



Smart contracts security assessment

Final report

[Tariff: Standard](#)

Hanzo

June 2022



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	3
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	5
7. Conclusion	9
8. Disclaimer	10
9. Static code analysis result	11

Introduction

The report has been prepared for Hanzo. The audited project is a reflect ERC20 token.

Name	Hanzo
Audit date	2022-06-29 - 2022-06-29
Language	Solidity
Platform	Polygon Network

Contracts checked

Name	Address
Hanzo	0x37eB60F78e06c4BB2A5F836B0Fc6BCcBbaA995b3

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	failed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed
<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed

<u>Floating Pragma</u>	failed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

❏ Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

❏ Issues

High severity issues

1. Fees problems (Hanzo)

If the fees are changed with `setFee()` the total amount can be greater or equal to 100%. This can break the math and halt all token transfers.

Recommendation: The ownership can be transferred to the Timelock contract with a minimum delay of 48 hours. The contract should also validate that the total fee amount is less than 100%. The governance contract can not include this set function to not bother users at all.

2. Excessive owner's rights (Hanzo)

- a. The owner can set `_maxTxAmount` to zero, which would disable all token transfers;
- b. The owner can assign a bot status to any address, which would restrict all transfers from or to this address;
- c. The owner can set `_maxWalletSize` to zero, which would disable all token transfers.

Recommendation: The ownership can be transferred to the Timelock contract with a minimum delay of 48 hours. The contract should also assert max transaction amount and max wallet size aren't changed harshly and don't fall below a reasonable limit. The governance contract can not include these set functions to not bother users at all.

Medium severity issues

1. Broken presale logic (Hanzo)

In L362 it's required that a sender has `preTrader` status, but in L358 the if-statement checks that it doesn't have one. The requirements are contradictory and can not be satisfied simultaneously.

```
if (from != owner() && to != owner() && !preTrader[from] && !preTrader[to]) {  
  
    //Trade start check  
    if (!tradingOpen) {  
        require(preTrader[from], "TOKEN: This account cannot send tokens until trading  
is enabled");  
    }  
  
    ...  
}
```

2. Lack of `excludeFromReward` logic (Hanzo)

The original SafeMoon project supports the functionality for excluding addresses from receiving holder rewards. It was designed with the intent to remove addresses with exchange liquidity, so after

reflection, holders increase their share in these liquidity assets. For this financial model to work properly, swap pairs and whale holders must not get rewards. But be aware of problems with initial exclude reward logic architecture, you can read more about it [here](#).

Low severity issues

1. Gas optimization (Hanzo)

- a. `uniswapV2Router`, `uniswapV2Pair` should be marked as immutable;
- b. All public functions' visibility can be changed to external;
- c. MATIC for marketing fees distribution and liquifying can be obtained in one swap operation.

2. Swaps without slippage (Hanzo)

Swaps are performed with 0 slippage parameters. This means that the actual swaps will be done with 100% slippage and may be frontrun. The issue may have a significant impact on big token swaps.

3. Undefined beneficiary (Hanzo)

In the `addLiquidityETH()` receiver of LP tokens is the null address. The generated income doesn't belong to anyone. This disables liquidity migrations possibilities to other DEXs and results in scale options being cut off.

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    address(0),
    block.timestamp
);
```

4. No events (Hanzo)

No events are emitted in `blockBots()`, `unblockBot()`, `setTrading()`, `blockBots()`, `unblockBot()`, `setTaxAddresses()`, `excludeMultipleAccountsFromFees()`, `setMinSwapTokensThreshold()`, `toggleSwap()`, `setMaxTxnAmount()`, `setMaxWalletSize()`, `allowPreTrading()`.

Conclusion

Hanzo Hanzo contract was audited. 2 high, 2 medium, 4 low severity issues were found.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Static code analysis result

Hanzo.sendETHToFee(uint256) (contracts/Token.sol#483-489) sends eth to arbitrary user

Dangerous calls:

- _developmentAddress.transfer(developmentShare) (contracts/Token.sol#487)
- _marketingAddress.transfer(marketingShare) (contracts/Token.sol#488)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations>

Reentrancy in Hanzo._transfer(address,address,uint256) (contracts/Token.sol#351-425):

External calls:

- swapTokensForEth(marketingTokens) (contracts/Token.sol#391)
 - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#436-442)
- swapAndLiquify(liquidityTokens) (contracts/Token.sol#399)
 - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)
 - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#436-442)

External calls sending eth:

- sendETHToFee(contractETHBalance) (contracts/Token.sol#395)
 - _developmentAddress.transfer(developmentShare) (contracts/Token.sol#487)

- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)
- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
(`address(this),tokenAmount,0,0,address(0),block.timestamp`) (contracts/Token.sol#473-480)

State variables written after the call(s):

- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
- `_previousredisFee = _redisFee` (contracts/Token.sol#328)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
- `_previoustaxFee = _taxFee` (contracts/Token.sol#329)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
- `_rOwned[address(this)] = _rOwned[address(this)].add(rTeam)` (contracts/Token.sol#551)
- `_rOwned[sender] = _rOwned[sender].sub(rAmount)` (contracts/Token.sol#541)
- `_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)` (contracts/Token.sol#542)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
- `_rTotal = _rTotal.sub(rFee)` (contracts/Token.sol#555)
- `_redisFee = _redisFeeOnBuy` (contracts/Token.sol#412)
- `_redisFee = _redisFeeOnSell` (contracts/Token.sol#418)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
- `_redisFee = _previousredisFee` (contracts/Token.sol#336)

- _redisFee = 0 (contracts/Token.sol#331)
- _tokenTransfer(from,to,amount,takeFee) (contracts/Token.sol#424)
- _tFeeTotal = _tFeeTotal.add(tFee) (contracts/Token.sol#556)
- _taxFee = _totalTaxFeeOnBuy (contracts/Token.sol#413)
- _taxFee = _totalTaxFeeOnSell (contracts/Token.sol#419)
- _tokenTransfer(from,to,amount,takeFee) (contracts/Token.sol#424)
- _taxFee = _previoustaxFee (contracts/Token.sol#337)
- _taxFee = 0 (contracts/Token.sol#332)
- swapAndLiquify(liquidityTokens) (contracts/Token.sol#399)
- inSwap = true (contracts/Token.sol#221)
- inSwap = false (contracts/Token.sol#223)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities>

Hanzo.addLiquidity(uint256,uint256) (contracts/Token.sol#467-481) ignores return value by
 uniswapV2Router.addLiquidityETH{value: ethAmount}
 (address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

Hanzo.allowance(address,address).owner (contracts/Token.sol#277) shadows:

- Ownable.owner() (contracts/Token.sol#61-63) (function)

Hanzo._approve(address,address,uint256).owner (contracts/Token.sol#341) shadows:

- Ownable.owner() (contracts/Token.sol#61-63) (function)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing>

Hanzo.setFee(uint256,uint256,uint256,uint256,uint256,uint256,uint256) (contracts/Token.sol#638-652) should emit an event for:

- _redisFeeOnBuy = redisFeeOnBuy (contracts/Token.sol#639)
- _totalTaxFeeOnBuy = totalTaxFeeOnBuy (contracts/Token.sol#640)
- _redisFeeOnSell = redisFeeOnSell (contracts/Token.sol#642)
- _marketingFee = marketingFee (contracts/Token.sol#644)
- _developmentFee = developmentFee (contracts/Token.sol#645)
- _liquidityFee = liquidityFee (contracts/Token.sol#646)
- _burnFee = burnFee (contracts/Token.sol#647)

- totalFee =

_redisFeeOnSell.add(_marketingFee).add(_developmentFee).add(_liquidityFee).add(_burnFee) (contracts/Token.sol#649)

- _totalTaxFeeOnSell =

_marketingFee.add(_developmentFee).add(_liquidityFee).add(_burnFee) (contracts/Token.sol#650)

Hanzo.setMinSwapTokensThreshold(uint256) (contracts/Token.sol#666-668) should emit an event for:

- _swapTokensAtAmount = swapTokensAtAmount (contracts/Token.sol#667)

Hanzo.setMaxTxnAmount(uint256) (contracts/Token.sol#676-678) should emit an event for:

- _maxTxAmount = maxTxAmount (contracts/Token.sol#677)

Hanzo.setMaxWalletSize(uint256) (contracts/Token.sol#680-682) should emit an event for:

- _maxWalletSize = maxWalletSize (contracts/Token.sol#681)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic>

Ownable.constructor().msgSender (contracts/Token.sol#56) lacks a zero-check on :

- _owner = msgSender (contracts/Token.sol#57)

Hanzo.constructor(address,address).marketingAddress (contracts/Token.sol#226) lacks a zero-check on :

- _marketingAddress = marketingAddress (contracts/Token.sol#228)

Hanzo.constructor(address,address).developmentAddress (contracts/Token.sol#226) lacks a zero-check on :

- _developmentAddress = developmentAddress (contracts/Token.sol#229)

Hanzo.setTaxAddresses(address,address).marketingAddress (contracts/Token.sol#654) lacks a zero-check on :

- _marketingAddress = marketingAddress (contracts/Token.sol#655)

Hanzo.setTaxAddresses(address,address).developmentAddress (contracts/Token.sol#654) lacks a zero-check on :

- _developmentAddress = developmentAddress (contracts/Token.sol#656)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

Reentrancy in Hanzo._transfer(address,address,uint256) (contracts/Token.sol#351-425):

External calls:

- swapTokensForEth(marketingTokens) (contracts/Token.sol#391)
 - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#436-442)
- swapAndLiquify(liquidityTokens) (contracts/Token.sol#399)
 - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)
 - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#436-442)

External calls sending eth:

- sendETHToFee(contractETHBalance) (contracts/Token.sol#395)
 - _developmentAddress.transfer(developmentShare) (contracts/Token.sol#487)
 - _marketingAddress.transfer(marketingShare) (contracts/Token.sol#488)
- swapAndLiquify(liquidityTokens) (contracts/Token.sol#399)
 - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)

State variables written after the call(s):

- swapAndLiquify(liquidityTokens) (contracts/Token.sol#399)
 - _allowances[owner][spender] = amount (contracts/Token.sol#347)

Reentrancy in Hanzo.swapAndLiquify(uint256) (contracts/Token.sol#445-465):

External calls:

- swapTokensForEth(half) (contracts/Token.sol#458)
 - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#436-442)
- addLiquidity(otherHalf,newBalance) (contracts/Token.sol#464)
 - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)

External calls sending eth:

- addLiquidity(otherHalf,newBalance) (contracts/Token.sol#464)
 - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)

State variables written after the call(s):

- addLiquidity(otherHalf,newBalance) (contracts/Token.sol#464)
 - _allowances[owner][spender] = amount (contracts/Token.sol#347)

Reentrancy in Hanzo.transferFrom(address,address,uint256) (contracts/Token.sol#295-310):

External calls:

- _transfer(sender,recipient,amount) (contracts/Token.sol#300)
 - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)
 - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,

nt,0,path,address(this),block.timestamp) (contracts/Token.sol#436-442)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (contracts/Token.sol#300)

- `uniswapV2Router.addLiquidityETH{value: ethAmount}`

(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)

- `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)

- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)

State variables written after the call(s):

- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))` (contracts/Token.sol#301-308)

- `_allowances[owner][spender] = amount` (contracts/Token.sol#347)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2>

Reentrancy in `Hanzo._transfer(address,address,uint256)` (contracts/Token.sol#351-425):

External calls:

- `swapTokensForEth(marketingTokens)` (contracts/Token.sol#391)

- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (contracts/Token.sol#436-442)

- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)

- `uniswapV2Router.addLiquidityETH{value: ethAmount}`

(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/Token.sol#473-480)

- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (contracts/Token.sol#436-442)

External calls sending eth:

- `sendETHToFee(contractETHBalance)` (contracts/Token.sol#395)
- `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)
- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)
- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
(`address(this),tokenAmount,0,0,address(0),block.timestamp`) (contracts/Token.sol#473-480)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (contracts/Token.sol#348)
- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
- `Transfer(sender,recipient,tTransferAmount)` (contracts/Token.sol#545)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)

Reentrancy in `Hanzo.swapAndLiquify(uint256)` (contracts/Token.sol#445-465):

External calls:

- `swapTokensForEth(half)` (contracts/Token.sol#458)
- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (contracts/Token.sol#436-442)
- `addLiquidity(otherHalf,newBalance)` (contracts/Token.sol#464)

- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
`(address(this),tokenAmount,0,0,address(0),block.timestamp)` (contracts/Token.sol#473-480)

External calls sending eth:

- `addLiquidity(otherHalf,newBalance)` (contracts/Token.sol#464)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
`(address(this),tokenAmount,0,0,address(0),block.timestamp)` (contracts/Token.sol#473-480)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (contracts/Token.sol#348)
- `addLiquidity(otherHalf,newBalance)` (contracts/Token.sol#464)

Reentrancy in `Hanzo.transferFrom(address,address,uint256)` (contracts/Token.sol#295-310):

External calls:

- `_transfer(sender,recipient,amount)` (contracts/Token.sol#300)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
`(address(this),tokenAmount,0,0,address(0),block.timestamp)` (contracts/Token.sol#473-480)
- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (contracts/Token.sol#436-442)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (contracts/Token.sol#300)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
`(address(this),tokenAmount,0,0,address(0),block.timestamp)` (contracts/Token.sol#473-480)

- `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)
- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (contracts/Token.sol#348)
- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20:transfer amount exceeds allowance))` (contracts/Token.sol#301-308)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3>

Hanzo.totalFee (contracts/Token.sol#191) is set pre-construction with a non-constant function or state variable:

```
-
_redisFeeOnSell.add(_marketingFee).add(_developmentFee).add(_liquidityFee).add(_burnFee)
```

Hanzo._totalTaxFeeOnSell (contracts/Token.sol#192) is set pre-construction with a non-constant function or state variable:

```
- _marketingFee.add(_developmentFee).add(_liquidityFee).add(_burnFee)
```

Hanzo._redisFee (contracts/Token.sol#195) is set pre-construction with a non-constant function or state variable:

```
- _redisFeeOnSell
```

Hanzo._taxFee (contracts/Token.sol#196) is set pre-construction with a non-constant function or state variable:

```
- _totalTaxFeeOnSell
```

Hanzo._previousredisFee (contracts/Token.sol#198) is set pre-construction with a non-constant

function or state variable:

- _redisFee

Hanzo._previoustaxFee (contracts/Token.sol#199) is set pre-construction with a non-constant function or state variable:

- _taxFee

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state>

Function IUniswapV2Router02.WETH() (contracts/Token.sol#144) is not in mixedCase

Parameter Hanzo.setTrading(bool)._tradingOpen (contracts/Token.sol#491) is not in mixedCase

Parameter Hanzo.toggleSwap(bool)._swapEnabled (contracts/Token.sol#671) is not in mixedCase

Constant Hanzo._name (contracts/Token.sol#167) is not in UPPER_CASE_WITH_UNDERSCORES

Constant Hanzo._symbol (contracts/Token.sol#168) is not in UPPER_CASE_WITH_UNDERSCORES

Constant Hanzo._decimals (contracts/Token.sol#169) is not in
UPPER_CASE_WITH_UNDERSCORES

Constant Hanzo._tTotal (contracts/Token.sol#175) is not in UPPER_CASE_WITH_UNDERSCORES

Variable Hanzo._marketingAddress (contracts/Token.sol#204) is not in mixedCase

Variable Hanzo._developmentAddress (contracts/Token.sol#205) is not in mixedCase

Constant Hanzo._burnAddress (contracts/Token.sol#206) is not in
UPPER_CASE_WITH_UNDERSCORES

Variable Hanzo._maxTxAmount (contracts/Token.sol#215) is not in mixedCase

Variable Hanzo._maxWalletSize (contracts/Token.sol#216) is not in mixedCase

Variable `Hanzo._swapTokensAtAmount` (contracts/Token.sol#217) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Reentrancy in `Hanzo._transfer(address,address,uint256)` (contracts/Token.sol#351-425):

External calls:

- `sendETHToFee(contractETHBalance)` (contracts/Token.sol#395)
- `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)
- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)

External calls sending eth:

- `sendETHToFee(contractETHBalance)` (contracts/Token.sol#395)
- `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)
- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)
- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
(`address(this),tokenAmount,0,0,address(0),block.timestamp`) (contracts/Token.sol#473-480)

State variables written after the call(s):

- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
- `_allowances[owner][spender] = amount` (contracts/Token.sol#347)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)

- `_previousredisFee = _redisFee` (contracts/Token.sol#328)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
 - `_previoustaxFee = _taxFee` (contracts/Token.sol#329)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
 - `_rOwned[address(this)] = _rOwned[address(this)].add(rTeam)` (contracts/Token.sol#551)
 - `_rOwned[sender] = _rOwned[sender].sub(rAmount)` (contracts/Token.sol#541)
 - `_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)` (contracts/Token.sol#542)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
 - `_rTotal = _rTotal.sub(rFee)` (contracts/Token.sol#555)
- `_redisFee = _redisFeeOnBuy` (contracts/Token.sol#412)
- `_redisFee = _redisFeeOnSell` (contracts/Token.sol#418)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
 - `_redisFee = _previousredisFee` (contracts/Token.sol#336)
 - `_redisFee = 0` (contracts/Token.sol#331)
- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
 - `_tFeeTotal = _tFeeTotal.add(tFee)` (contracts/Token.sol#556)
- `_taxFee = _totalTaxFeeOnBuy` (contracts/Token.sol#413)
- `_taxFee = _totalTaxFeeOnSell` (contracts/Token.sol#419)

- `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)
 - `_taxFee = _previoustaxFee` (contracts/Token.sol#337)
 - `_taxFee = 0` (contracts/Token.sol#332)
- `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
 - `inSwap = true` (contracts/Token.sol#221)
 - `inSwap = false` (contracts/Token.sol#223)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (contracts/Token.sol#348)
 - `swapAndLiquify(liquidityTokens)` (contracts/Token.sol#399)
- `Transfer(sender,recipient,tTransferAmount)` (contracts/Token.sol#545)
 - `_tokenTransfer(from,to,amount,takeFee)` (contracts/Token.sol#424)

Reentrancy in `Hanzo.transferFrom(address,address,uint256)` (contracts/Token.sol#295-310):

External calls:

- `_transfer(sender,recipient,amount)` (contracts/Token.sol#300)
 - `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)
 - `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (contracts/Token.sol#300)

- `uniswapV2Router.addLiquidityETH{value: ethAmount}`
(`address(this),tokenAmount,0,0,address(0),block.timestamp`) (contracts/Token.sol#473-480)

- `_developmentAddress.transfer(developmentShare)` (contracts/Token.sol#487)

- `_marketingAddress.transfer(marketingShare)` (contracts/Token.sol#488)

State variables written after the call(s):

- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20:transfer amount exceeds allowance))` (contracts/Token.sol#301-308)

- `_allowances[owner][spender] = amount` (contracts/Token.sol#347)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (contracts/Token.sol#348)

- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20:transfer amount exceeds allowance))` (contracts/Token.sol#301-308)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4>

Variable `Hanzo._getRValues(uint256,uint256,uint256,uint256).rTransferAmount` (contracts/Token.sol#619) is too similar to `Hanzo._transferStandard(address,address,uint256).tTransferAmount` (contracts/Token.sol#537)

Variable `Hanzo._transferStandard(address,address,uint256).rTransferAmount` (contracts/Token.sol#535) is too similar to `Hanzo._getTValues(uint256,uint256,uint256).tTransferAmount` (contracts/Token.sol#597)

Variable `Hanzo._getRValues(uint256,uint256,uint256,uint256).rTransferAmount` (contracts/Token.sol#619) is too similar to `Hanzo._getValues(uint256).tTransferAmount` (contracts/Token.sol#573)

Variable Hanzo._getValues(uint256).rTransferAmount (contracts/Token.sol#576) is too similar to Hanzo._getTValues(uint256,uint256,uint256).tTransferAmount (contracts/Token.sol#597)

Variable Hanzo._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#535) is too similar to Hanzo._getValues(uint256).tTransferAmount (contracts/Token.sol#573)

Variable Hanzo._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#535) is too similar to Hanzo._transferStandard(address,address,uint256).tTransferAmount (contracts/Token.sol#537)

Variable Hanzo._getValues(uint256).rTransferAmount (contracts/Token.sol#576) is too similar to Hanzo._getValues(uint256).tTransferAmount (contracts/Token.sol#573)

Variable Hanzo._getValues(uint256).rTransferAmount (contracts/Token.sol#576) is too similar to Hanzo._transferStandard(address,address,uint256).tTransferAmount (contracts/Token.sol#537)

Variable Hanzo._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#619) is too similar to Hanzo._getTValues(uint256,uint256,uint256).tTransferAmount (contracts/Token.sol#597)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar>

Hanzo.slitherConstructorConstantVariables() (contracts/Token.sol#163-689) uses literals with too many digits:

- _tTotal = 1000000000000000 * 10 ** 9 (contracts/Token.sol#175)

Hanzo.slitherConstructorConstantVariables() (contracts/Token.sol#163-689) uses literals with too many digits:

- _burnAddress = address(0x00dEaD) (contracts/

Token.sol#206)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits>

Ownable._previousOwner (contracts/Token.sol#49) is never used in Hanzo (contracts/Token.sol#163-689)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable>

Ownable._previousOwner (contracts/Token.sol#49) should be constant

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant>

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (contracts/Token.sol#70-73)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (contracts/Token.sol#75-79)

name() should be declared external:

- Hanzo.name() (contracts/Token.sol#248-250)

symbol() should be declared external:

- Hanzo.symbol() (contracts/Token.sol#252-254)

decimals() should be declared external:

- Hanzo.decimals() (contracts/Token.sol#256-258)

totalSupply() should be declared external:

- Hanzo.totalSupply() (contracts/Token.sol#260-262)

transfer(address,uint256) should be declared external:

- Hanzo.transfer(address,uint256) (contracts/Token.sol#268-275)

allowance(address,address) should be declared external:

- Hanzo.allowance(address,address) (contracts/Token.sol#277-284)

approve(address,uint256) should be declared external:

- Hanzo.approve(address,uint256) (contracts/Token.sol#286-293)

transferFrom(address,address,uint256) should be declared external:

- Hanzo.transferFrom(address,address,uint256) (contracts/Token.sol#295-310)

setTrading(bool) should be declared external:

- Hanzo.setTrading(bool) (contracts/Token.sol#491-493)

blockBots(address[]) should be declared external:

- Hanzo.blockBots(address[]) (contracts/Token.sol#507-511)

unblockBot(address) should be declared external:

- Hanzo.unblockBot(address) (contracts/Token.sol#513-515)

setFee(uint256,uint256,uint256,uint256,uint256,uint256,uint256) should be declared external:

- Hanzo.setFee(uint256,uint256,uint256,uint256,uint256,uint256,uint256) (contracts/Token.sol#638-652)

setTaxAddresses(address,address) should be declared external:

- Hanzo.setTaxAddresses(address,address) (contracts/Token.sol#654-657)

excludeMultipleAccountsFromFees(address[],bool) should be declared external:

- Hanzo.excludeMultipleAccountsFromFees(address[],bool) (contracts/Token.sol#659-663)

setMinSwapTokensThreshold(uint256) should be declared external:

- Hanzo.setMinSwapTokensThreshold(uint256) (contracts/Token.sol#666-668)

toggleSwap(bool) should be declared external:

- Hanzo.toggleSwap(bool) (contracts/Token.sol#671-673)

setMaxTxnAmount(uint256) should be declared external:

- Hanzo.setMaxTxnAmount(uint256) (contracts/Token.sol#676-678)

setMaxWalletSize(uint256) should be declared external:

- Hanzo.setMaxWalletSize(uint256) (contracts/Token.sol#680-682)

allowPreTrading(address,bool) should be declared external:

- Hanzo.allowPreTrading(address,bool) (contracts/Token.sol#684-687)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

/mnt/d/initial-repository analyzed (7 contracts with 78 detectors), 75 result(s) found



 Guard