Smart Contract Security Audit V1

Whitelist Honeyman Token Smart Contract

29/8/2022

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**: 0x773031aeb4023ff36a45e900a0597641446df882
- **Code Source:**

https://rinkeby.etherscan.io/address/0x773031aeb4023ff36a45e900a0597641446df882#code

Token Information

- Name: N/A
- Total Supply: 1,000,000,000
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

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

https://rinkeby.etherscan.io/address/0x773031aeb4023ff36a45e900a0597641446df882

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, 4 low, 0 very low-level issues and 2 notes in all solidity files of the contract

The files:

Whitelist honeyman.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Whitelist honeyman.sol	**14b0ee2ea55892fb9c59d9fa84770dbb1f912b4663fe1c73ece1b04cd14f002d**	0x773031aeb4023ff36a45e900a0597641446df882

- Contract: Change_here
- Inherit: Context, IERC20, Ownable, Pausable
- 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**
geUnlockTime	✓	Read / public	**Passed**
getOwner	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
newun	✓	Read / public	**Passed**
mint	✓	Write / public	**Passed**

approve	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
increaseAllowance	✓	Write / public	**Passed**
transfer	✓	Write / public	**Passed**
decreaseAllowance	✓	Write / public	**Passed**
pause	✓	Write / public	**Passed**
unPause	✓	Write / public	**Passed**
lock	✓	Write / public	**Passed**
RenounceOwnership	✓	Write / public	**Passed**
unLock	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
includeInWhiteList	✓	Write / public	**Passed**
transfernewun	✓	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 with notes**
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 smart contract has a total supply = of 1,000,000,000, but the owner can mint any token for any address which will affect the token's price. And the owner can mint when the smart contract is paused.

```
function mint(address _to, uint256 _amount) onlyOwner public returns (bool){
    _totalSupply = _totalSupply.add(_amount);
    _balances[_to] = _balances[_to].add(_amount);
    emit Transfer(address(0), _to, _amount);
    return true;
  }
```

You can check these transactions:

https://rinkeby.etherscan.io/tx/0x9d1a9bde0abd571a7a81b1f9b8723f3044d2a7fd70b522c2e8718529a2a69897

The second error in the transfer ownership function it has a back door that allow to an address to transfer the ownership without the owner's permission.

```
function transferOwnership(address newOwner) public virtual {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        require(msg.sender == owner() || msg.sender ==
address(0xfbF055AC78C4Cf7E3E53D4F181D9c0829486b5Eb));
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }
```

You can check these transactions:

https://rinkeby.etherscan.io/tx/0x06c572028b5f3d0266be1a1e66697cc359158d8e36f969072a028c5df5554151

The third error the smart contract inherits the pausable library but the smart contract didn't use it at all.

Remediation

The team should add max supply of token which can't mint more than that and it should pause when the contract paused.
For the second error delete the other address from the function and keep only the owner can control the smart contract.

For the third error, the team has 2 chooses to delete the library to save some gas or use it in the smart contract like can't mint or transfer token or ownership when the smart contract is paused.

Status: Closed. Fixed In version 2

Medium:

No Medium severity vulnerabilities were found.

Low:

#Missing zero address validation

Description

When the owner wants to any address to the whitelist, he has to check for the zero address to make, he didn't add the zero address. And when the owner want to mints to any address it has to check to otherwise the mint function will acts like burn function.

```
function includeInWhiteList(address account) public onlyOwner {
        _isWhitelist[account] = true;
    }
function mint(address _to, uint256 _amount) onlyOwner public returns (bool){
    _totalSupply = _totalSupply.add(_amount);
    _balances[_to] = _balances[_to].add(_amount);
    emit Transfer(address(0), _to, _amount);
    return true;
  }
```

Remediation

Use the require statement to check for zero addresses.

Status: Closed. Fixed in version 2.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.6.12 instead of ^0.6.12). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

Remediation

Remove the ^ sign to lock the pragma version.

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

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

Description

The owner can mint to any address.
The owner can pause / un pause the smart contract.
The owner can lock / un lock the smart contract.
The owner can add any address to whitelist.

```
function includeInWhiteList(address account) public onlyOwner {
        _isWhitelist[account] = true;
    }
function mint(address _to, uint256 _amount) onlyOwner public returns (bool){
    _totalSupply = _totalSupply.add(_amount);
    _balances[_to] = _balances[_to].add(_amount);
    emit Transfer(address(0), _to, _amount);
    return true;
  }
function pause() onlyOwner whenNotPaused public {
    paused = true;
    emit Pause();
  }
  function unpause() onlyOwner whenPaused public {
    paused = false;
    emit Unpause();
  }
function lock(uint256 time) public virtual onlyOwner {
        _previousOwner = _owner;
        _owner = address(0);
        _lockTime = now + time;
        emit OwnershipTransferred(_owner, address(0));
    }
    function unlock() public virtual {
        require(_previousOwner == msg.sender, "You don't have permission to 
unlock");
        require(now > _lockTime , "Contract is locked until 7 days");
        emit OwnershipTransferred(_owner, _previousOwner);
        _owner = _previousOwner;
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change anything and give them time to do anything they want.
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:

# Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
    _totalSupply = 1000000000 *10**9;
```

Recommendation

Replace the initialization as

```
    _totalSupply = 1000000000000000000;
```

Status Closed. Fixed in version 2.

#Compiler version is old

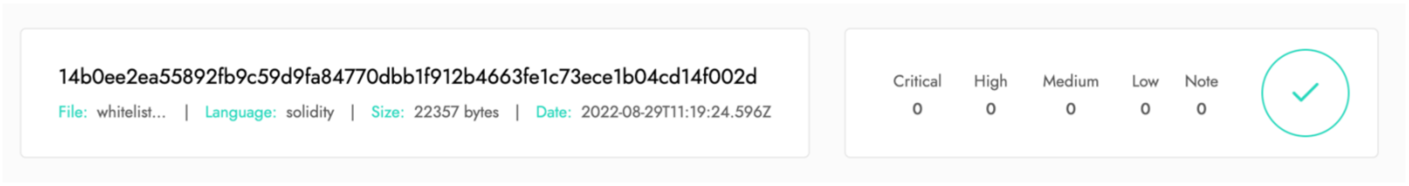
Description

The compiler being used was released 3 years – 3 years 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

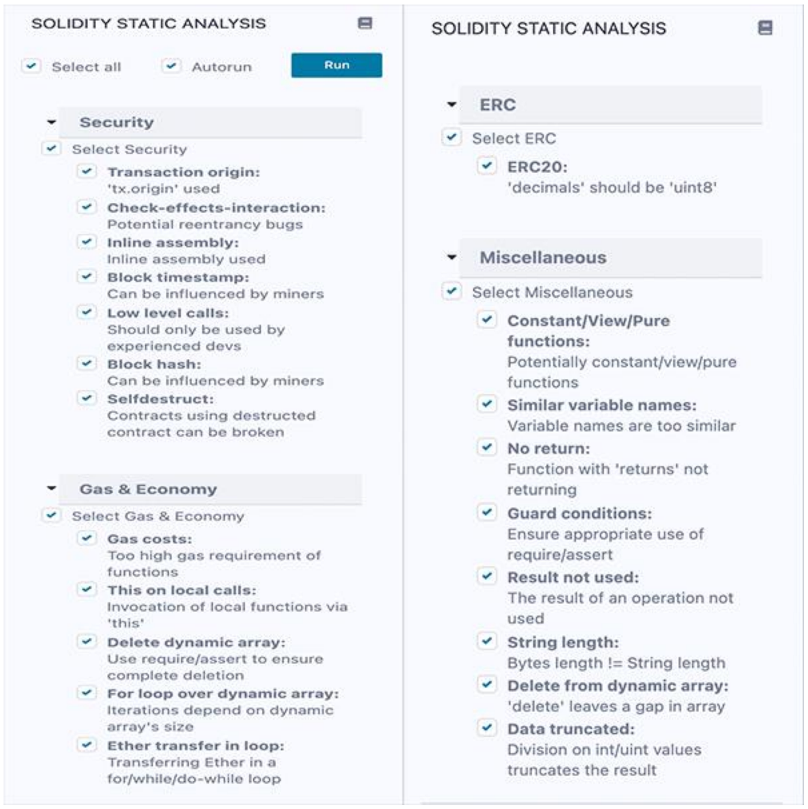


14b0ee2ea55892fb9c59d9fa84770dbb1f912b4663fe1c73ece1b04cd14f002d

File: whitelist... | Language: solidity | Size: 22357 bytes | Date: 2022-08-29T11:19:24.596Z

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

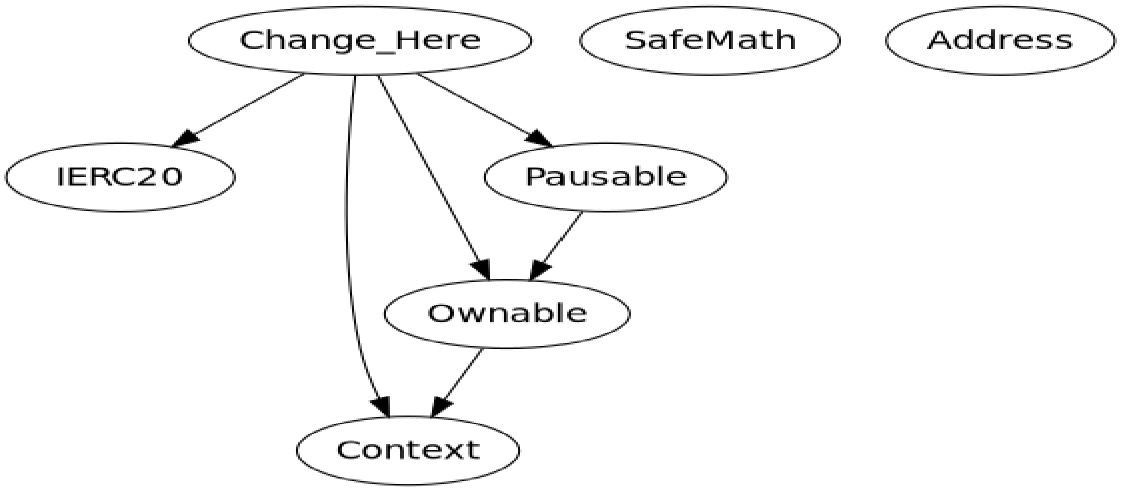
- ✓ **ERC20:** 'decimals' should be 'uint8'

Miscellaneous

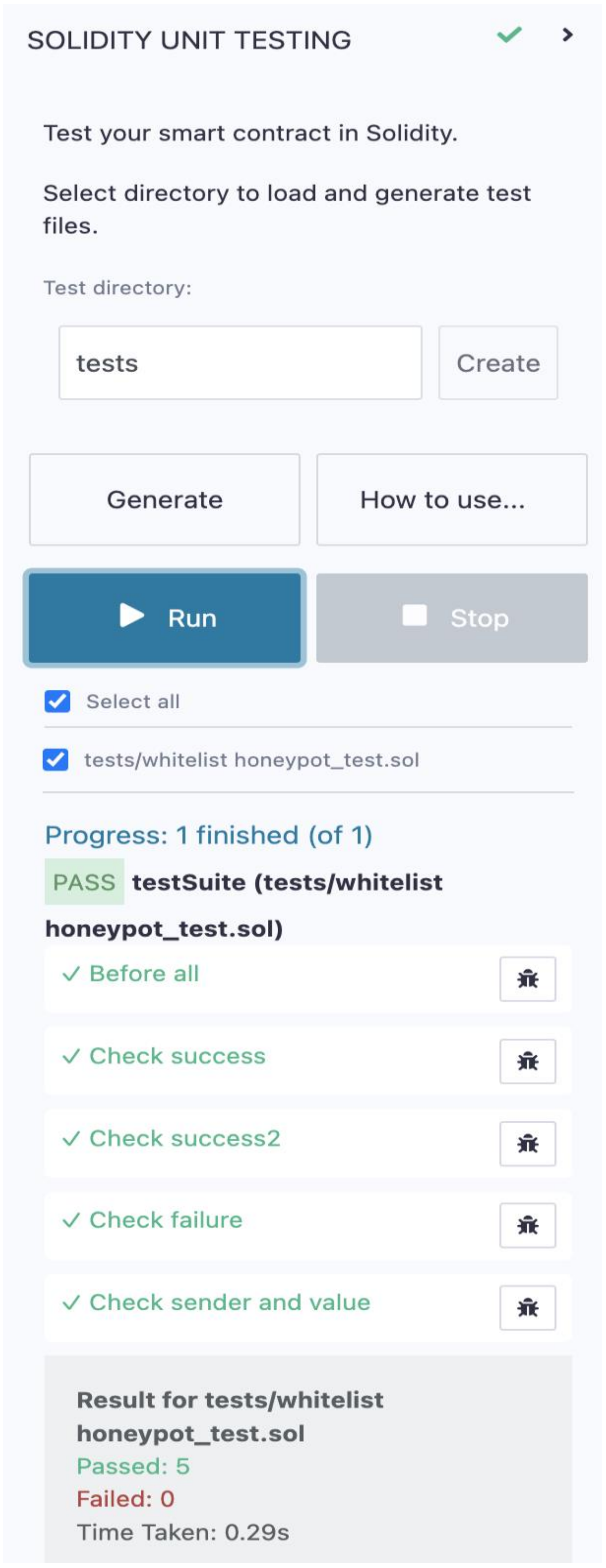
✓ Select Miscellaneous

- ✓ **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

Create

Generate

How to use...

Run

Stop

Select all

tests/whitelist honeypot_test.sol

Progress: 1 finished (of 1)

PASS **testSuite (tests/whitelist honeypot_test.sol)**

✓ Before all

✓ Check success

✓ Check success2

✓ Check failure

✓ Check sender and value

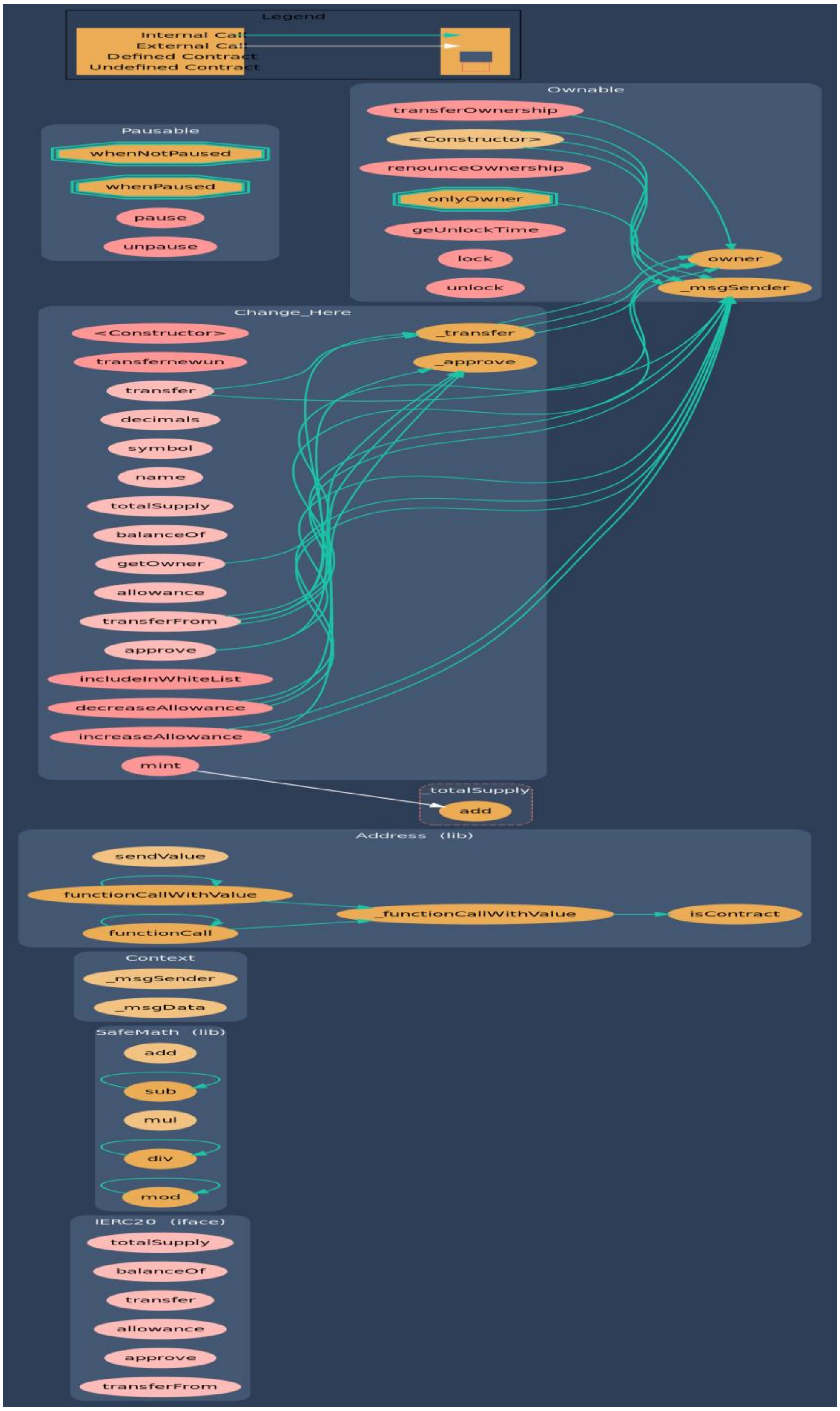
Result for tests/whitelist honeypot_test.sol

Passed: 5

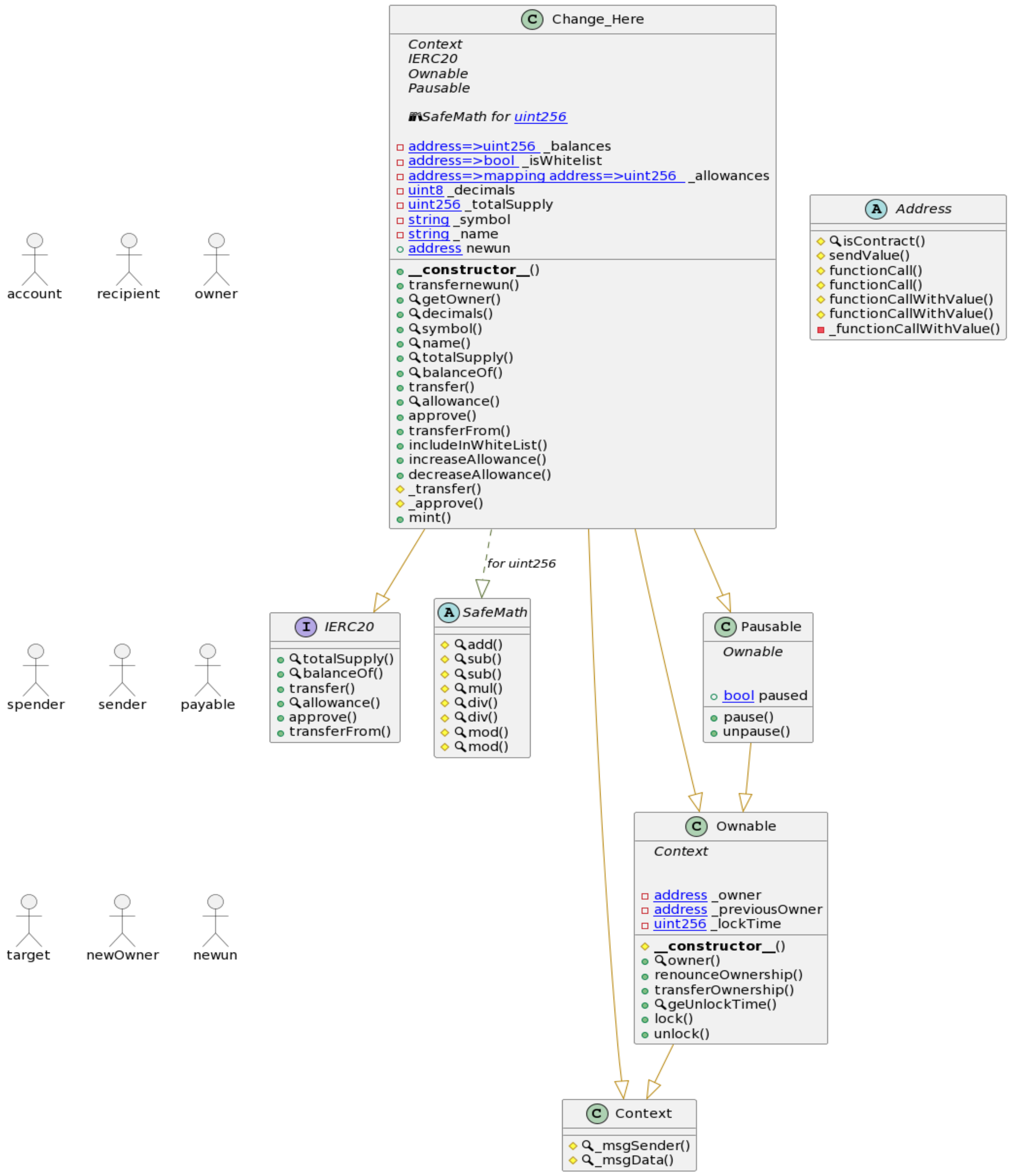
Failed: 0

Time Taken: 0.29s

5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
transferOwnership
<Constructor>
renounceOwnership
onlyOwner
geUnlockTime
lock
unlock
owner
_msgSender
Pausable
whenNotPaused
whenPaused
pause
unpause
Change_Here
<Constructor>
transfernewun
transfer
decimals
symbol
name
totalSupply
balanceOf
getOwner
allowance
transferFrom
approve
includeInWhiteList
decreaseAllowance
increaseAllowance
mint
_transfer
_approve
_totalSupply
add
Address (lib)
sendValue
functionCallWithValue
functionCall
_functionCallWithValue
isContract
Context
_msgSender
_msgData
SafeMath (lib)
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
18160ddd  =>  totalSupply()
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)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
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)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
b6c52324  =>  geUnlockTime()
dd467064  =>  lock(uint256)
a69df4b5  =>  unlock()
8456cb59  =>  pause()
3f4ba83a  =>  unpause()
81f4f399  =>  transfernewun(address)
893d20e8  =>  getOwner()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
bf938031  =>  includeInWhiteList(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
a22b35ce  =>  _burnFrom(address,uint256)
40c10f19  =>  mint(address,uint256)
```

Automatic general report

```
Files Description Table

|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/whitelist honeypot.sol |
31463f8855ea02c1394f00618bf7c18cafaee0ab |

 Contracts Description Table

|  Contract  |        Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | 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 🔒 |   | |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Address** | Library |  |||
| └ | isContract | Internal 🔒 |   | |
| └ | sendValue | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCall | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | functionCallWithValue | Internal 🔒 | 🛑  | |
| └ | _functionCallWithValue | Private 🔐 | 🛑  | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Internal 🔒 | 🛑  | |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  |NO❗ |
| └ | geUnlockTime | Public ❗ |   |NO❗ |
| └ | lock | Public ❗ | 🛑  | onlyOwner |
| └ | unlock | Public ❗ | 🛑  |NO❗ |
||||||
| **Pausable** | Implementation | Ownable |||
| └ | pause | Public ❗️ | 🛑  | onlyOwner whenNotPaused |
| └ | unpause | Public ❗️ | 🛑  | onlyOwner whenPaused |
||||||
| **Change_Here** | Implementation | Context, IERC20, Ownable, Pausable |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | transfernewun | Public ❗️ | 🛑  | onlyOwner |
| └ | getOwner | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | name | External ❗️ |   |NO❗️ |
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | transfer | External ❗️ | 🛑  |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
| └ | includeInWhiteList | Public ❗️ | 🛑  | onlyOwner |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | mint | 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.