INFINITY

BLOCKCHAIN SOLUTIONS

Smart contract Audit
Full detailed Report

🌐 infinityblockchainsolutions.com
✉ contact@infinityblockchainsolutions.com
t.me/InfinityDev_77

Table of contents

Contract Review

- Drilo is an ERC20-based standard smart contract deployed on the Ethereum blockchain.

Contract Name	**StandardToken**
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Explorer Link	https://etherscan.io/token/0x9c08eCeeAA3082516c9D1D72c05B69afA0cB414A#code
Contract Address	0x9c08eCeeAA3082516c9D1D72c05B69afA0cB414A
Network	ETHEREUM
Symbol	DLO
Decimals	18
Supply	1,000,000,000 DLO
File Name	Drilo.sol
Audit Date	Oct 04, 2024

Audit Overview

This Audit's purpose was to assess any security issues, logic concerns, and potential improvements. After our audit assessment, we have labelled the security state of the contract of Drilo to be **"SECURED".**

We have made the use of tools such as Solidity Static analysis, Remix IDE, Slither, Surya along with manual code analysis to inspect the token code.

The details of our findings are presented below.

Functions overview

Sr.	Functions	Type	Observation	Result
1.	constructor	write	Passed	Cleared
2.	name	read	Passed	Cleared
3.	symbol	read	Passed	Cleared
4.	decimals	read	Passed	Cleared
5.	totalSupply	read	Passed	Cleared
6.	transfer	write	Passed	Cleared
7.	allowance	read	Passed	Cleared
8.	approve	write	Passed	Cleared
9.	transferFrom	write	Passed	Cleared
10.	balanceOf	read	Passed	Cleared

11.	_transfer	internal	Passed	Cleared
12.	IncreaseAllowance	write	Passed	Cleared
13.	decreaseAllowance	write	Passed	Cleared
14.	_transfer	internal	Passed	Cleared
15.	_mint	internal	Passed	Cleared
16.	_burn	internal	Passed	Cleared
17.	_approve	internal	Passed	Cleared
18.	_spendAllowance	internal	Passed	Cleared
19.	_beforeTokenTransfer	internal	Passed	Cleared
20.	_afterTokenTransfer	internal	Passed	Cleared
21.	_setupDecimals	internal	Passed	Cleared
22.	onlyOwner	modifier	Passed	Cleared
23.	owner	read	Passed	Cleared
24.	_setOwner	internal	Passed	Cleared
25.	renounceOwnership	write	Access only owner	Cleared
26.	transferOwnership	write	Access only owner	Cleared
27.	_transferOwnership	internal	Passed	Cleared

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are on industry standard libraries like Openzeppelin.

IBS Risk Analysis

Category	Result
🟢 **Buy Fee**	0%
🟢 **Sell Fee**	0%
🟢 **Cannot Buy**	False
🟢 **Cannot Sell**	False
🟢 **Maximum Tax Cap**	False
🟢 **Tax Modifiable?**	Not Detected
🟢 **Fee Check**	Not Detected
🟢 **Honeypot Issue?**	Not Detected
🟢 **Trading Cooldown**	Not Detected
🟢**Trade Pausable?**	No
🟢 **Transfer Pausable?**	No
🟢 **Is it Anti-whale?**	No

🟢 **Is Anti-bot?**	Not Detected
🟢 **Can Addresses be Blacklisted?**	Not Detected
🟢 **Blacklist Check**	Passed
🟢 **Mint After deployment?**	No
🟢 **Is it Proxy?**	No
🟢 **Hidden Owner?**	Not Detected
🟢 **Self-Destruction?**	Not Detected

Risk Analysis Result: PASSED

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities that can lead to token loss etc.
High	Will definitely cause problems; this needs to be adjusted.
Medium	Will likely cause problems and it is recommended to adjust
Low	Won't cause any problems, but can be adjusted for improvement
Informational	Does not compromise the functionality of the contract in any way

Findings Break Down

Risk Level	Unresolved	Acknowledged	Resolved
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	1	1	0

Centralization

Criticality	**Informational**
Status	**Acknowledged**

Description:

This smart contract has some functions which can be executed by the Admin (Owner) only. If the owner wallet private key would be compromised, then it would create risk.

Following are Owner functions:

Ownable.sol

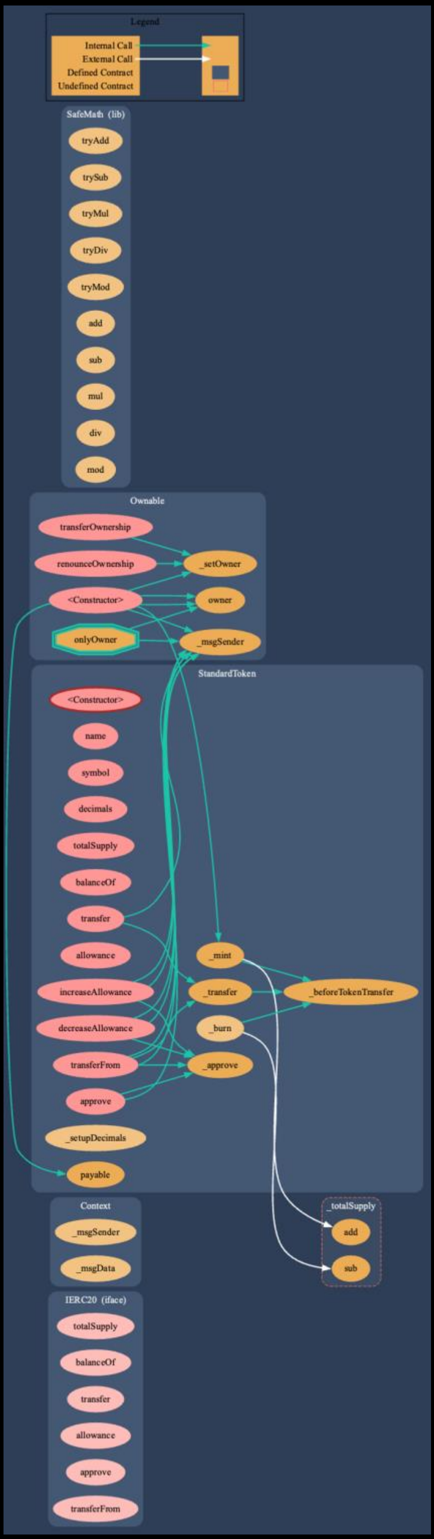
1. renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
2. transferOwnership: The current owner can transfer ownership of the contract to a new account.

Solution

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Graphical Findings

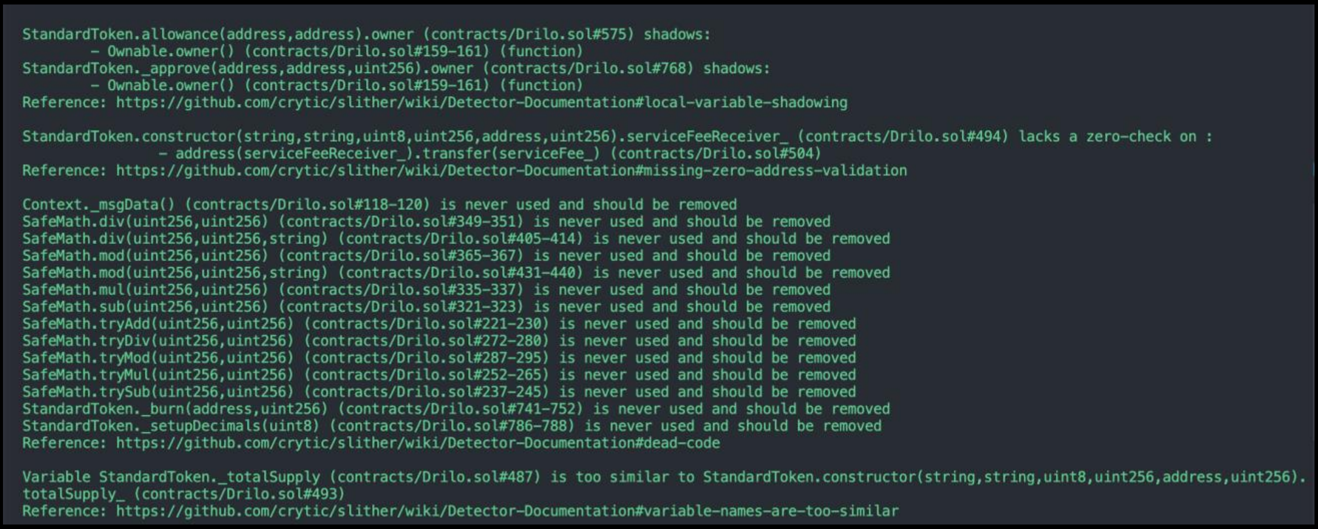
Call Graph Diagram – Drilo.sol



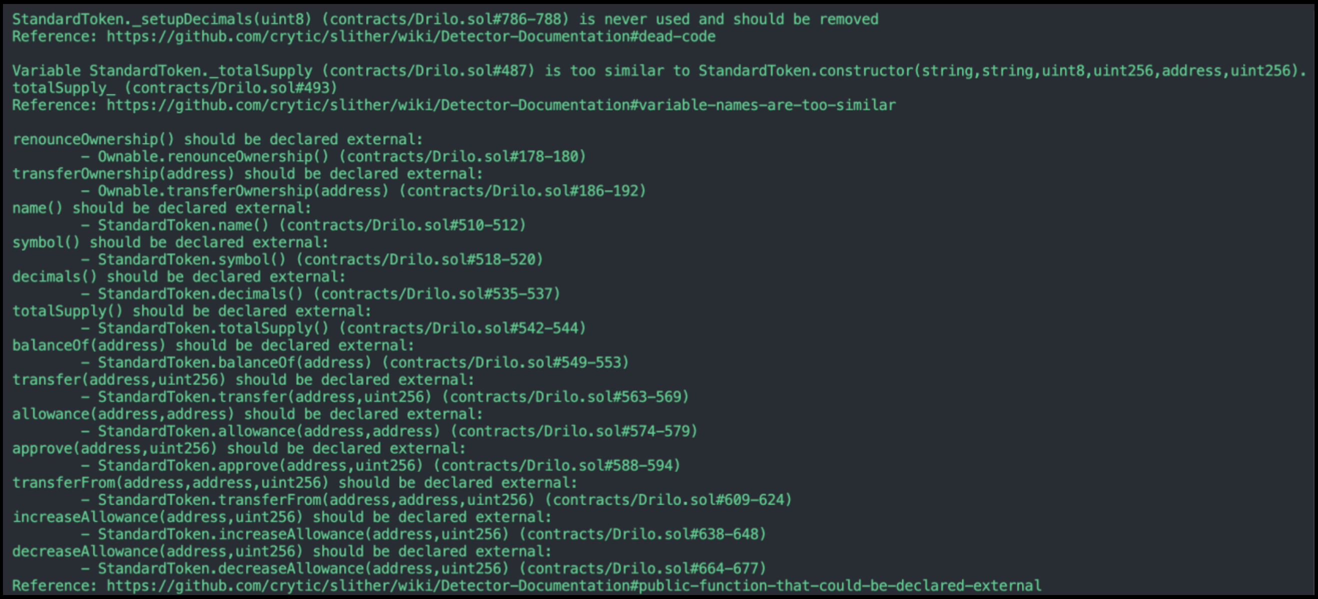
Slither Findings Log

Slither is a static analysis tool specifically designed for auditing Solidity smart contracts. It provides fast and comprehensive security analysis, helping developers and auditors identify potential vulnerabilities and bugs in smart contracts before deployment.

Slither Log of Drilo.sol:



```
StandardToken.allowance(address,address).owner (contracts/Drilo.sol#575) shadows:
        - Ownable.owner() (contracts/Drilo.sol#159-161) (function)
StandardToken._approve(address,address,uint256).owner (contracts/Drilo.sol#768) shadows:
        - Ownable.owner() (contracts/Drilo.sol#159-161) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

StandardToken.constructor(string,string,uint8,uint256,address,uint256).serviceFeeReceiver_ (contracts/Drilo.sol#494) lacks a zero-check on :
                - address(serviceFeeReceiver_).transfer(serviceFee_) (contracts/Drilo.sol#504)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Context._msgData() (contracts/Drilo.sol#118-120) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Drilo.sol#349-351) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Drilo.sol#405-414) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Drilo.sol#365-367) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Drilo.sol#431-440) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Drilo.sol#335-337) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Drilo.sol#321-323) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Drilo.sol#221-230) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Drilo.sol#272-280) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Drilo.sol#287-295) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Drilo.sol#252-265) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Drilo.sol#237-245) is never used and should be removed
StandardToken._burn(address,uint256) (contracts/Drilo.sol#741-752) is never used and should be removed
StandardToken._setupDecimals(uint8) (contracts/Drilo.sol#786-788) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Variable StandardToken._totalSupply (contracts/Drilo.sol#487) is too similar to StandardToken.constructor(string,string,uint8,uint256,address,uint256).
totalSupply_ (contracts/Drilo.sol#493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

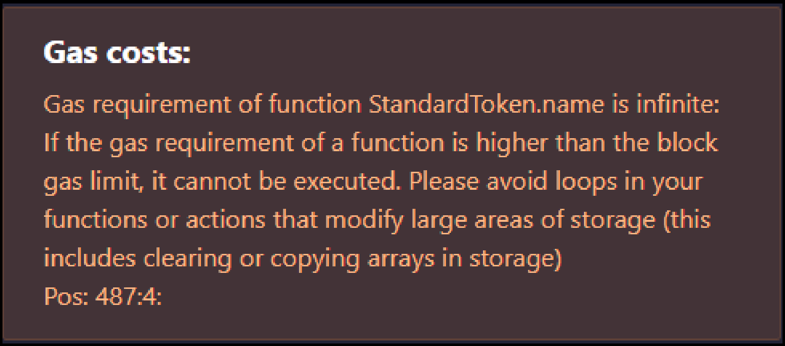


```
StandardToken._setupDecimals(uint8) (contracts/Drilo.sol#786-788) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Variable StandardToken._totalSupply (contracts/Drilo.sol#487) is too similar to StandardToken.constructor(string,string,uint8,uint256,address,uint256).
totalSupply_ (contracts/Drilo.sol#493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/Drilo.sol#178-180)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/Drilo.sol#186-192)
name() should be declared external:
        - StandardToken.name() (contracts/Drilo.sol#510-512)
symbol() should be declared external:
        - StandardToken.symbol() (contracts/Drilo.sol#518-520)
decimals() should be declared external:
        - StandardToken.decimals() (contracts/Drilo.sol#535-537)
totalSupply() should be declared external:
        - StandardToken.totalSupply() (contracts/Drilo.sol#542-544)
balanceOf(address) should be declared external:
        - StandardToken.balanceOf(address) (contracts/Drilo.sol#549-553)
transfer(address,uint256) should be declared external:
        - StandardToken.transfer(address,uint256) (contracts/Drilo.sol#563-569)
allowance(address,address) should be declared external:
        - StandardToken.allowance(address,address) (contracts/Drilo.sol#574-579)
approve(address,uint256) should be declared external:
        - StandardToken.approve(address,uint256) (contracts/Drilo.sol#588-594)
transferFrom(address,address,uint256) should be declared external:
        - StandardToken.transferFrom(address,address,uint256) (contracts/Drilo.sol#609-624)
increaseAllowance(address,uint256) should be declared external:
        - StandardToken.increaseAllowance(address,uint256) (contracts/Drilo.sol#638-648)
decreaseAllowance(address,uint256) should be declared external:
        - StandardToken.decreaseAllowance(address,uint256) (contracts/Drilo.sol#664-677)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Solidity Static Analysis

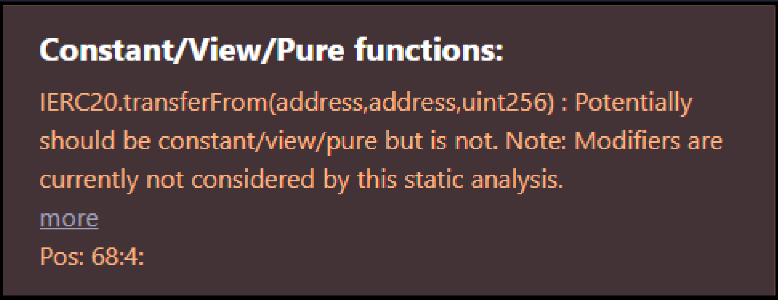
Static code analysis is a technique used to detect common coding issues before the release of a program. It involves reviewing the code either manually or through the use of automated tools. These tools can scan the code without the need for execution, identifying potential problems in advance.

Drilo.sol



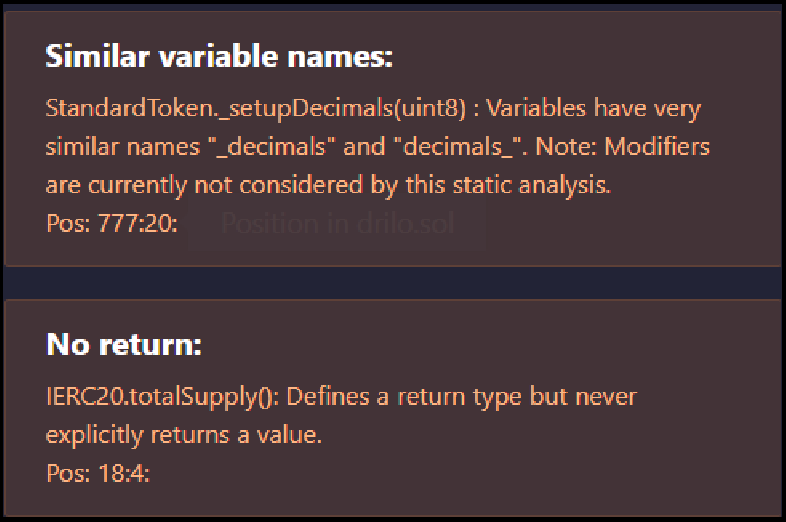
Gas costs:

Gas requirement of function StandardToken.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 487:4:



Constant/View/Pure functions:

IERC20.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 68:4:

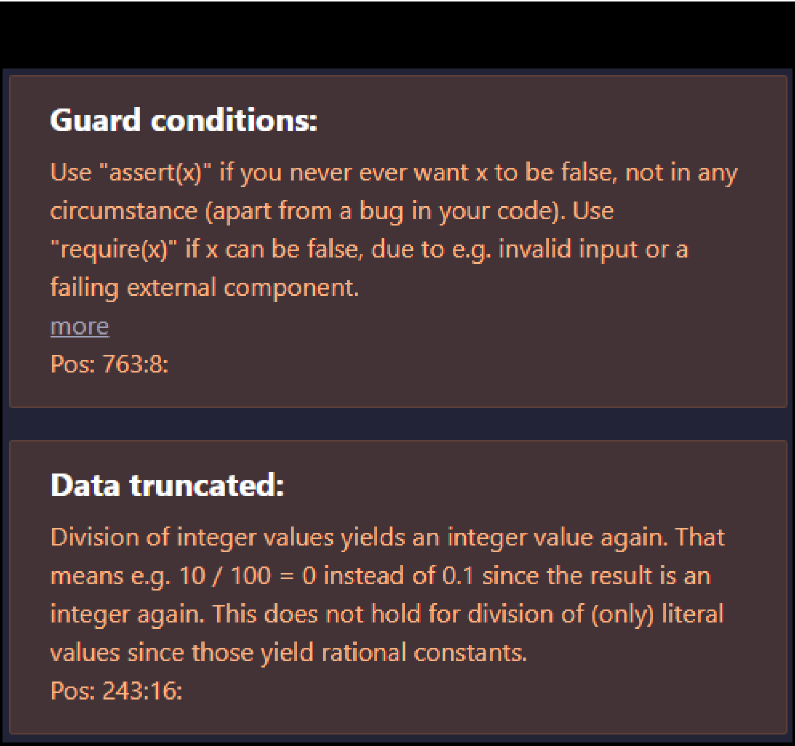


Similar variable names:

StandardToken._setupDecimals(uint8) : Variables have very similar names "_decimals" and "decimals_". Note: Modifiers are currently not considered by this static analysis.
Pos: 777:20: Position in drilo.sol

No return:

IERC20.totalSupply(): Defines a return type but never explicitly returns a value.
Pos: 18:4:



Guard conditions:

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

more

Pos: 763:8:

Data truncated:

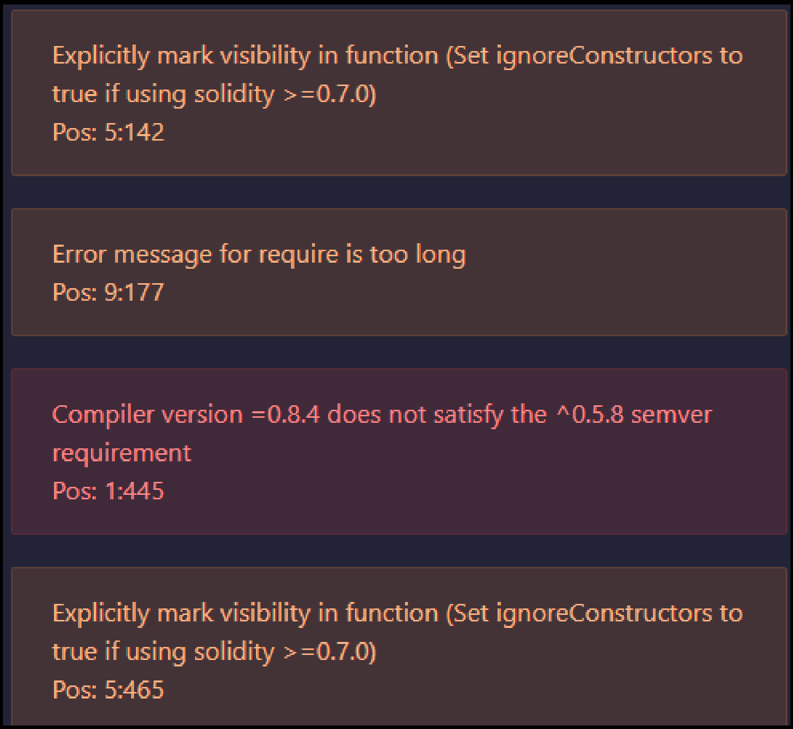
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 243:16:

Solhint Linter

Linters are utility tools designed to analyze source code and identify programming errors, bugs, and stylistic issues.

Drilo.sol



Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

Pos: 5:142

Error message for require is too long

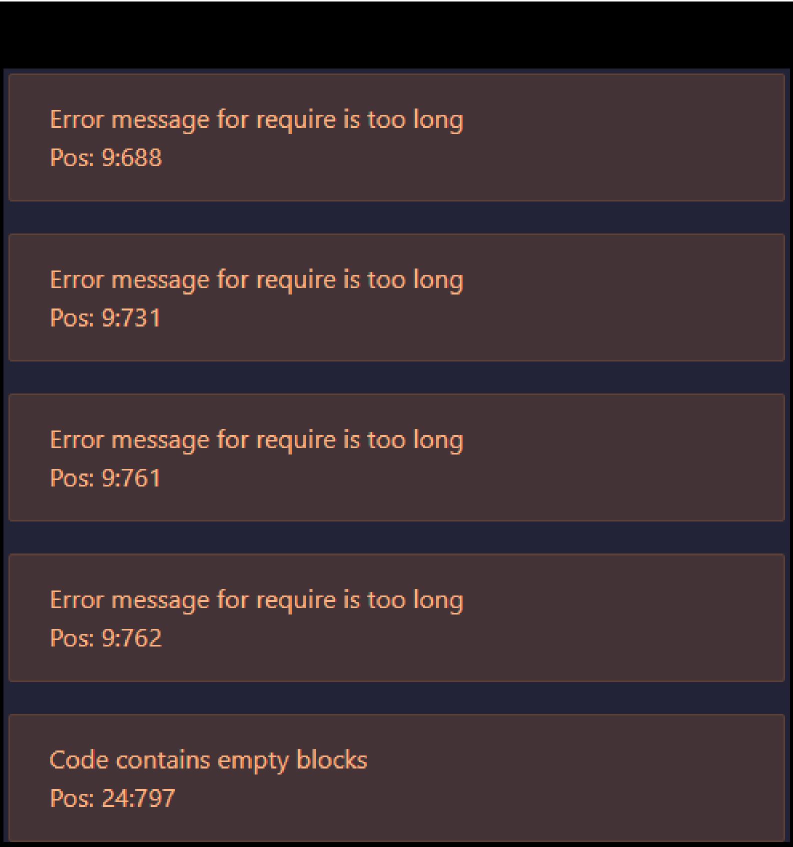
Pos: 9:177

Compiler version =0.8.4 does not satisfy the ^0.5.8 semver requirement

Pos: 1:445

Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

Pos: 5:465



Error message for require is too long
Pos: 9:688

Error message for require is too long
Pos: 9:731

Error message for require is too long
Pos: 9:761

Error message for require is too long
Pos: 9:762

Code contains empty blocks
Pos: 24:797

Software analysis result:

These tools reported many **false positive results** and some are informational issues. So, those issues can be **safely ignored**.

Software analysis result: **PASSED**

Final Summary

This audit investigated any possible issues inside Drilo token contract. Our analysis reported no major issues or critical errors. One point to be noted is that the contract owner can access some functions but they cannot be used in a malicious way to disturb user's transactions.

Given that potential test cases for such smart contract protocols can be limitless, we cannot guarantee future outcomes. We have utilized the latest static tools and conducted thorough manual reviews to cover as many test cases as possible.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

As already stated above, we have concluded that the contract's security state is **"SECURED"**

Disclaimer

The information provided in this audit report is based on a thorough analysis conducted by **Infinity Blockchain Solutions**. This audit is intended to evaluate the security, functionality, and best practices of the token's smart contract code.

Our audit is limited to the analysis of the smart contract code provided to us. It does not cover vulnerabilities that may arise from external systems, third-party integrations, or the operational environment in which the token will be deployed.

While we employ industry-standard methodologies and tools, including both manual and automated processes, this audit cannot guarantee the complete absence of vulnerabilities. There may be undiscovered security flaws that were not identified during the review process.

The audit reflects the state of the token's smart contract at the time of review. We cannot predict or protect against future vulnerabilities or exploits that may emerge due to changes in the contract, its environment, or advancements in attack techniques.

Implementing the audit recommendations is the sole responsibility of the project team. Infinity Blockchain Solutions is not liable for any losses or damages resulting from the failure to address issues identified in the audit or any future vulnerabilities that arise post-audit.

This audit does not constitute financial advice. The evaluation is focused solely on the technical aspects of the smart contract. Investors and stakeholders should conduct their own independent research and due diligence before making any financial decisions related to the audited token.

Infinity Blockchain Solutions makes no warranties or representations, either express or implied, regarding the safety, reliability, or security of the audited smart contract. We are not responsible for any loss, damage, or legal issues that may arise from the use or misuse of the token.

About Infinity Blockchain Solutions

Infinity Blockchain Solutions is a leading Web 3.0 development company dedicated to advancing the blockchain ecosystem. Founded with a mission to empower the future of digital innovation, Infinity Blockchain Solutions offers a comprehensive suite of services, including Crypto Token creation, Website development, ICO creation, smart contract audits and more.

With a reputation for excellence and a commitment to security, we have collaborated with numerous projects, contributing to the growth and integrity of the blockchain space. Our expert team provides reliable solutions that ensure the success and safety of our clients' ventures, securing their digital assets and fostering innovation in the decentralized landscape.

INFINITY
BLOCKCHAIN SOLUTIONS

infinityblockchainsolutions.com