SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

GDDC

Audit

Security Assessment
01. April, 2023

For

VIRTAL RIDE TOKEN

SolidProof_io @solidproof_io

Disclaimer 3
Description 5
Project Engagement 5
Logo 5
Contract Link 5
Methodology 7
Used Code from other Frameworks/Smart Contracts (direct imports) 8
Tested Contract Files 9
Source Lines 10
Risk Level 10
Capabilities 11
Inheritance Graph 12
CallGraph 13
Scope of Work/Verify Claims 14
Modifiers and public functions 24
Source Units in Scope 26
Critical issues 27
High issues 27
Medium issues 27
Low issues 28
Informational issues 28
Audit Comments 29
SWC Attacks 30

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	29. March 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Network
GDCC

Website
www.gdc.world

Telegram
https://t.me/VirtualRideToken

Twitter
https://twitter.com/gdc_world

Description

VRT is an essential part of the GLOBAL DIGITAL CITY platform and DC are working on establishing key mechanics that make it intrinsically tied to The DC platform and its value. VRT is a GDCC-20 utility token built on the GDCC blockchain that serves as the basis for transactions within GLOBAL DIGITAL CITY.

Project Engagement

During the 29th of March 2023, **GDCC Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- https://www.gdccscan.io/address/0x88849DEE8Fc0c2bd48C8a82329B89F45e8b4715d/contracts#address-tabs
- https://www.gdccscan.io/address/0xE04c1725192aaeE58d66DECAFCB9a72f067DD089/contracts#address-tabs

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

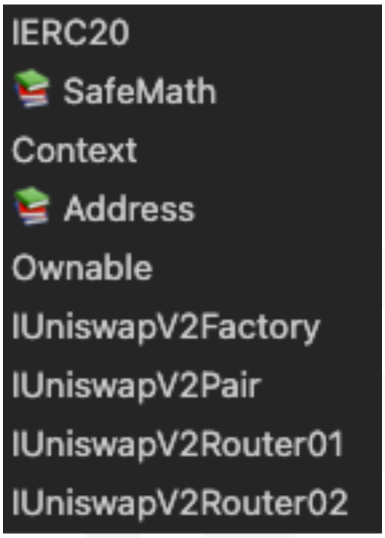
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

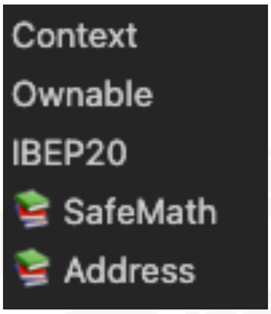
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:



```
IERC20
SafeMath
Context
Address
Ownable
IUniswapV2Factory
IUniswapV2Pair
IUniswapV2Router01
IUniswapV2Router02
```



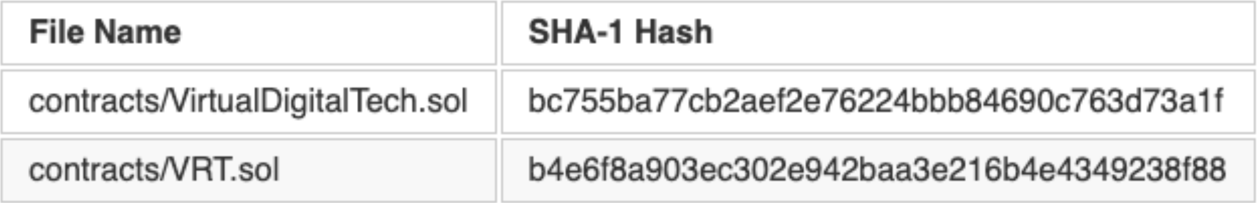
```
Context
Ownable
IBEP20
SafeMath
Address
```

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

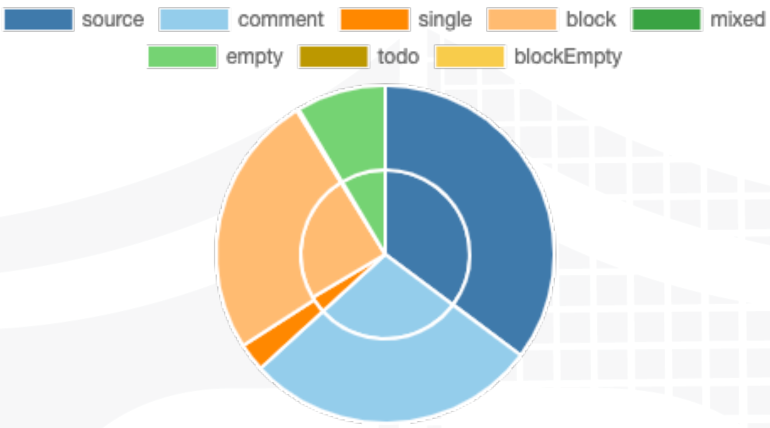


File Name	SHA-1 Hash
contracts/VirtualDigitalTech.sol	bc755ba77cb2aef2e76224bbb84690c763d73a1f
contracts/VRT.sol	b4e6f8a903ec302e942baa3e216b4e4349238f88

Metrics

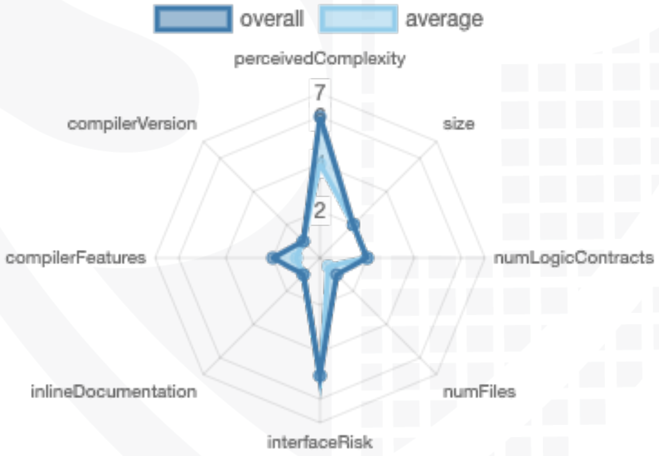
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	4	6	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	127	5

Version	External	Internal	Private	Pure	View
1.0	82	140	23	29	62

State Variables

Version	Total	Public
1.0	34	7

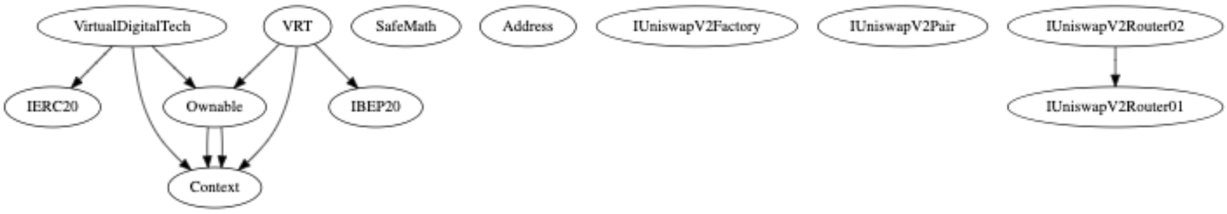
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

1.0	`^0.6.12`
	`0.6.12`
			`yes`	`yes`
				(4 asm blocks)	

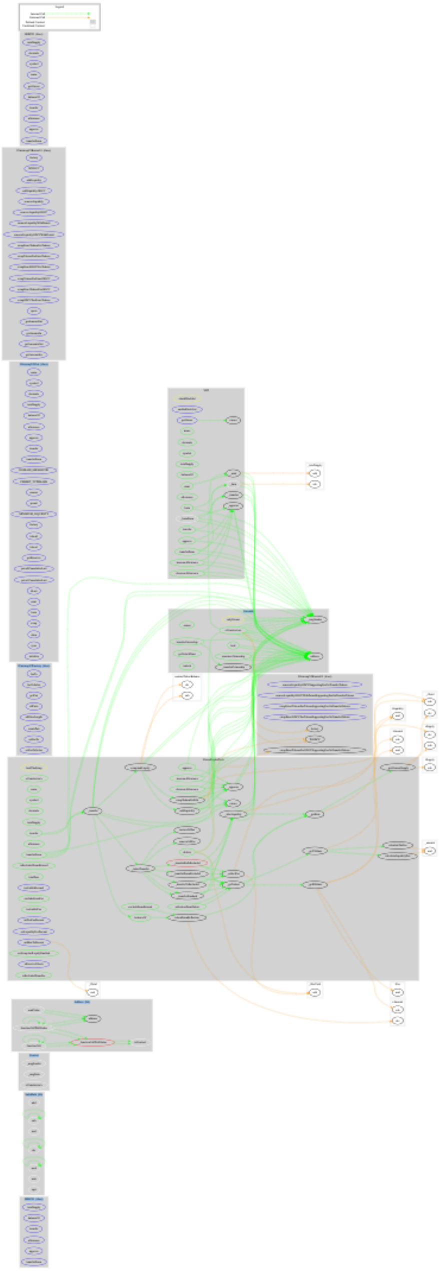
Inheritance Graph

v1.0



CallGraph

v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

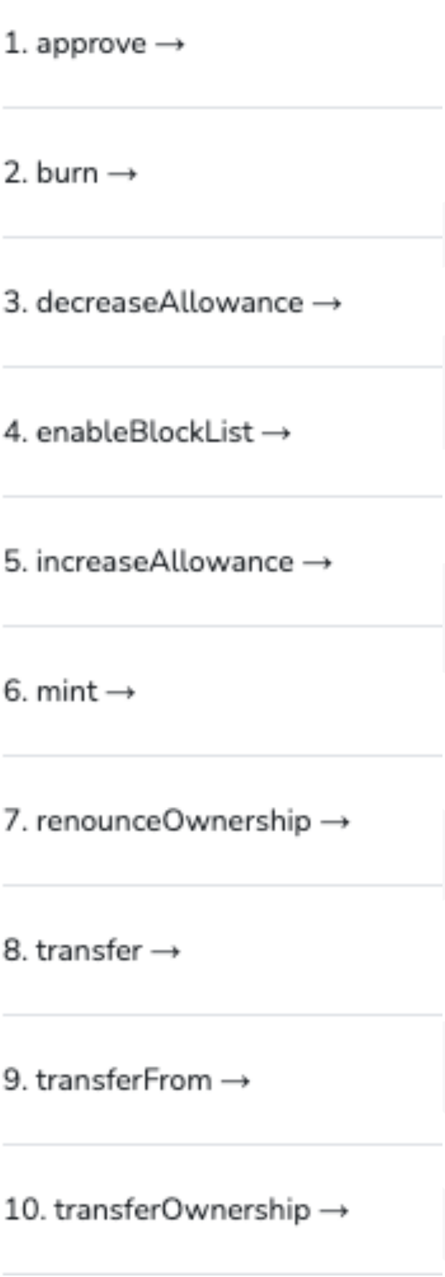
Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

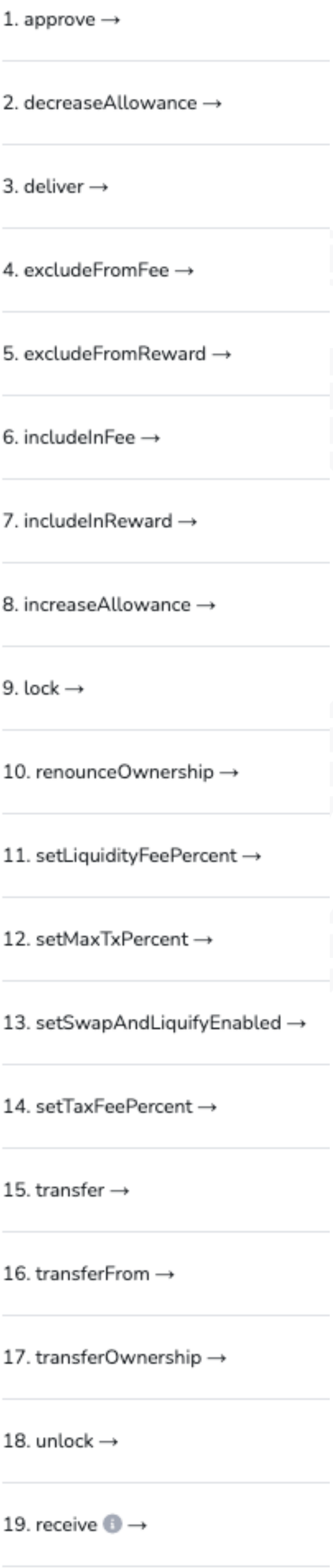
Write functions of contract v1.0

VRT



1. approve →
2. burn →
3. decreaseAllowance →
4. enableBlockList →
5. increaseAllowance →
6. mint →
7. renounceOwnership →
8. transfer →
9. transferFrom →
10. transferOwnership →

Virtual Digital Technology Token



1. approve →
2. decreaseAllowance →
3. deliver →
4. excludeFromFee →
5. excludeFromReward →
6. includeInFee →
7. includeInReward →
8. increaseAllowance →
9. lock →
10. renounceOwnership →
11. setLiquidityFeePercent →
12. setMaxTxPercent →
13. setSwapAndLiquifyEnabled →
14. setTaxFeePercent →
15. transfer →
16. transferFrom →
17. transferOwnership →
18. unlock →
19. receive ⓘ →

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗

Comments:

v1.0

- VRT
  - Owner can mint new tokens in the VRT Token contract to arbitrary addresses without any limitations

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- VirtualDigitalTech
  - Owner can lock user funds by
    - Setting max tx amount to 0
    - Setting max wallet amount to 0

- VRT
  - Tokens
    - can be burned by msg.sender

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✗
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✗

Comments:

v1.0

- VirtualDigitalTech
    - Fees can be set without any limitations

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.0

- VRT
    - Owner can blacklist addresses and lock user funds

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

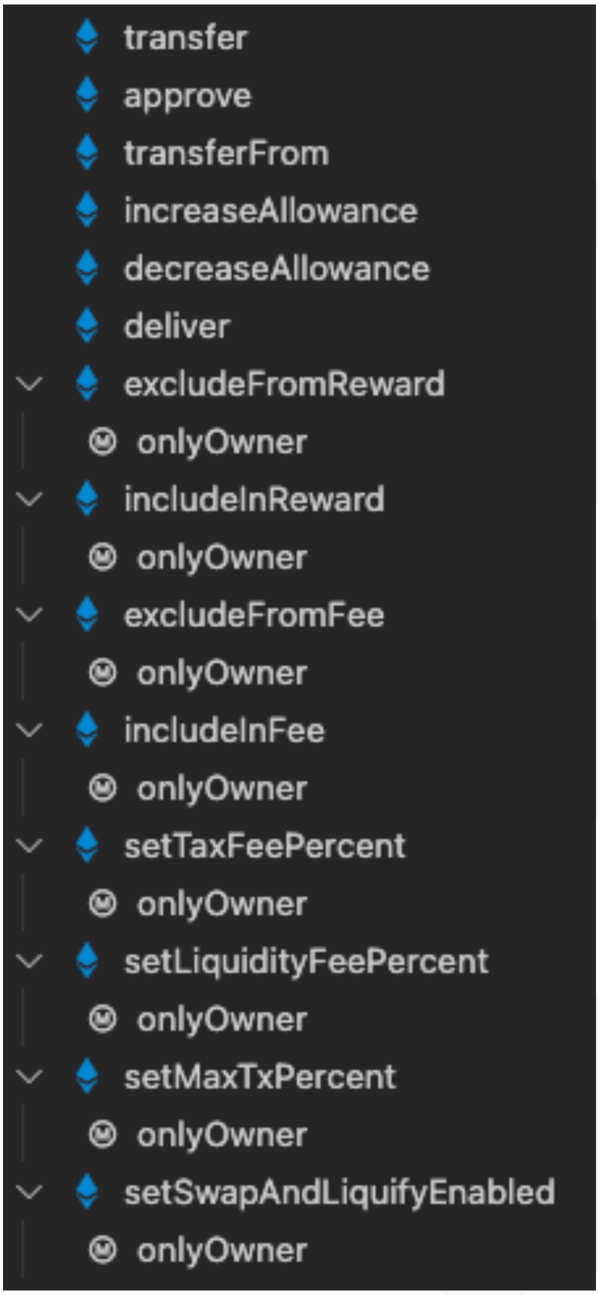
Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	–

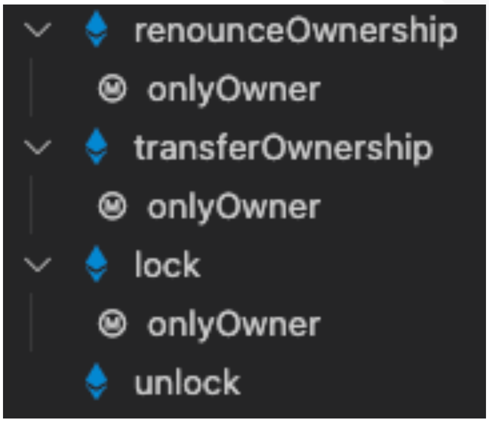
Modifiers and public functions

v1.0

VirtualDigitalTech

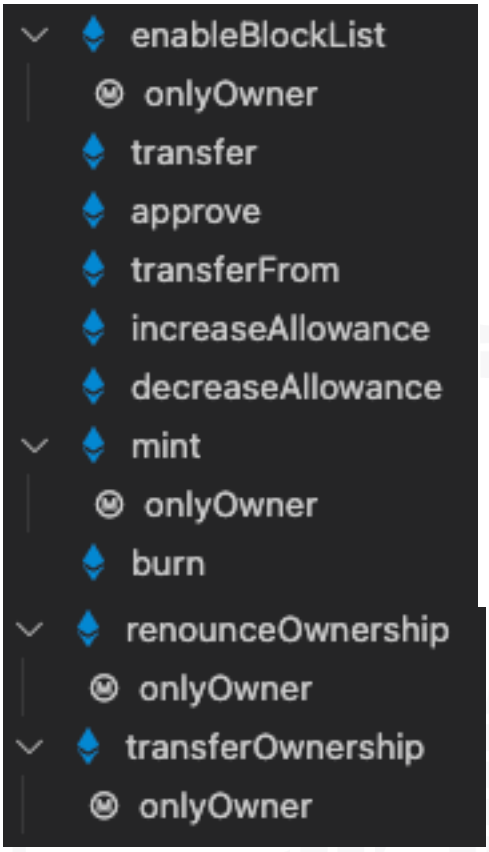


```
transfer
approve
transferFrom
increaseAllowance
decreaseAllowance
deliver
excludeFromReward
  onlyOwner
includeInReward
  onlyOwner
excludeFromFee
  onlyOwner
includeInFee
  onlyOwner
setTaxFeePercent
  onlyOwner
setLiquidityFeePercent
  onlyOwner
setMaxTxPercent
  onlyOwner
setSwapAndLiquifyEnabled
  onlyOwner
```



```
renounceOwnership
  onlyOwner
transferOwnership
  onlyOwner
lock
  onlyOwner
unlock
```

VRT



```
enableBlockList
  onlyOwner
transfer
approve
transferFrom
increaseAllowance
decreaseAllowance
mint
  onlyOwner
burn
renounceOwnership
  onlyOwner
transferOwnership
  onlyOwner
```

Comments

- *Deployer can set following state variables without any limitations*
  - VirtualDigitalTech
    - _maxTxAmount
    - _liquidityFee
    - _taxFee

- *Deployer can enable/disable following state variables*
  - VRT
    - addBlockList
  - VirtualDigitalTech
    - swapAndLiquifyEnabled
    - _isExcludedFromFee
    - _isExcluded
    - _excluded

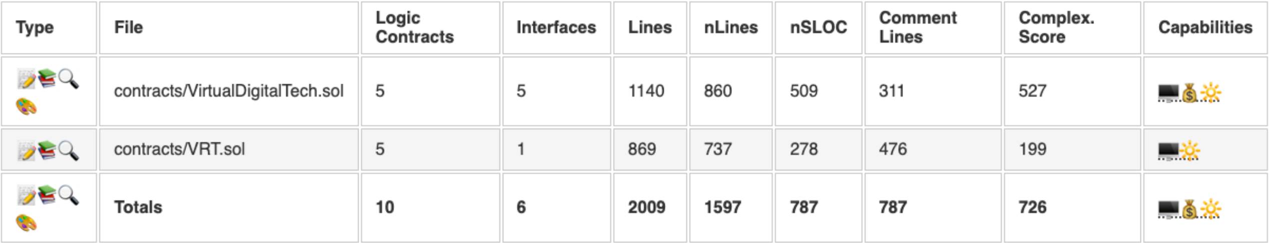
- *Existing Modifiers*
  - VirtualDigitalTech
    - onlyOwner
    - lockTheSwap
  - VRT
    - checkBlacklist

- VRT
  - Owner is able to
    - Mint new tokens without limitations

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope

v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/VirtualDigitalTech.sol	5	5	1140	860	509	311	527	💻💰🔆
📝📚🔍	contracts/VRT.sol	5	1	869	737	278	476	199	💻🔆
📝📚🔍🎨	**Totals**	**10**	**6**	**2009**	**1597**	**787**	**787**	**726**	💻💰🔆

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

Medium issues found

Issue	File	Type	Line	Description
#1	VirtualDigitalTech	Regain ownership	See description	Owner can regain ownership after transferring it with following steps: 1. Call lock function to set _previousOwner to the own address 2. Call unlock function to get ownership back 3. Transfer/renounce ownership 4. Call unlock function to get ownership back Make sure to set the _previousOwnership back to address zero after using the unlock function
#2	VirtualDigitalTech	Fees can be 100% or more	884	The owner can set the fees as 100% or more which may lead to loss of user funds and may lead to some functions to revert due to integer over/underflow
#3	VirtualDigitalTech	Owner can drain liquidity	1074, 1084	The liquidity of the contract is sent to the owner's address directly every time "addLiquidity" function is called.

Low issues

Issue	File	Type	Line	Description
#1	All	Contract doesn’t import npm packages from source (like OpenZeppelin etc.)	—	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	VirtualDigitalTech	A floating pragma is set	—	The current pragma Solidity directive is „“^0.6.12”.
#3	VirtualDigitalTech	Missing Zero Address Validation (missing-zero-check)	835, 845	Check that the address is not zero
#4	All	Old Compiler Version	—	The contracts use a very old compiler version which is not recommended for deployment as it is outdated and is susceptible to knows vulnerabilities.
#5	VRT	Local variables shadowing	832, 673	Rename the local variables that shadow another component
#6	VirtualDigitalTech	Missing Events Arithmetic	835, 845, 868-890	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	VirtualDigitalTech	State variables that could be declared constant (constable-states)	695, 696, 699, 716	Add the `constant` attributes to state variables that never change
#2	VirtualDigitalTech	Unused return values	1075	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

01. April 2023:

- There is still an owner (Owner still has not renounced ownership)
- VirtualDigitalTech
    - Liquidity will be added to the owner. Be aware of it
- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**NOT PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**NOT PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**NOT PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY