

SMART CONTRACT

Security Audit Report

Customer: DeGeThal
Website: <https://www.degethal.io>
Platform: Binance Smart Chain
Language: Solidity
Date: June 19th, 2021

Table of contents

Introduction	4
Project Background	4
Audit Scope	4
Claimed Smart Contract Features	5
Audit Summary	6
Technical Quick Stats	7
Code Quality	8
Documentation	8
Use of Dependencies	8
AS-IS overview	9
Severity Definitions	12
Audit Findings	12
Conclusion	16
Our Methodology	17
Disclaimers	19
Appendix	
• Code Flow Diagram	20
• Slither Report Log	21

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

We were contracted by the DeGeThal team to perform the Security audit of the DeGeThal Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 19th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

DeGeThal is: easy to use, securely stored and it is tailored to everyone's needs. DeGeThal is building a DeGeThal crypto currency for everyone, the currency that is designed to serve as a payment token for all transactions in the ecosystem.

Audit scope

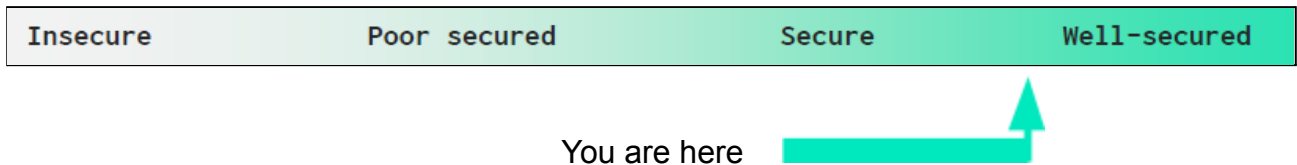
Name	Code Review and Security Analysis Report for DeGeThal Token Smart Contract
Platform	BSC / Solidity
File	DeGeThal.sol
Smart Contract Online Code	https://testnet.bscscan.com/address/0xC62b6518B8CF5F5BB9A1440d110C9F648da695Ee#code
File MD5 Hash	F0868A76BCB6BF2402585236F09A8A96
Audit Date	June 19th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: DeGeThal	YES, This is valid.
Symbol: DTM	YES, This is valid.
Decimal: 18	YES, This is valid.
TaxFee: 3%	YES, This is valid.Owner can change this fee.
MaxTxAmount: 500000	YES, This is valid.Owner can change this fee.
LiquidityFee: 3%	YES, This is valid.Owner can change this fee.
The owner can access functions like :updateRouter, updateTokenLimit, excludeFromReward, includeInReward, excludeFromFee, includeInFee, setTaxFeePercent, setLiquidityFeePercent, setMaxTxPercent, setSwapAndLiquifyEnabled, etc.	YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.

Audit Summary

According to the standard audit assessment, Customer's solidity smart contract is **secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.



We used various tools like MythX, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 5 low and some very low level issues.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Moderated
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. This is a compact and well written contract.

The libraries in the DeGeThal Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the DeGeThal Token.

The DeGeThal team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

Documentation

We were given DeGeThal Token smart contract code in the form of a BscScan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website <https://www.degethal.io> which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

DeGeThal token is a smart contract, having functionality like swap and Liquify, burn, etc.

DeGeThal.sol

(1) Interface

- (a) interface
- (b) IUniswapV2Factory
- (c) IUniswapV2Pair
- (d) IUniswapV2Router01
- (e) IUniswapV2Router02

(2) Inherited contracts

- (a) Context
- (b) Ownable
- (c) IERC20

(3) Usages

- (a) using SafeMath for uint256;
- (b) using Address for address;

(4) Events

- (a) event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
- (b) event SwapAndLiquifyEnabledUpdated(bool enabled);
- (c) event SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lockTheSwap	modifier	Passed	No Issue
2	updateRouter	external	access only Owner	No Issue
3	updateTokenLimit	write	Passed	No Issue
4	name	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	decimals	read	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	balanceOf	read	Passed	No Issue
9	transfer	write	Passed	No Issue
10	allowance	read	Passed	No Issue
11	approve	write	Passed	No Issue
12	transferFrom	write	Passed	No Issue
13	increaseAllowance	write	Passed	No Issue
14	decreaseAllowance	write	Passed	No Issue
15	isExcludedFromReward	read	Passed	No Issue
16	totalFees	read	Passed	No Issue
17	deliver	write	Passed	No Issue
18	reflectionFromToken	read	Passed	No Issue
19	tokenFromReflection	read	Passed	No Issue
20	excludeFromReward	write	Missing Events	No Issue
21	includeInReward	external	Infinite loop possibility	Refer Audit Findings
22	excludeFromFee	write	Missing Events	No Issue
23	includeInFee	write	Missing Events	No Issue
24	setTaxFeePercent	external	Missing Events	No Issue
25	setLiquidityFeePercent	external	Missing Events	No Issue
26	setMaxTxPercent	external	Missing Events	No Issue
27	setSwapAndLiquifyEnabled	write	Missing Events	No Issue
28	_reflectFee	write	Passed	No Issue
29	_getValues	read	Passed	No Issue
30	_getTValues	read	Passed	No Issue
31	_getRValues	write	Passed	No Issue
32	_getRate	read	Passed	No Issue
33	_getCurrentSupply	read	Passed	No Issue
34	_takeLiquidity	write	Passed	No Issue
35	calculateTaxFee	read	Passed	No Issue
36	calculateLiquidityFee	read	Passed	No Issue
37	removeAllFee	write	Passed	No Issue
38	restoreAllFee	write	Passed	No Issue
39	isExcludedFromFee	read	Passed	No Issue
40	approve	write	Passed	No Issue
41	transfer	write	Passed	No Issue
42	swapAndLiquify	write	Passed	No Issue

43	swapTokensForEth	write	Passed	No Issue
44	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
45	tokenTransfer	write	Passed	No Issue
46	transferBothExcluded	write	Passed	No Issue
47	transferStandard	write	Passed	No Issue
48	transferToExcluded	write	Passed	No Issue
49	transferFromExcluded	write	Passed	No Issue
50	owner	read	Passed	No Issue
51	onlyOwner	modifier	Passed	No Issue
52	renounceOwnership	write	access only Owner	No Issue
53	transferOwnership	write	access only Owner	No Issue
54	geUnlockTime	read	Passed	No Issue
55	lock	write	access only Owner	No Issue
56	unlock	write	Ownership can be regained	Refer Audit Findings
57	_msgSender	internal	Passed	No Issue
58	_msgData	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token 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 loss
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	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

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loop possibility:

```
function includeInReward(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

If there are so many excluded wallets, then this logic will fail, as it might hit the block's gas limit. If there are very limited exceptions, then this will work, but will cost more gas.

Resolution: We suggest excluding limited wallets only.

(2) Make variables constant:

```
string private _name = "DeGeThal";
string private _symbol = "DTM";
uint8 private _decimals = 18;
```

Following variables will be unchanged. So, please make it constant. It will save some gas.

- name
- symbol
- decimals

Resolution: Declare those variables as constant. Just put a constant keyword.

(3) Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but he can regain ownership by following the steps below:

- Owner calls the lock function in contract to set the current owner as `_previousOwner`.
- Owner calls unlock to unlock contract and set `_owner = _previousOwner`.
- Owner called `renounceOwnership` to leave the contract without the owner.
- Owner calls unlock to regain ownership.

Resolution: We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership first before calling the lock function.

(4) Centralized risk in `addLiquidity`:

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

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

In `addLiquidityETH` function, the owner gets DTM Tokens from the Pool. If the private key of the owner's wallet is compromised, then it will create a problem.

Resolution: Ideally this can be a governance smart contract. On another hand, the owner can accept this risk and handle the private key very securely.

(5) Missing Events:

Missing Events log for some functions:

- excludeFromFee
- excludeFromReward
- includeInFee
- includeInReward
- setLiquidityFeePercent
- setMaxTxPercent
- setTaxFeePercent
- setSwapAndLiquifyEnabled

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler level bugs.

Resolution: Please use 0.8.5 which is the latest version.

(2) external instead of public:

If any function is not called from inside the smart contract, then it is better to declare it as external instead of public. As it saves some gas as well.

<https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices>

Centralization

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

- `updateRouter`: Owner can Create a uniswap pair for this new token.
- `updateTokenLimit`: The Owner can change the number of tokens required to trigger Swap.
- `excludeFromReward`: Owner can check if the account is excluded or not.
- `includeInReward`: Owner can check `includeInReward`.
- `excludeFromFee`: Owner can check if the account is excludedFee or not.
- `includeInFee`: Owner can check `includeInFee`.
- `setTaxFeePercent`: Owner can set Tax Fee Percent.
- `setLiquidityFeePercent`: Owner can set Liquidity Fee Percent.
- `setMaxTxPercent`: Owner can set Max Tx Percent.
- `setSwapAndLiquifyEnabled`: Owner can set Swap And Liquify Enabled.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts 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.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

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

Security state of the reviewed contract, based on standard audit procedure scope, is “**Secured**”.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

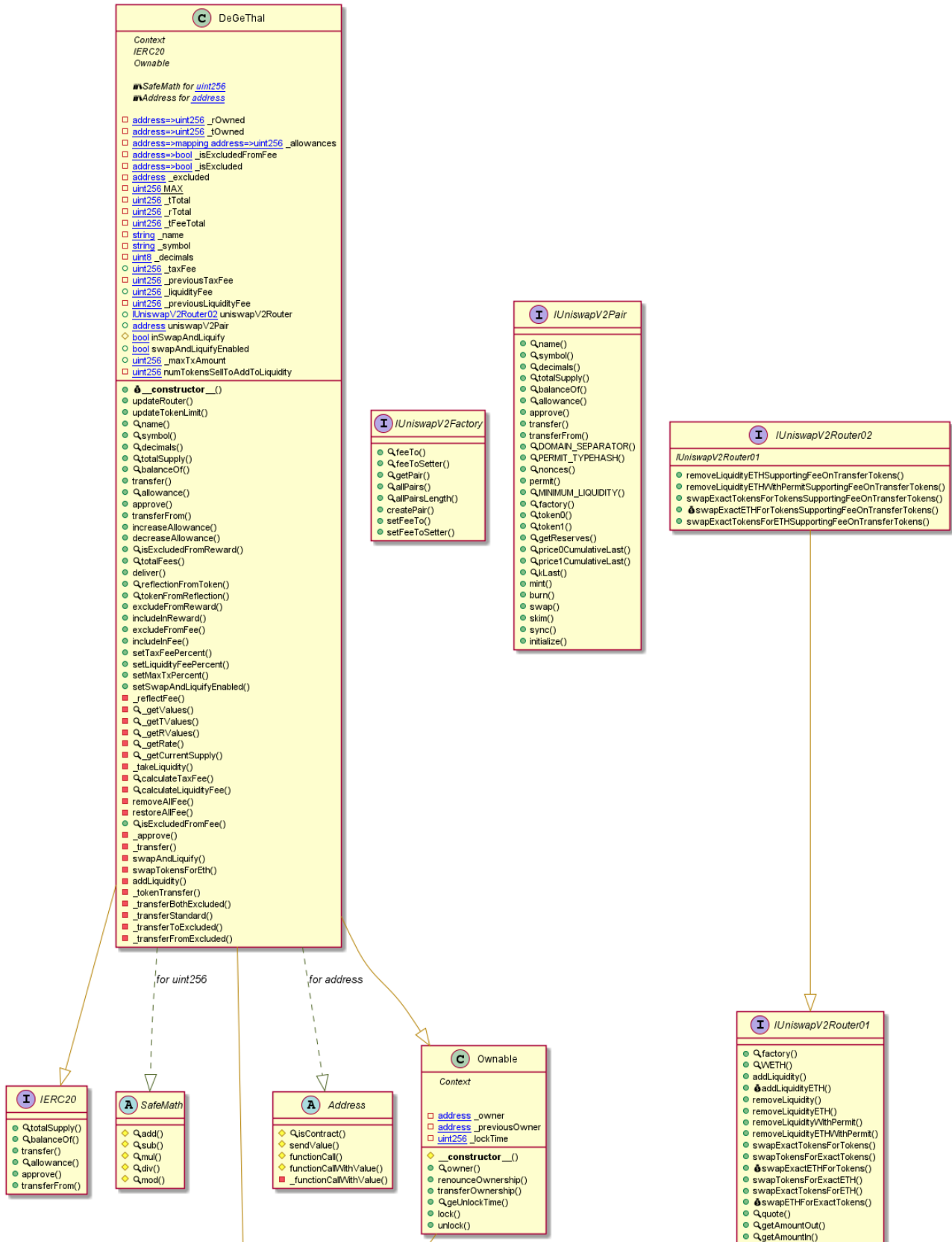
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - DegeThal Token



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Slither Results Log

Slither log >> DeGeThal.sol

INFO:Detectors:

DeGeThal.addLiquidity(uint256,uint256) (DeGeThal.sol#1071-1084) sends eth to arbitrary user

Dangerous calls:

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

Reference:

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

INFO:Detectors:

Reentrancy in DeGeThal._transfer(address,address,uint256) (DeGeThal.sol#985-1028):

External calls:

- swapAndLiquify(contractTokenBalance) (DeGeThal.sol#1015)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

-

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (DeGeThal.sol#1062-1068)

External calls sending eth:

- swapAndLiquify(contractTokenBalance) (DeGeThal.sol#1015)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

State variables written after the call(s):

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (DeGeThal.sol#941)

- _rOwned[sender] = _rOwned[sender].sub(rAmount) (DeGeThal.sol#1120)

- _rOwned[sender] = _rOwned[sender].sub(rAmount) (DeGeThal.sol#1129)

- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (DeGeThal.sol#1121)

- _rOwned[sender] = _rOwned[sender].sub(rAmount) (DeGeThal.sol#1110)

- _rOwned[sender] = _rOwned[sender].sub(rAmount) (DeGeThal.sol#1140)

- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (DeGeThal.sol#1141)

- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (DeGeThal.sol#1131)

- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (DeGeThal.sol#1112)

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _rTotal = _rTotal.sub(rFee) (DeGeThal.sol#896)

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (DeGeThal.sol#943)

- _tOwned[sender] = _tOwned[sender].sub(tAmount) (DeGeThal.sol#1139)

- _tOwned[sender] = _tOwned[sender].sub(tAmount) (DeGeThal.sol#1109)

- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (DeGeThal.sol#1130)

- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (DeGeThal.sol#1111)

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

INFO:Detectors:

DeGeThal.addLiquidity(uint256,uint256) (DeGeThal.sol#1071-1084) ignores return value by

uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

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

INFO:Detectors:

DeGeThal.allowance(address,address).owner (DeGeThal.sol#783) shadows:

- Ownable.owner() (DeGeThal.sol#416-418) (function)

DeGeThal._approve(address,address,uint256).owner (DeGeThal.sol#977) shadows:

- Ownable.owner() (DeGeThal.sol#416-418) (function)

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

INFO:Detectors:

Reentrancy in DeGeThal._transfer(address,address,uint256) (DeGeThal.sol#985-1028):

External calls:

- swapAndLiquify(contractTokenBalance) (DeGeThal.sol#1015)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (DeGeThal.sol#1062-1068)

External calls sending eth:

- swapAndLiquify(contractTokenBalance) (DeGeThal.sol#1015)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

State variables written after the call(s):

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _liquidityFee = _previousLiquidityFee (DeGeThal.sol#970)

- _liquidityFee = 0 (DeGeThal.sol#965)

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _previousLiquidityFee = _liquidityFee (DeGeThal.sol#962)

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _previousTaxFee = _taxFee (DeGeThal.sol#961)

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _tFeeTotal = _tFeeTotal.add(tFee) (DeGeThal.sol#897)

- _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)

- _taxFee = _previousTaxFee (DeGeThal.sol#969)

- _taxFee = 0 (DeGeThal.sol#964)

Reentrancy in DeGeThal.constructor() (DeGeThal.sol#723-739):

External calls:

- uniswapV2Pair =

IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (DeGeThal.sol#728-729)

State variables written after the call(s):

- _isExcludedFromFee[owner()] = true (DeGeThal.sol#735)

- _isExcludedFromFee[address(this)] = true (DeGeThal.sol#736)

- uniswapV2Router = _uniswapV2Router (DeGeThal.sol#732)

Reentrancy in DeGeThal.swapAndLiquify(uint256) (DeGeThal.sol#1030-1051):

External calls:

- swapTokensForEth(half) (DeGeThal.sol#1042)

-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (DeGeThal.sol#1062-1068)

- addLiquidity(otherHalf,newBalance) (DeGeThal.sol#1048)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

External calls sending eth:

- addLiquidity(otherHalf,newBalance) (DeGeThal.sol#1048)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

State variables written after the call(s):

- addLiquidity(otherHalf,newBalance) (DeGeThal.sol#1048)

- _allowances[owner()][spender] = amount (DeGeThal.sol#981)

Reentrancy in DeGeThal.transferFrom(address,address,uint256) (DeGeThal.sol#792-796):

External calls:

- _transfer(sender,recipient,amount) (DeGeThal.sol#793)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (DeGeThal.sol#1062-1068)

External calls sending eth:

- _transfer(sender,recipient,amount) (DeGeThal.sol#793)

- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)

State variables written after the call(s):

- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (DeGeThal.sol#794)

- _allowances[owner()][spender] = amount (DeGeThal.sol#981)

Reentrancy in DeGeThal.updateRouter(address) (DeGeThal.sol#741-749):

External calls:

- uniswapV2Pair =
 IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
 (DeGeThal.sol#744-745)
 State variables written after the call(s):
 - uniswapV2Router = _uniswapV2Router (DeGeThal.sol#748)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2>
 INFO:Detectors:
 Reentrancy in DeGeThal._transfer(address,address,uint256) (DeGeThal.sol#985-1028):
 External calls:
 - swapAndLiquify(contractTokenBalance) (DeGeThal.sol#1015)
 - uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)
 -
 uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
 his),block.timestamp) (DeGeThal.sol#1062-1068)
 External calls sending eth:
 - swapAndLiquify(contractTokenBalance) (DeGeThal.sol#1015)
 - uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)
 Event emitted after the call(s):
 - Transfer(sender,recipient,tTransferAmount) (DeGeThal.sol#1124)
 - _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)
 - Transfer(sender,recipient,tTransferAmount) (DeGeThal.sol#1144)
 - _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)
 - Transfer(sender,recipient,tTransferAmount) (DeGeThal.sol#1134)
 - _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)
 - Transfer(sender,recipient,tTransferAmount) (DeGeThal.sol#1115)
 - _tokenTransfer(from,to,amount,takeFee) (DeGeThal.sol#1027)
 Reentrancy in DeGeThal.constructor() (DeGeThal.sol#723-739):
 External calls:
 - uniswapV2Pair =
 IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
 (DeGeThal.sol#728-729)
 Event emitted after the call(s):
 - Transfer(address(0),_msgSender(),_tTotal) (DeGeThal.sol#738)
 Reentrancy in DeGeThal.swapAndLiquify(uint256) (DeGeThal.sol#1030-1051):
 External calls:
 - swapTokensForEth(half) (DeGeThal.sol#1042)
 -
 uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
 his),block.timestamp) (DeGeThal.sol#1062-1068)
 - addLiquidity(otherHalf,newBalance) (DeGeThal.sol#1048)
 - uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)
 External calls sending eth:
 - addLiquidity(otherHalf,newBalance) (DeGeThal.sol#1048)
 - uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)
 Event emitted after the call(s):
 - Approval(owner,spender,amount) (DeGeThal.sol#982)
 - addLiquidity(otherHalf,newBalance) (DeGeThal.sol#1048)
 - SwapAndLiquify(half,newBalance,otherHalf) (DeGeThal.sol#1050)
 Reentrancy in DeGeThal.transferFrom(address,address,uint256) (DeGeThal.sol#792-796):
 External calls:
 - _transfer(sender,recipient,amount) (DeGeThal.sol#793)
 - uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)
 -
 uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
 his),block.timestamp) (DeGeThal.sol#1062-1068)
 External calls sending eth:
 - _transfer(sender,recipient,amount) (DeGeThal.sol#793)
 - uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (DeGeThal.sol#1076-1083)
 Event emitted after the call(s):

- Approval(owner,spender,amount) (DeGeThal.sol#982)
 - `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20:transfer amount exceeds allowance))` (DeGeThal.sol#794)

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

Ownable.unlock() (DeGeThal.sol#463-468) uses timestamp for comparisons
 Dangerous comparisons:

- `require(bool,string)(now > _lockTime,Contract is locked until 7 days)` (DeGeThal.sol#465)

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

Address.isContract(address) (DeGeThal.sol#268-277) uses assembly

- INLINE ASM (DeGeThal.sol#275)

 Address._functionCallWithValue(address,bytes,uint256,string) (DeGeThal.sol#361-382) uses assembly

- INLINE ASM (DeGeThal.sol#374-377)

 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>
 INFO:Detectors:

Address._functionCallWithValue(address,bytes,uint256,string) (DeGeThal.sol#361-382) is never used and should be removed
 Address.functionCall(address,bytes) (DeGeThal.sol#321-323) is never used and should be removed
 Address.functionCall(address,bytes,string) (DeGeThal.sol#331-333) is never used and should be removed
 Address.functionCallWithValue(address,bytes,uint256) (DeGeThal.sol#346-348) is never used and should be removed
 Address.functionCallWithValue(address,bytes,uint256,string) (DeGeThal.sol#356-359) is never used and should be removed
 Address.isContract(address) (DeGeThal.sol#268-277) is never used and should be removed
 Address.sendValue(address,uint256) (DeGeThal.sol#295-301) is never used and should be removed
 Context._msgData() (DeGeThal.sol#240-243) is never used and should be removed
 SafeMath.mod(uint256,uint256) (DeGeThal.sol#213-215) is never used and should be removed
 SafeMath.mod(uint256,uint256,string) (DeGeThal.sol#229-232) is never used and should be removed
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>
 INFO:Detectors:

DeGeThal._rTotal (DeGeThal.sol#687) is set pre-construction with a non-constant function or state variable:

- (MAX - (MAX % _tTotal))

 DeGeThal._previousTaxFee (DeGeThal.sol#695) is set pre-construction with a non-constant function or state variable:

- `_taxFee`

 DeGeThal._previousLiquidityFee (DeGeThal.sol#698) is set pre-construction with a non-constant function or state variable:

- `_liquidityFee`

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

Low level call in Address.sendValue(address,uint256) (DeGeThal.sol#295-301):

- (success) = recipient.call{value: amount}() (DeGeThal.sol#299)

 Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (DeGeThal.sol#361-382):

- (success,returndata) = target.call{value: weiValue}(data) (DeGeThal.sol#365)

 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls>
 INFO:Detectors:

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (DeGeThal.sol#502) is not in mixedCase
 Function IUniswapV2Pair.PERMIT_TYPEHASH() (DeGeThal.sol#503) is not in mixedCase
 Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (DeGeThal.sol#520) is not in mixedCase
 Function IUniswapV2Router01.WETH() (DeGeThal.sol#540) is not in mixedCase
 Parameter DeGeThal.updateRouter(address)._newDexAddress (DeGeThal.sol#741) is not in mixedCase
 Parameter DeGeThal.setSwapAndLiquifyEnabled(bool)._enabled (DeGeThal.sol#887) is not in mixedCase
 Parameter DeGeThal.calculateTaxFee(uint256)._amount (DeGeThal.sol#946) is not in mixedCase
 Parameter DeGeThal.calculateLiquidityFee(uint256)._amount (DeGeThal.sol#952) is not in mixedCase
 Variable DeGeThal._taxFee (DeGeThal.sol#694) is not in mixedCase
 Variable DeGeThal._liquidityFee (DeGeThal.sol#697) is not in mixedCase
 Variable DeGeThal._maxTxAmount (DeGeThal.sol#706) is not in mixedCase
 Reference:
<https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>
 INFO:Detectors:

Redundant expression "this (DeGeThal.sol#241)" inContext (DeGeThal.sol#235-244)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>
 INFO:Detectors:

Variable

IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (DeGeThal.sol#545) is too similar to

IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (DeGeThal.sol#546)

Variable DeGeThal._transferBothExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1108) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (DeGeThal.sol#917) is too similar to DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (DeGeThal.sol#917) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal._transferStandard(address,address,uint256).rTransferAmount (DeGeThal.sol#1119) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (DeGeThal.sol#917) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal._transferStandard(address,address,uint256).rTransferAmount (DeGeThal.sol#1119) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal._transferBothExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1108) is too similar to DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal._transferBothExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1108) is too similar to DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal._transferFromExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1138) is too similar to

DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal._transferFromExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1138) is too similar to

DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (DeGeThal.sol#917) is too similar to DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal._transferStandard(address,address,uint256).rTransferAmount (DeGeThal.sol#1119) is too similar to DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal._transferStandard(address,address,uint256).rTransferAmount (DeGeThal.sol#1119) is too similar to DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal.reflectionFromToken(uint256,bool).rTransferAmount (DeGeThal.sol#831) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal.reflectionFromToken(uint256,bool).rTransferAmount (DeGeThal.sol#831) is too similar to DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal.reflectionFromToken(uint256,bool).rTransferAmount (DeGeThal.sol#831) is too similar to DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal.reflectionFromToken(uint256,bool).rTransferAmount (DeGeThal.sol#831) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal._transferToExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1128) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal.reflectionFromToken(uint256,bool).rTransferAmount (DeGeThal.sol#831) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

Variable DeGeThal._getValues(uint256).rTransferAmount (DeGeThal.sol#902) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (DeGeThal.sol#917) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._transferStandard(address,address,uint256).rTransferAmount (DeGeThal.sol#1119) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._getValues(uint256).rTransferAmount (DeGeThal.sol#902) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

Variable DeGeThal._transferStandard(address,address,uint256).rTransferAmount (DeGeThal.sol#1119) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

Variable DeGeThal.reflectionFromToken(uint256,bool).rTransferAmount (DeGeThal.sol#831) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._getValues(uint256).rTransferAmount (DeGeThal.sol#902) is too similar to DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal._getValues(uint256).rTransferAmount (DeGeThal.sol#902) is too similar to DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal._getValues(uint256).rTransferAmount (DeGeThal.sol#902) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._getValues(uint256).rTransferAmount (DeGeThal.sol#902) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal._transferBothExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1108) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal._transferFromExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1138) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal._transferFromExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1138) is too similar to DeGeThal._transferStandard(address,address,uint256).tTransferAmount (DeGeThal.sol#1119)

Variable DeGeThal._transferFromExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1138) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

Variable DeGeThal._transferBothExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1108) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

Variable DeGeThal._transferToExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1128) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

Variable DeGeThal._transferToExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1128) is too similar to DeGeThal._transferBothExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1108)

Variable DeGeThal._transferToExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1128) is too similar to DeGeThal._transferFromExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1138)

Variable DeGeThal._transferToExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1128) is too similar to DeGeThal._getValues(uint256).tTransferAmount (DeGeThal.sol#901)

Variable DeGeThal._transferBothExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1108) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._transferFromExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1138) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._transferToExcluded(address,address,uint256).rTransferAmount (DeGeThal.sol#1128) is too similar to DeGeThal._transferToExcluded(address,address,uint256).tTransferAmount (DeGeThal.sol#1128)

Variable DeGeThal._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (DeGeThal.sol#917) is too similar to DeGeThal._getTValues(uint256).tTransferAmount (DeGeThal.sol#909)

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

INFO:Detectors:

DeGeThal.slitherConstructorVariables() (DeGeThal.sol#673-1148) uses literals with too many digits:

- _maxTxAmount = 500000 * 10 ** 18 (DeGeThal.sol#706)

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

INFO:Detectors:

DeGeThal._decimals (DeGeThal.sol#692) should be constant

DeGeThal._name (DeGeThal.sol#690) should be constant

DeGeThal._symbol (DeGeThal.sol#691) should be constant

DeGeThal._tTotal (DeGeThal.sol#686) should be constant

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

INFO:Detectors:

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (DeGeThal.sol#435-438)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (DeGeThal.sol#444-448)

geUnlockTime() should be declared external:

- Ownable.geUnlockTime() (DeGeThal.sol#450-452)

lock(uint256) should be declared external:

- Ownable.lock(uint256) (DeGeThal.sol#455-460)

unlock() should be declared external:

- Ownable.unlock() (DeGeThal.sol#463-468)

updateTokenLimit(uint256) should be declared external:
- DeGeThal.updateTokenLimit(uint256) (DeGeThal.sol#752-755)
name() should be declared external:
- DeGeThal.name() (DeGeThal.sol#757-759)
symbol() should be declared external:
- DeGeThal.symbol() (DeGeThal.sol#761-763)
decimals() should be declared external:
- DeGeThal.decimals() (DeGeThal.sol#765-767)
totalSupply() should be declared external:
- DeGeThal.totalSupply() (DeGeThal.sol#769-771)
transfer(address,uint256) should be declared external:
- DeGeThal.transfer(address,uint256) (DeGeThal.sol#778-781)
allowance(address,address) should be declared external:
- DeGeThal.allowance(address,address) (DeGeThal.sol#783-785)
approve(address,uint256) should be declared external:
- DeGeThal.approve(address,uint256) (DeGeThal.sol#787-790)
transferFrom(address,address,uint256) should be declared external:
- DeGeThal.transferFrom(address,address,uint256) (DeGeThal.sol#792-796)
increaseAllowance(address,uint256) should be declared external:
- DeGeThal.increaseAllowance(address,uint256) (DeGeThal.sol#798-801)
decreaseAllowance(address,uint256) should be declared external:
- DeGeThal.decreaseAllowance(address,uint256) (DeGeThal.sol#803-806)
isExcludedFromReward(address) should be declared external:
- DeGeThal.isExcludedFromReward(address) (DeGeThal.sol#808-810)
totalFees() should be declared external:
- DeGeThal.totalFees() (DeGeThal.sol#812-814)
deliver(uint256) should be declared external:
- DeGeThal.deliver(uint256) (DeGeThal.sol#816-823)
reflectionFromToken(uint256,bool) should be declared external:
- DeGeThal.reflectionFromToken(uint256,bool) (DeGeThal.sol#825-834)
excludeFromReward(address) should be declared external:
- DeGeThal.excludeFromReward(address) (DeGeThal.sol#842-850)
excludeFromFee(address) should be declared external:
- DeGeThal.excludeFromFee(address) (DeGeThal.sol#865-867)
includeInFee(address) should be declared external:
- DeGeThal.includeInFee(address) (DeGeThal.sol#869-871)
setSwapAndLiquifyEnabled(bool) should be declared external:
- DeGeThal.setSwapAndLiquifyEnabled(bool) (DeGeThal.sol#887-890)
isExcludedFromFee(address) should be declared external:
- DeGeThal.isExcludedFromFee(address) (DeGeThal.sol#973-975)

Reference:

<https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external>
INFO:Sliether:DeGeThal.sol analyzed (10 contracts with 75 detectors), 117 result(s) found
INFO:Sliether:Use <https://crytic.io/> to get access to additional detectors and Github integration



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io