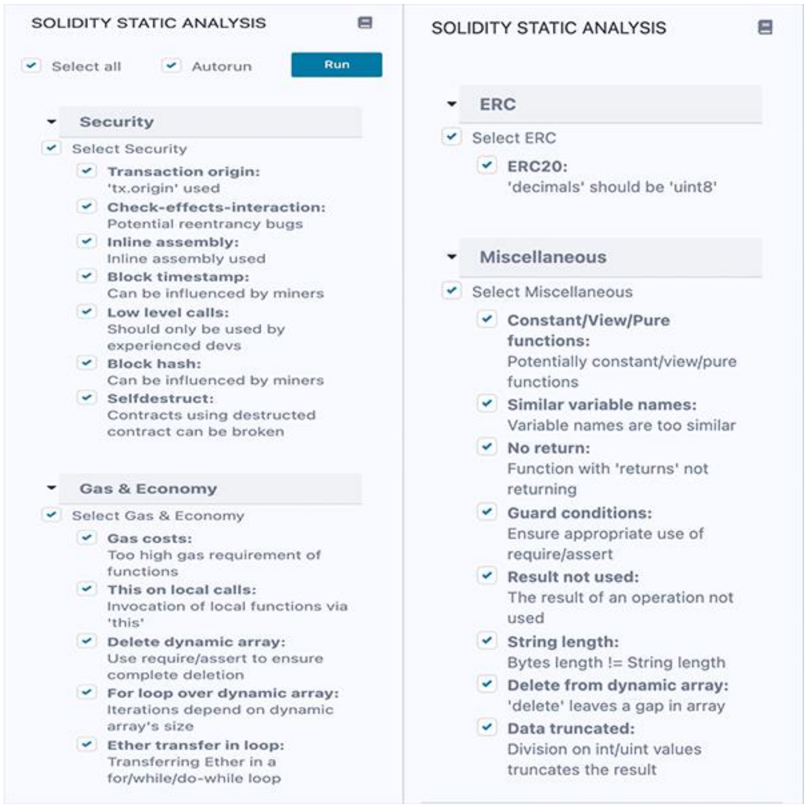


20ebd92a085b2b9596888b4e0291a66220b290e0bf8752d0c40d8aeb26c...

File: Honeyp... | Language: solidity | Size: 10408 bytes | Date: 2022-08-23T09:03:18.197Z

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 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

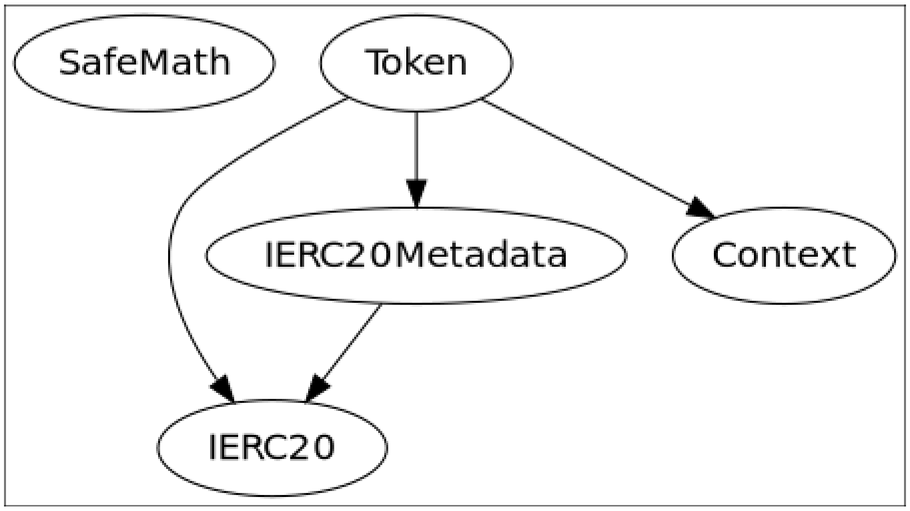
- ✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

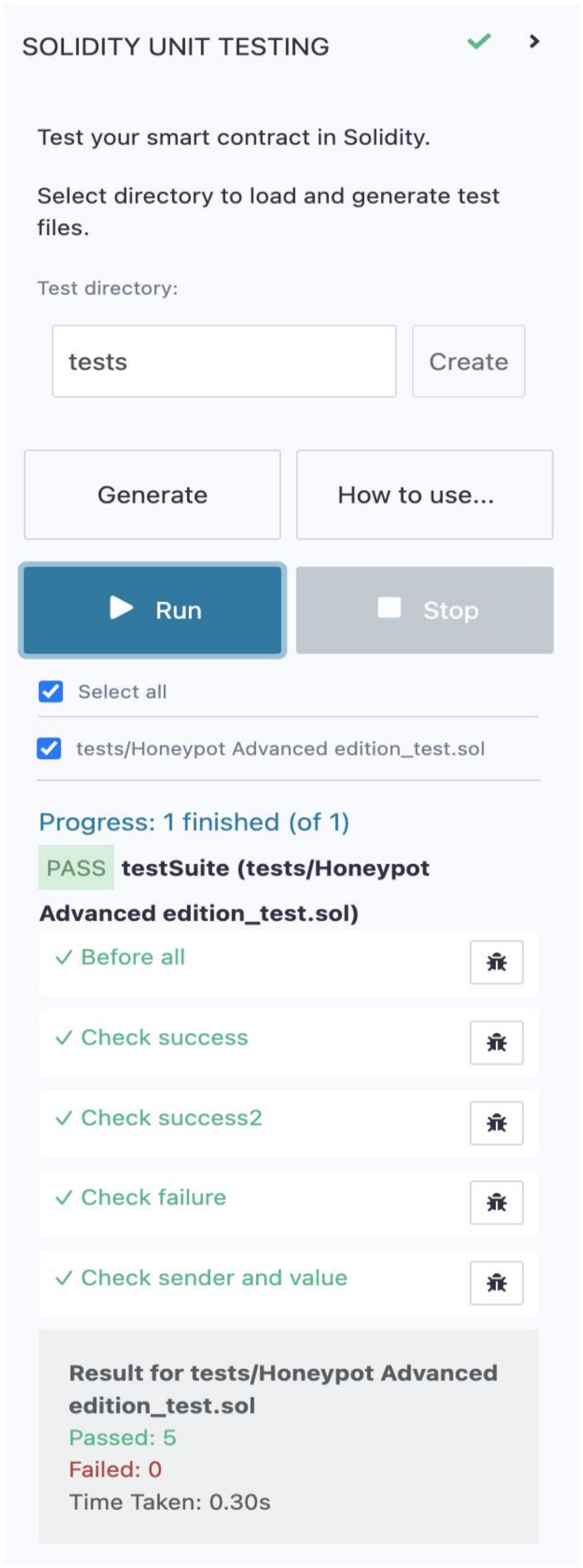
✓ Select Miscellaneous

- ✓ **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/Honeypot Advanced edition_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/Honeypot Advanced edition_test.sol)**

✓ Before all

✓ Check success

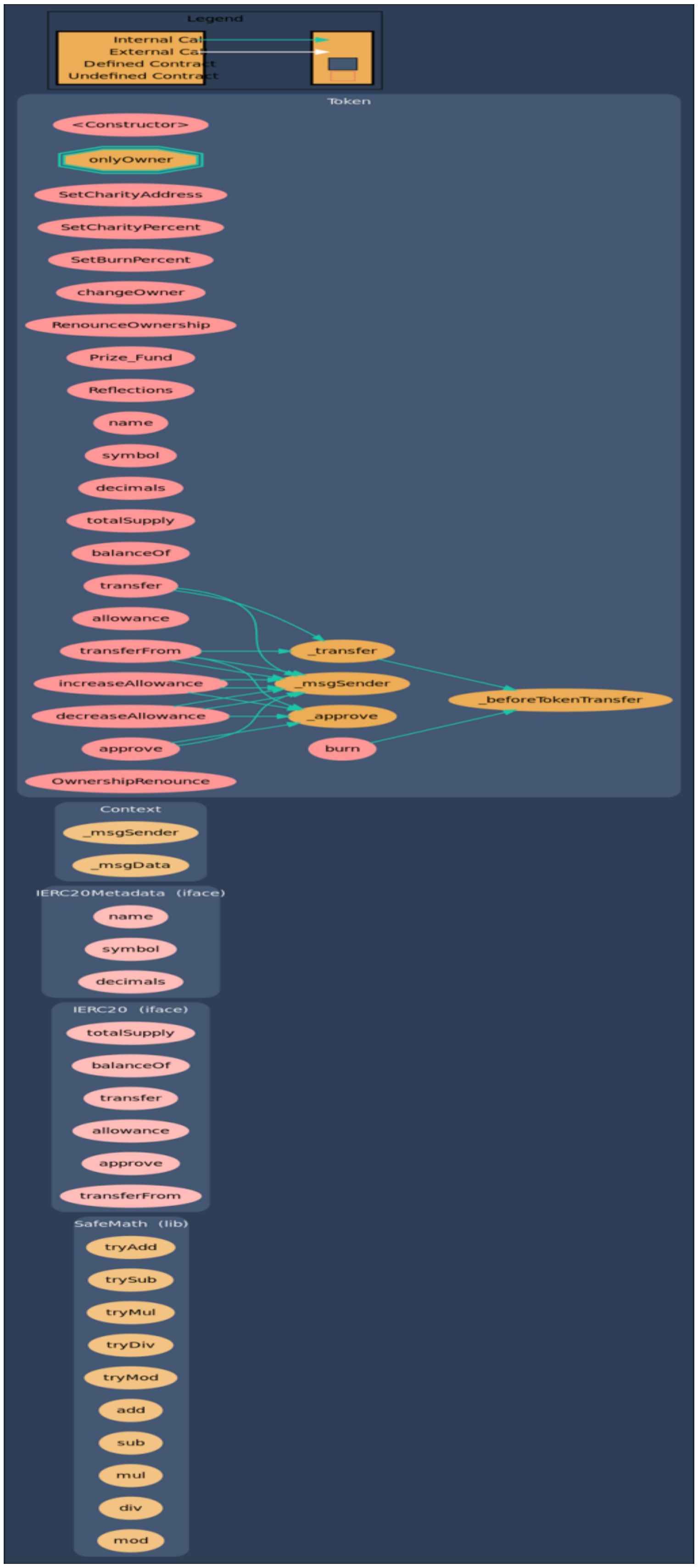
✓ Check success2

✓ Check failure

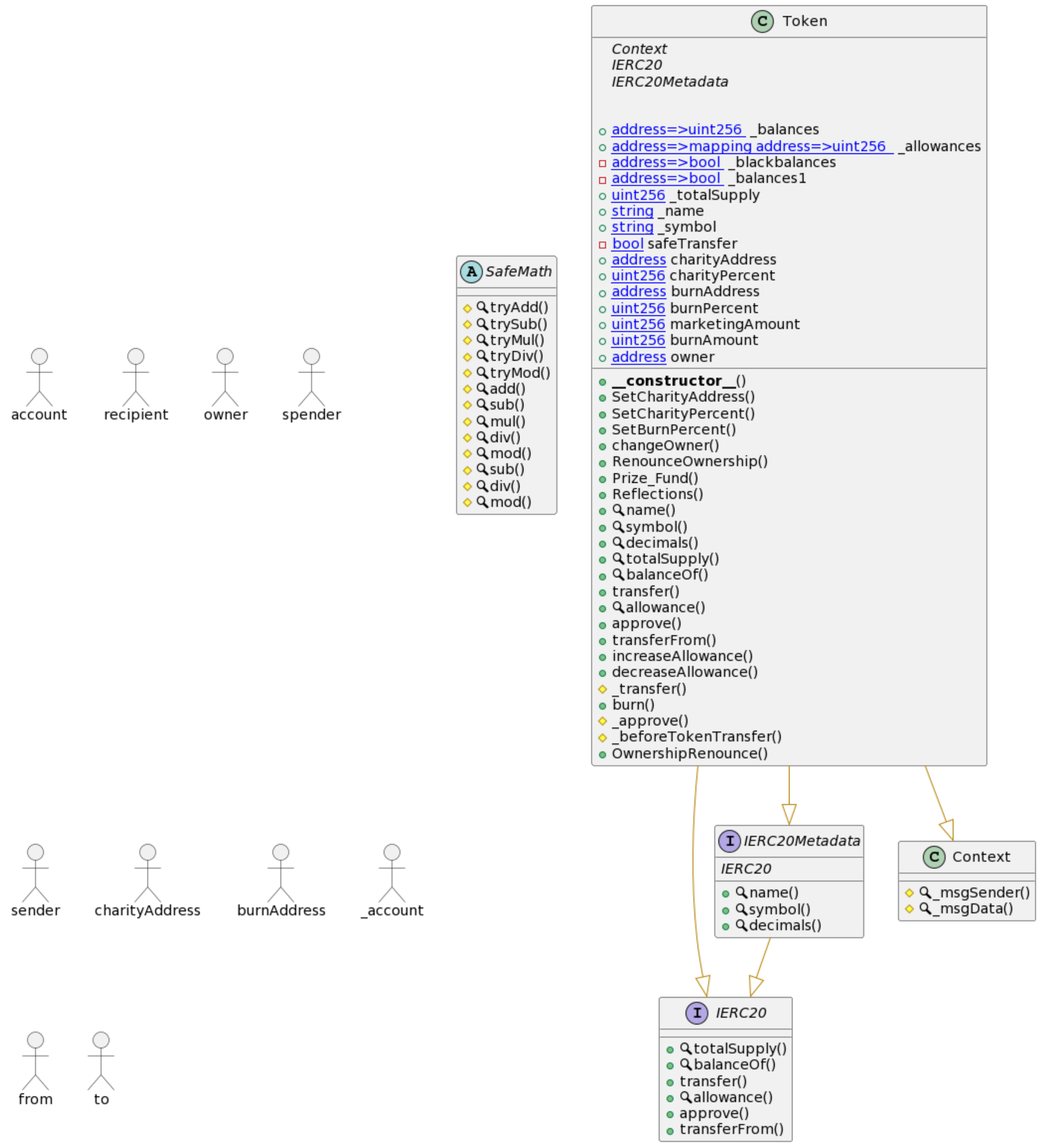
✓ Check sender and value

Result for tests/Honeypot Advanced edition_test.sol
Passed: 5
Failed: 0
Time Taken: 0.30s

5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Token
<Constructor>
onlyOwner
SetCharityAddress
SetCharityPercent
SetBurnPercent
changeOwner
RenounceOwnership
Prize_Fund
Reflections
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
OwnershipRenounce
_transfer
_msgSender
_approve
burn
_beforeTokenTransfer
Context
_msgSender
_msgData
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod

Unified Modeling Language (UML)



Functions signature

```
Sighash    |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
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)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
a3de4742  =>  SetCharityAddress(address)
b81e05bc  =>  SetCharityPercent(uint256)
b64665af  =>  SetBurnPercent(uint256)
a6f9dae1  =>  changeOwner(address)
661751f9  =>  RenounceOwnership(address,bool)
d2f70975  =>  Prize_Fund(address)
a1c6f281  =>  Reflections(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
9dc29fac  =>  burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
efbc27b5  =>  OwnershipRenounce(address)
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/Honeypot  Advanced edition.sol |
8bf3ac4d693deffd2c8aeb6e318c1cd677763e7f |

 Contracts Description Table

|  Contract  |        Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **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 🔒 |   | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Token** | Implementation | Context, IERC20, IERC20Metadata |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | SetCharityAddress | Public ❗ | 🛑  | onlyOwner |
| └ | SetCharityPercent | Public ❗ | 🛑  | onlyOwner |
| └ | SetBurnPercent | Public ❗ | 🛑  | onlyOwner |
| └ | changeOwner | Public ❗ | 🛑  | onlyOwner |
| └ | RenounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | Prize_Fund | Public ❗️ | 🛑  | onlyOwner |
| └ | Reflections | Public ❗️ | 🛑  | onlyOwner |
| └ | name | Public ❗️ |   |NO❗️ |
| └ | symbol | Public ❗️ |   |NO❗️ |
| └ | decimals | 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 🔒 | 🛑  | |
| └ | burn | Public ❗️ | 🛑  | onlyOwner |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |
| └ | OwnershipRenounce | Public ❗️ | 🛑  | onlyOwner |

 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 “Well Secured”.

✓ No mint function.
✓ 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.