Background pattern

Description automatically generated with low confidence

TNDY Audit

Tendy, LLC ERC-20 Smart Contract Token Audit 10/09/2021. Version 1.0

Audited by Horizon Globex

2021

Contents

[Disclaimer 2](#_Toc69744972)

[Actors 3](#_Toc69744973)

[Introduction 3](#_Toc69744974)

[Audit Summary 4](#_Toc69744975)

[Overview 4](#_Toc69744976)

[Methodology 4](#_Toc69744977)

[Taxonomy 4](#_Toc69744978)

[Smart Contract 5](#_Toc69744979)

[Actors 5](#_Toc69744980)

[Additional Requirements 5](#_Toc69744981)

[Issues Identified 6](#_Toc69744982)

[Requirements 6](#_Toc69744983)

[Security 6](#_Toc69744984)

[Static 7](#_Toc69744985)

[Appendix 8](#_Toc69744986)

[UML 8](#_Toc69744987)

[Unit Testing 8](#_Toc69744988)

[Slither Output 9](#_Toc69744989)

# Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The content of this audit report is provided “as is”, without representations and warranties of any kind, and Horizon Globex disclaims any liability for damage arising out of, or in connection with, this audit report. Copyright of this report remains with Horizon Globex.

# Actors

* [**Tendy,**](https://cannappscorp.com/)[**LLC**](https://tendynft.com/) **:** Tendy LLC contracted Abbey Technology to create a marketing awareness campaign to promote their domain trading business. Abbey Technology decided to market to the Uniswap demographic using a TNDY ERC-20 token.
* [**Abbey Technology**](https://abbey.ch/)**:** The marketing technology service provider company implementing and deploying the TNDY token.
* [**Horizon Globex**](https://www.horizonfintex.com/): 3rd Party, independent, technology company. Experts in Ethereum smart contract development, requested to perform an external audit of the TNDY token.

# Introduction

Horizon Globex was requested to review the TNDY Token implementation, auditing for code vulnerabilities and bugs. Under consideration is TNDYToken.sol available from the Abbey Technology public GITHub repository at <https://github.com/abbey-ch/TNDY> .

The token uses [Open Zepplin](https://openzeppelin.com/) ERC-20 libraries as part of the implementation. These are not considered within the scope of the code review as they have been extensively reviewed by other providers.

The TNDY smart contract is deployed to Ethereum at this address <https://etherscan.io/address/0xe6525fbdfce39ec03e78c13ef556f7d331318b20#code>

# Audit Summary

|  |  |  |
| --- | --- | --- |
| Type | #Issues | Status |
| Requirements | 3 | Fixed |
| Security | 2 | Fixed |
| Static | 20 | No code changes required. False positive Slither recommendations. |

# Overview

## Methodology

Each of the extended requirements of the smart contract are audited for a corresponding implementation.

Static analysis tooling is applied to the implementation to highlight security risks. The static analysis tools [slither](https://github.com/crytic/slither) and [surya](https://github.com/ConsenSys/surya) are executed on the TNDYToken.sol source code.

## Taxonomy

* **Requirement**: Code prevents requirements being met.
* **Security**: Code creates a security attack vector.
* **Static**: Code has potential for a bug as revealed by static analysis tools.

# Smart Contract

The TNDY token is created with the intention of subsequently listing as a Uniswap pair for secondary trading.

## Actors

* **Liquidity Provider**: Abbey Technology is the token holder who will seed liquidity for the corresponding Uniswap token post-deployment with 16,500 TNDY tokens and 4.18 ETH. The liquidity provider address is constrained by the smart contract such that it may only swap its tokens on Uniswap and may only do so after it has announced its intention to redeem liquidity from the pool at a future date, basically, an early warning system for the Uniswap community.
* **Treasury**: A holding address that is constrained by the smart contract such that it may only swap its tokens on Uniswap and may only do so after it has announced its intention to swap some of its tokens at a future date, basically, an early warning system for the Uniswap community.

## Additional Requirements

In addition to the standard feature set provided by an ERC-20 implementation, the following extra requirements are implemented in the GCAC token. The general purpose of these requirements is to mitigate market manipulation and provide Uniswap community comfort:

1. Do not allow Abbey, the Uniswap liquidity provider, to remove liquidity without advance notice.
2. Do not allow Treasury to transfer tokens to anywhere other than the Uniswap pair.  This is to prevent the treasury from profiting at trading venues other than Uniswap.
3. Treasury can swap her tokens into the pair, but only if X days’ notice is explicitly given to the ERC-20 contract for a future date and amount.  If any of the amount is subsequently transferred after the notice period, then the notice is reset and must be given again for a transfer to occur.

# Issues Identified

Fixed **✓**

No Issues Identified

Fixed **✓**

## Static

No code changes were required because of static analyses. Full report available in the appendix.

# Appendix

## Unit Testing

 C:\Dev\talketh.io\Solidity\TNDY\ERC20>truffle run coverage

> Using Truffle library from local node\_modules.

> server: http://127.0.0.1:8555

> truffle: v5.4.9

> ganache-core: v2.13.0

> solidity-coverage: v0.7.16

Network Info

============

> id: \*

> port: 8555

> network: soliditycoverage

Instrumenting for coverage...

=============================

> DebugLogger.sol

> FakePair.sol

> FakePositions.sol

> TNDYToken.sol

Coverage skipped for:

=====================

> Migrations.sol

Compiling your contracts...

===========================

> Compiling @openzeppelin\contracts\token\ERC20\ERC20.sol

> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol

> Compiling @openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol

> Compiling @openzeppelin\contracts\token\ERC721\ERC721.sol

> Compiling @openzeppelin\contracts\token\ERC721\IERC721.sol

> Compiling @openzeppelin\contracts\token\ERC721\IERC721Receiver.sol

> Compiling @openzeppelin\contracts\token\ERC721\extensions\IERC721Metadata.sol

> Compiling @openzeppelin\contracts\utils\Address.sol

> Compiling @openzeppelin\contracts\utils\Context.sol

> Compiling @openzeppelin\contracts\utils\Strings.sol

> Compiling @openzeppelin\contracts\utils\introspection\ERC165.sol

> Compiling @openzeppelin\contracts\utils\introspection\IERC165.sol

> Compiling .\.coverage\_contracts\DebugLogger.sol

> Compiling .\.coverage\_contracts\FakePair.sol

> Compiling .\.coverage\_contracts\FakePositions.sol

> Compiling .\.coverage\_contracts\Migrations.sol

> Compiling .\.coverage\_contracts\TNDYToken.sol

> Artifacts written to C:\Dev\talketh.io\Solidity\TNDY\ERC20\.coverage\_artifacts\contracts

> Compiled successfully using:

- solc: 0.8.7+commit.e28d00a7.Emscripten.clang

Compiling your contracts...

===========================

> Compiling @openzeppelin\contracts\token\ERC20\ERC20.sol

> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol

> Compiling @openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol

> Compiling @openzeppelin\contracts\token\ERC721\ERC721.sol

> Compiling @openzeppelin\contracts\token\ERC721\IERC721.sol

> Compiling @openzeppelin\contracts\token\ERC721\IERC721Receiver.sol

> Compiling @openzeppelin\contracts\token\ERC721\extensions\IERC721Metadata.sol

> Compiling @openzeppelin\contracts\utils\Address.sol

> Compiling @openzeppelin\contracts\utils\Context.sol

> Compiling @openzeppelin\contracts\utils\Strings.sol

> Compiling @openzeppelin\contracts\utils\introspection\ERC165.sol

> Compiling @openzeppelin\contracts\utils\introspection\IERC165.sol

> Compiling .\.coverage\_contracts\DebugLogger.sol

> Compiling .\.coverage\_contracts\FakePair.sol

> Compiling .\.coverage\_contracts\FakePositions.sol

> Compiling .\.coverage\_contracts\Migrations.sol

> Compiling .\.coverage\_contracts\TNDYToken.sol

> Artifacts written to C:\Dev\talketh.io\Solidity\TNDY\ERC20\.coverage\_artifacts\contracts

> Compiled successfully using:

- solc: 0.8.7+commit.e28d00a7.Emscripten.clang

Owner = 0x66e19FCC6F7a5A1bddBAd6aF858588ccC0e0Eab4

Buyback = 0x26Cd859D033ce05BC0cEEFb33A477479E1Af6b52

Treasury = 0x97C16915B273b1a6FAcdD17C3170cAbB3725888e

Flip = 0x5Bab52bC8B473646002aB036CE9486b6E63937d3

Trader = 0x3737c23A81E9248E21242D8092704784340eDc0a

WethAddress = 0x8DBbB534008689d0105D169ea4F79D4c91cd84D6

Router = 0xE592427A0AEce92De3Edee1F18E0157C05861564

positions = 0x8DBbB534008689d0105D169ea4F79D4c91cd84D6

Network = local

Is Test Network? = true

gasPrice (param) = undefined

gasPrice = 1

owner Eth = 99999999999999708633

buyback Eth = 100000000000000000000

treasury Eth = 100000000000000000000

trader Eth = 100000000000000000000

owner = 0x66e19FCC6F7a5A1bddBAd6aF858588ccC0e0Eab4

buyback = 0x26Cd859D033ce05BC0cEEFb33A477479E1Af6b52

treasury = 0x97C16915B273b1a6FAcdD17C3170cAbB3725888e

flip = 0x5Bab52bC8B473646002aB036CE9486b6E63937d3

trader = 0x3737c23A81E9248E21242D8092704784340eDc0a

TNDY = 0x4956f480eefA9BF6B02e210742EfBd83CEBB6F1c

TNDY.owner() = 0x66e19FCC6F7a5A1bddBAd6aF858588ccC0e0Eab4

Pair = 0x34B6FC60e7aEDC477825C12aCF2C286b05167185

Token.totalSupply() = 3,300,000

Owner Token Balance = 16,500

Treasury Token Balance = 3,267,000

Buyback Token Balance = 0

Flip Token Balance = 16,500

Trader Token Balance = 0

pair.totalSupply() = 16,500

Owner Pair Balance = 16,500

Treasury Pair Balance = 0

Buyback Pair Balance = 0

Flip Pair Balance = 0

Trader Pair Balance = 0

Owner = 0x66e19FCC6F7a5A1bddBAd6aF858588ccC0e0Eab4

Buyback = 0x26Cd859D033ce05BC0cEEFb33A477479E1Af6b52

Treasury = 0x97C16915B273b1a6FAcdD17C3170cAbB3725888e

Flip = 0x5Bab52bC8B473646002aB036CE9486b6E63937d3

Trader = 0x3737c23A81E9248E21242D8092704784340eDc0a

Contract: TNDY

√ Verify deployment (677ms)

√ [Negative] Non-owner can't use setBuyback (1092ms)

√ [Negative] Non-owner can't use setTreasury (178ms)

√ [Negative] Non-owner can't use setFlip (193ms)

√ [Negative] Non-owner can't use setNetAssetValue (161ms)

√ [Negative] Non-owner can't use treasuryTransferNotice (175ms)

√ [Negative] Can't use treasuryTransferNotice on a non-treasury address (210ms)

√ [Negative] Non-owner can't use liquidityRedemptionNotice (251ms)

√ [Negative] Can't use liquidityRedemptionNotice until the NFT Id is set (174ms)

√ [Negative] Non-owner can't use setBinanceContract (176ms)

√ Set buyback as owner (304ms)

√ Set buyback as owner a second time. (333ms)

√ Set treasury as owner (339ms)

√ Set treasury as owner a second time (283ms)

√ Set flip as owner (339ms)

√ Set flip as owner a second time (296ms)

√ Set NAV as owner (320ms)

√ Set NAV as owner a second time (303ms)

√ Set setBinanceContract as owner (273ms)

√ Set setBinanceContract as owner a second time (254ms)

√ [Negative] Attempt to send tokens without notice period from treasury (113ms)

√ Transfer tokens to buyback as owner, Uniswap prevents us from auto burning them. (397ms)

√ [Negative] Attempt to send tokens from the buyback account (158ms)

√ [Negative] Attempt to move buyback tokens with transferFrom (327ms)

Contract: TNDY

√ Setup Contract for Tests (349ms)

√ Transfer tokens to treasury as owner (257ms)

√ [Negative] Attempt to send tokens to the owner from treasury with no notice (145ms)

√ [Negative] Attempt to send tokens from treasury without giving notice. (241ms)

√ Transfer tokens to buyback as owner, Uniswap prevents us from auto burning them. (415ms)

√ [Negative] Attempt to send tokens from the buyback account (211ms)

√ [Negative] Attempt to move buyback tokens with transferFrom (346ms)

Contract: TNDY

√ Setup Contract for Tests (694ms)

√ [Negative] Attempt to give Treasury notice by a non-owner address (143ms)

√ [Negative] Attempt to give Treasury notice for the null address (254ms)

√ [Negative] Attempt to give Treasury notice for a non-restricted address (142ms)

√ [Negative] Attempt to give notice for more TNDY tokens than owned (316ms)

√ Overwrite an expired Treasury TNDY notice (661ms)

√ [Negative] Attempt to overwrite an existing Treasury TNDY notice (480ms)

√ [Negative] Attempt to give Liquidity notice by a non-owner address (217ms)

√ [Negative] Attempt to give Liquidity notice without NFT Id set for the null address (176ms)

√ Set NFT Id for following tests to succeed (876ms)

√ [Negative] Attempt to give Liquidity notice without a Pool Address being set (224ms)

√ Set Pool Address for the following tests to succeed. (229ms)

√ [Negative] Attempt to give Liquidity notice for a non-restricted address (312ms)

√ [Negative] Attempt to give notice for more TNDY tokens than owned (324ms)

√ [Negative] Attempt to give notice for more Liquidity Tokens than owned (328ms)

√ Overwrite an expired Liquidity Token notice (1045ms)

√ [Negative] Attempt to overwrite an active notice that hasn't expired (617ms)

Contract: TNDY

√ Setup Contract for Tests (577ms)

√ Create a zero second notice for Treasury TNDY Tokens and spend some of it (1337ms)

√ [Negative] Attempt to swap when no Treasury notice is given (496ms)

√ [Negative] Attempt to spend more tokens than notice was given for (796ms)

√ [Negative] Attempt to spend more TNDY tokens than notice was given for (687ms)

√ [Negative] Attempt to spend TNDY tokens before notice expires (730ms)

Contract: TNDY

√ Setup Contract for Tests (274ms)

√ [Negative] Attempt to burn tokens as non-buyback account (158ms)

√ Burn buyback tokens (736ms)

Contract: TNDY

√ Setup Contract for Tests (774ms)

√ [Negative] Attempt to swap when no Treasury notice is given (643ms)

√ [Negative] Attempt to spend TNDY tokens before notice expires (1139ms)

√ Create a zero second notice for Treasury TNDY Tokens and spend some of it (7892ms)

√ [Negative] Attempt to swap when Treasury notice was used and should revert to zero/empty again (590ms)

Contract: TNDY

√ Setup Contract for Tests (3068ms)

√ [Negative] Attempt to give notice for more Liquidity Tokens than owned (531ms)

√ [Negative] Attempt to transfer Liquidity tokens without a notice (396ms)

√ Set zero second Liquidity Try transfer more than notice given for (828ms)

√ Set zero second Liquidity Token notice and transfer some of it (1609ms)

√ Set zero second Liquidity Token notice and transfer all of it (1611ms)

√ Transfer owner Tokens, no notice required or other restrictions in V3. (957ms)

Contract: TNDY

√ Setup Contract for Tests

√ [Negative] Attempt to give Liquidity notice when Positions Contract Address is unset (884ms)

71 passing (1m)

--------------------|----------|----------|----------|----------|----------------|

File | % Stmts | % Branch | % Funcs | % Lines |Uncovered Lines |

--------------------|----------|----------|----------|----------|----------------|

contracts\ | 100 | 100 | 100 | 100 | |

DebugLogger.sol | 100 | 100 | 100 | 100 | |

FakePair.sol | 100 | 100 | 100 | 100 | |

FakePositions.sol | 100 | 100 | 100 | 100 | |

TNDYToken.sol | 100 | 100 | 100 | 100 | |

--------------------|----------|----------|----------|----------|----------------|

All files | 100 | 100 | 100 | 100 | |

--------------------|----------|----------|----------|----------|----------------|

> Istanbul reports written to ./coverage/ and ./coverage.json

> solidity-coverage cleaning up, shutting down ganache server

## Slither Output

C:\Dev\talketh.io\Solidity\TNDY\ERC20>slither . --truffle-version truffle@0.8.7

'truffle.cmd compile --all' running (use --truffle-version truffle@x.x.x to use specific version)

Compiling your contracts...

===========================

> Compiling @openzeppelin\contracts\token\ERC20\ERC20.sol

> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol

> Compiling @openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol

> Compiling @openzeppelin\contracts\token\ERC721\IERC721.sol

> Compiling @openzeppelin\contracts\utils\Context.sol

> Compiling @openzeppelin\contracts\utils\introspection\IERC165.sol

> Compiling .\contracts\TNDYToken.sol

> Artifacts written to C:\Dev\talketh.io\Solidity\TNDY\ERC20\build\contracts

> Compiled successfully using:

- solc: 0.8.7+commit.e28d00a7.Emscripten.clang

TNDYToken.liquidityRedemptionNotice(address,uint256,uint256) (TNDYToken.sol#315-330) uses a dangerous strict equality:

- require(bool,string)(noticeLiquidity.releaseDate == 0 || block.timestamp >= noticeLiquidity.releaseDate,Cannot overwrite an active existing notice.) (TNDYToken.sol#326)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

TNDYToken.constructor(uint256,string,string,address,address,address,address).name (TNDYToken.sol#207) shadows:

- ERC20.name() (@openzeppelin\contracts\token\ERC20\ERC20.sol#61-63) (function)

- IERC20Metadata.name() (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#16) (function)

TNDYToken.constructor(uint256,string,string,address,address,address,address).symbol (TNDYToken.sol#207) shadows:

- ERC20.symbol() (@openzeppelin\contracts\token\ERC20\ERC20.sol#69-71) (function)

- IERC20Metadata.symbol() (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#21) (function)

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

TNDYToken.setBuyback(address) (TNDYToken.sol#235-237) should emit an event for:

- buyback = who (TNDYToken.sol#236)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

TNDYToken.setNftId(uint256) (TNDYToken.sol#263-265) should emit an event for:

- nftId = id (TNDYToken.sol#264)

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

TNDYToken.constructor(uint256,string,string,address,address,address,address).treasuryAddr (TNDYToken.sol#207) lacks a zero-check on :

- treasury = treasuryAddr (TNDYToken.sol#211)

TNDYToken.constructor(uint256,string,string,address,address,address,address).buybackAddr (TNDYToken.sol#207) lacks a zero-check on :

- buyback = buybackAddr (TNDYToken.sol#212)

TNDYToken.constructor(uint256,string,string,address,address,address,address).flipAddr (TNDYToken.sol#207) lacks a zero-check on :

- flip = flipAddr (TNDYToken.sol#213)

TNDYToken.constructor(uint256,string,string,address,address,address,address).positionsAddr (TNDYToken.sol#207) lacks a zero-check on :

- positionsAddress = positionsAddr (TNDYToken.sol#214)

TNDYToken.setTreasury(address).who (TNDYToken.sol#228) lacks a zero-check on :

- treasury = who (TNDYToken.sol#229)

TNDYToken.setBuyback(address).who (TNDYToken.sol#235) lacks a zero-check on :

- buyback = who (TNDYToken.sol#236)

TNDYToken.setFlip(address).who (TNDYToken.sol#242) lacks a zero-check on :

- flip = who (TNDYToken.sol#243)

TNDYToken.setPoolAddress(address).who (TNDYToken.sol#249) lacks a zero-check on :

- poolAddress = who (TNDYToken.sol#250)

TNDYToken.setPositionsAddress(address).who (TNDYToken.sol#256) lacks a zero-check on :

- positionsAddress = who (TNDYToken.sol#257)

TNDYToken.setBinanceContract(address).contractAddress (TNDYToken.sol#284) lacks a zero-check on :

- binanceContract = contractAddress (TNDYToken.sol#285)

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

TNDYToken.treasuryTransferNotice(address,uint256,uint256) (TNDYToken.sol#296-305) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(noticeTreasury.releaseDate == 0 || block.timestamp >= noticeTreasury.releaseDate,Cannot overwrite an active existing notice.) (TNDYToken.sol#301)

TNDYToken.liquidityRedemptionNotice(address,uint256,uint256) (TNDYToken.sol#315-330) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(noticeLiquidity.releaseDate == 0 || block.timestamp >= noticeLiquidity.releaseDate,Cannot overwrite an active existing notice.) (TNDYToken.sol#326)

TNDYToken.\_transfer(address,address,uint256) (TNDYToken.sol#340-364) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(noticeTreasury.releaseDate != 0 && block.timestamp >= noticeTreasury.releaseDate,Notice period has not been set or has not expired.) (TNDYToken.sol#343)

- require(bool,string)(amount <= noticeTreasury.amount,Treasury can't transfer more tokens than given notice for.) (TNDYToken.sol#344)

- require(bool,string)(noticeLiquidity.releaseDate != 0 && block.timestamp >= noticeLiquidity.releaseDate,LP notice period has not been set or has not expired.) (TNDYToken.sol#354)

- require(bool,string)(amount <= noticeLiquidity.amount,LP can't transfer more tokens than given notice for.) (TNDYToken.sol#355)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Different versions of Solidity is used:

- Version used: ['0.8.7', '^0.8.0']

- ^0.8.0 (@openzeppelin\contracts\utils\Context.sol#3)

- ^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3)

- ^0.8.0 (@openzeppelin\contracts\utils\introspection\IERC165.sol#3)

- ^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3)

- ^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#3)

- ^0.8.0 (@openzeppelin\contracts\token\ERC721\IERC721.sol#3)

- 0.8.7 (TNDYToken.sol#95)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (@openzeppelin\contracts\utils\Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (@openzeppelin\contracts\utils\introspection\IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC721\IERC721.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version0.8.7 (TNDYToken.sol#95) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

solc-0.8.7 is not recommended for deployment

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

TNDYToken.constructor(uint256,string,string,address,address,address,address) (TNDYToken.sol#207-223) uses literals with too many digits:

- netAssetValue = 132000000000000000000000000 (TNDYToken.sol#215)

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

name() should be declared external:

- ERC20.name() (@openzeppelin\contracts\token\ERC20\ERC20.sol#61-63)

symbol() should be declared external:

- ERC20.symbol() (@openzeppelin\contracts\token\ERC20\ERC20.sol#69-71)

decimals() should be declared external:

- ERC20.decimals() (@openzeppelin\contracts\token\ERC20\ERC20.sol#86-88)

totalSupply() should be declared external:

- ERC20.totalSupply() (@openzeppelin\contracts\token\ERC20\ERC20.sol#93-95)

allowance(address,address) should be declared external:

- ERC20.allowance(address,address) (@openzeppelin\contracts\token\ERC20\ERC20.sol#120-122)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#131-134)

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

- ERC20.transferFrom(address,address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#149-163)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#177-180)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#196-204)

setTreasury(address) should be declared external:

- TNDYToken.setTreasury(address) (TNDYToken.sol#228-230)

setBuyback(address) should be declared external:

- TNDYToken.setBuyback(address) (TNDYToken.sol#235-237)

setFlip(address) should be declared external:

- TNDYToken.setFlip(address) (TNDYToken.sol#242-244)

setPoolAddress(address) should be declared external:

- TNDYToken.setPoolAddress(address) (TNDYToken.sol#249-251)

setPositionsAddress(address) should be declared external:

- TNDYToken.setPositionsAddress(address) (TNDYToken.sol#256-258)

setNftId(uint256) should be declared external:

- TNDYToken.setNftId(uint256) (TNDYToken.sol#263-265)

setNetAssetValue(uint256) should be declared external:

- TNDYToken.setNetAssetValue(uint256) (TNDYToken.sol#274-278)

setBinanceContract(address) should be declared external:

- TNDYToken.setBinanceContract(address) (TNDYToken.sol#284-286)

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

- TNDYToken.treasuryTransferNotice(address,uint256,uint256) (TNDYToken.sol#296-305)

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

- TNDYToken.liquidityRedemptionNotice(address,uint256,uint256) (TNDYToken.sol#315-330)

burn() should be declared external:

- TNDYToken.burn() (TNDYToken.sol#370-372)

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

. analyzed (7 contracts with 75 detectors), 48 result(s) found

C:\Dev\talketh.io\Solidity\TNDY\ERC20>