SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

DGNX

Audit

Security Assessment

05.October,2022

For

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	03.October,2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Avalanche

Website
https://dgnx.finance/

Twitter
https://twitter.com/degenecosystem

Telegram
https://t.me/degentraderstoken

Instagram
https://instagram.com/degenecosystem

Discord
https://discord.gg/KWX3kmtX

Facebook
https://www.facebook.com/people/Degen-Trader/100078427221036/

TikTok
https://www.tiktok.com/@degen_traders

OpenSea
https://opensea.io/collection/thedegentrader

RaritySniffer
https://raritysniffer.com/viewcollection/degentraders

Description

DegenX is multichain ecosystem that offers a suite of decentralized applications (dApps) and services to provide solutions for projects and individuals in the DeFi space.

Project Engagement

During the 3rd of October 2022, **DGNX** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Links

v1.0

https://github.com/DEGENTOKENTEAM/DGNX/blob/controller-v2/contracts/dgnx/DGNXControllerV2.sol

Commit: d1d311badaf2427590c8a19a689bf6ad054c6e83

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 as possible.
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.
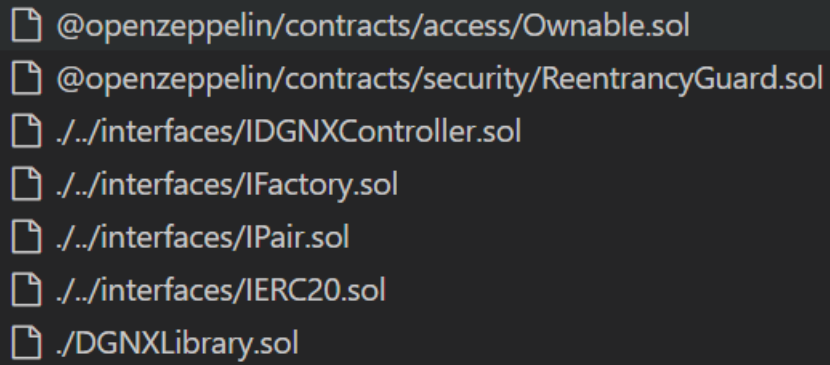
Methodology

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 analyzing 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:

```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol
./../interfaces/IDGNXController.sol
./../interfaces/IFactory.sol
./../interfaces/IPair.sol
./../interfaces/IERC20.sol
./DGNXLibrary.sol
```

Tested Contract Files

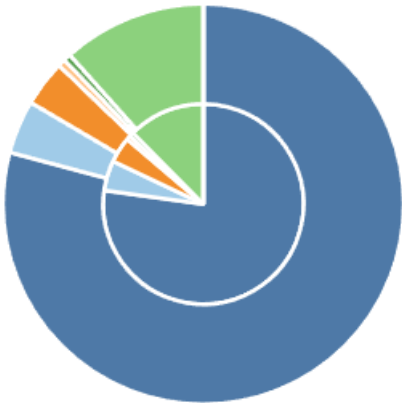This audit covered the following files listed below with a SHA-1 Hash.

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.
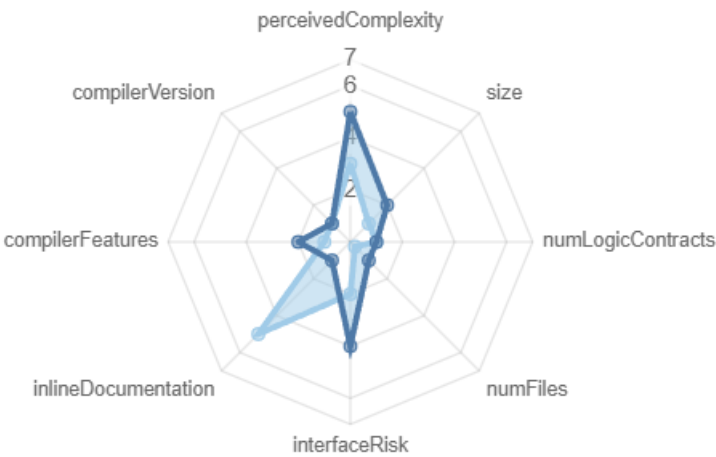
v1.0

File Name	SHA-1 Hash
contracts/interfaces/IPair.sol	a02b34b76c1cc03698334db50edf1a12d0a94e6b
contracts/interfaces/IFactory.sol	f57260fcd3a94464156c241eb4c0d8c0af9ac16b
contracts/interfaces/IERC20.sol	5809a3dad57a1ecd735433bfbe53a12ec8c045eb
contracts/dgnx/DGNXControllerV2.sol	5cbb63163611049d967e74552162ee29712c186c

Metrics

Source Lines
v1.0

Risk Level
v1.0

Capabilities

v1.0

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
1	0	3	0

Exposed Functions

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

🌐 Public	💰 Payable
33	0

External	Internal	Private	Pure	View
25	32	2	0	14

StateVariables

Total	🌐 Public
25	20

Capabilities

Solidity Versions observed	📏 Experimental Features	💰 Can Receive Funds	📋 Uses Assembly	💣 Has Destroyable Contracts
^0.8.4				

📥 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	📇 Uses Hash Functions	⚡ ECRecover	🌀 New/Create/Create2

♻️ TryCatch	Σ Unchecked

Inheritance Graph
v1.0

Call Graph
v1.0
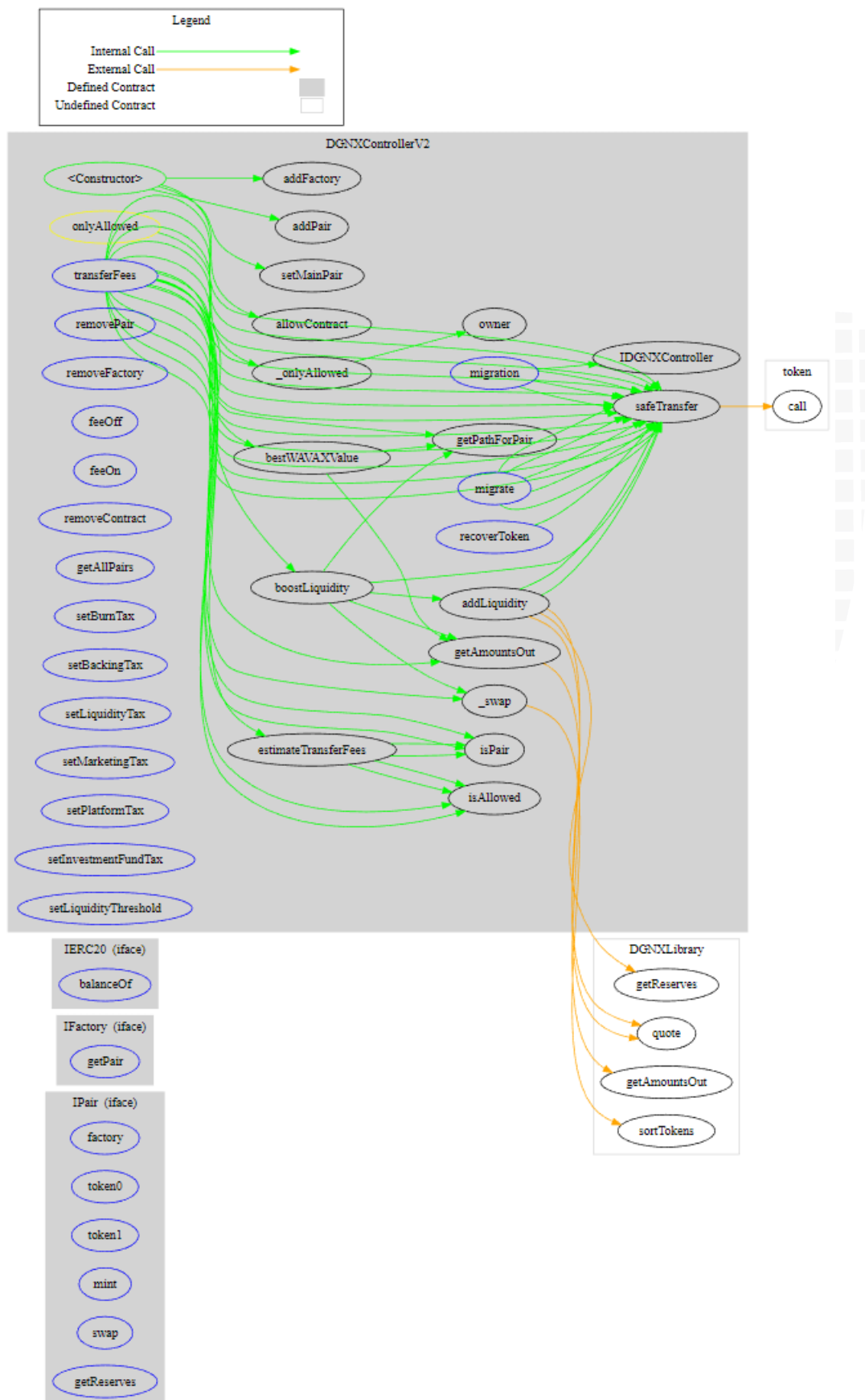
Scope of Work/Verify Claims

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 can set fees
7. Deployer can blacklist/antisnipe address
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	**No**

Comments:
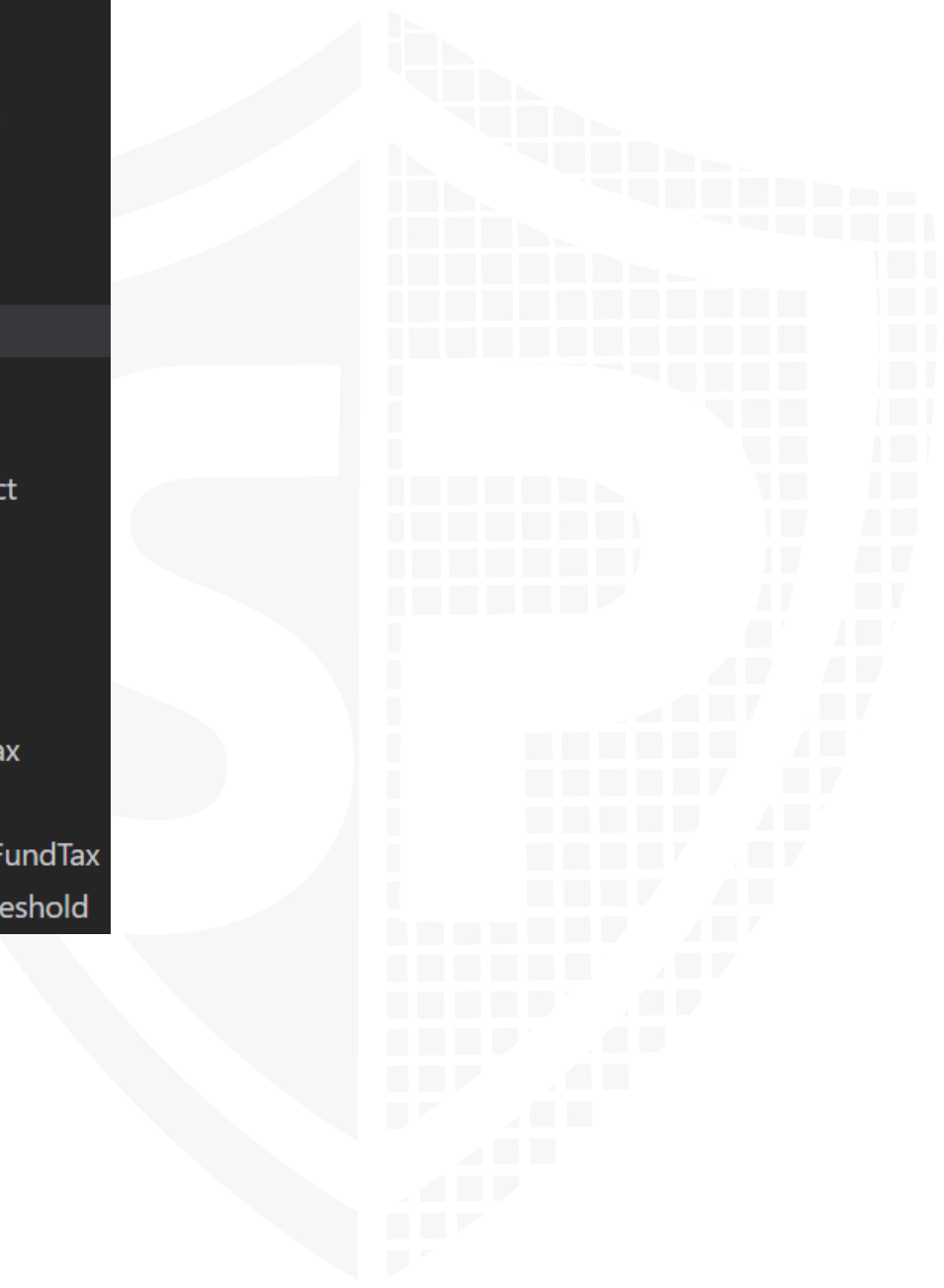
The contract is not an upgradable contract but it can be migrated to other contract along with the funds. But, the funds can also be transferred to a wallet instead of a contract.
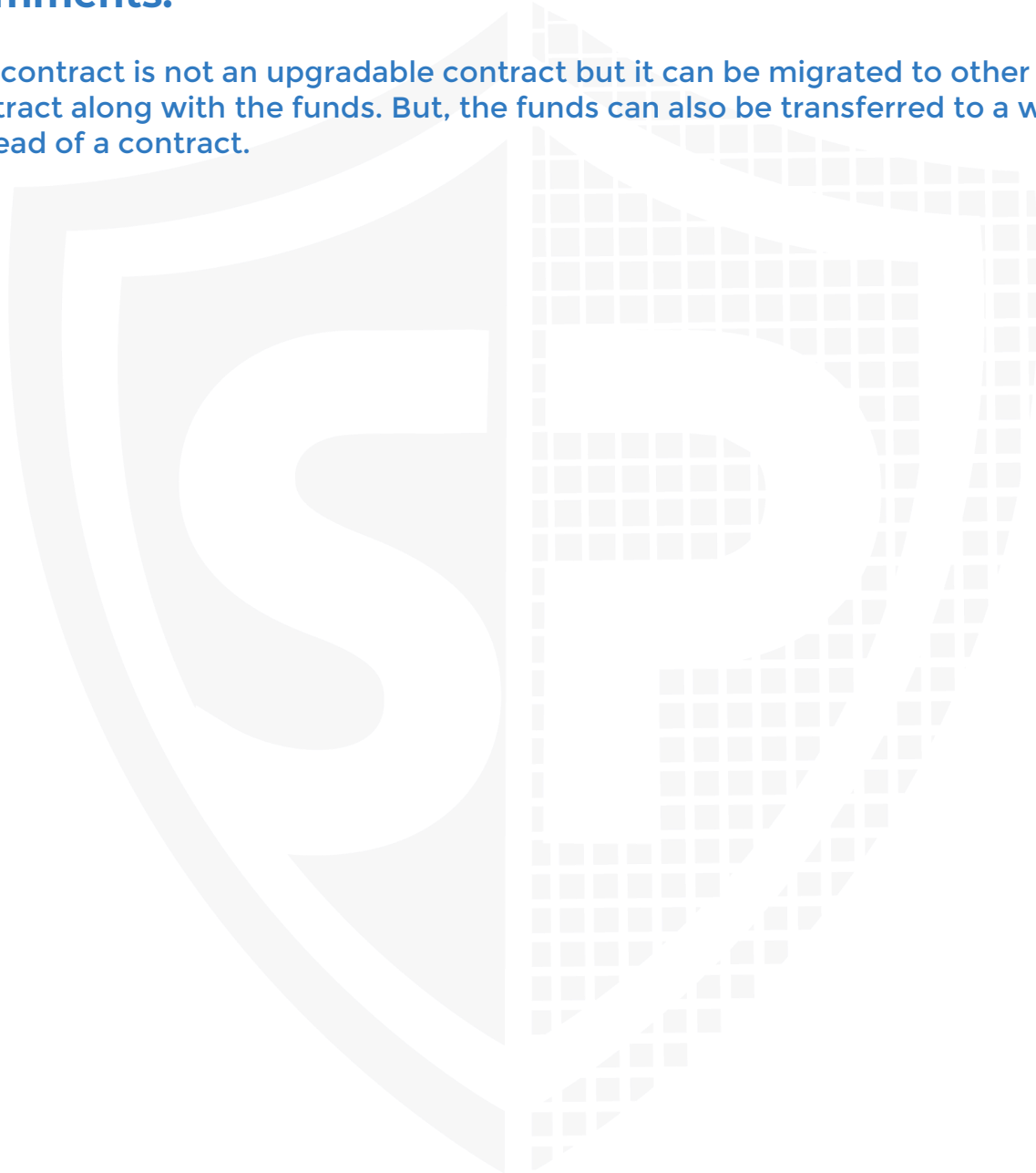
Correct implementation of Token standard

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 contracts
v1.0

- transferFees
- boostLiquidity
- addPair
- removePair
- addFactory
- removeFactory
- feeOff
- feeOn
- migrate
- migration
- recoverToken
- allowContract
- removeContract
- setMainPair
- setBurnTax
- setBackingTax
- setLiquidityTax
- setMarketingTax
- setPlatformTax
- setInvestmentFundTax
- setLiquidityThreshold

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	N/A		

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

Deployer can set fees

Name	Exist	Tested	Status
Deployer can set fees over 25%			
Deployer can set fees to nearly 100% or more			

Deployer cannot blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses			

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

- ♦ transferFees
  - Ⓜ onlyAllowed
- ♦ boostLiquidity
  - Ⓜ onlyAllowed
  - Ⓜ nonReentrant
- ♦ addPair
  - Ⓜ onlyAllowed
- ♦ removePair
  - Ⓜ onlyOwner
- ♦ addFactory
  - Ⓜ onlyOwner
- ♦ removeFactory
  - Ⓜ onlyOwner
- ♦ feeOff
  - Ⓜ onlyAllowed
- ♦ feeOn
  - Ⓜ onlyAllowed
- ♦ migrate
  - Ⓜ onlyAllowed
  - Ⓜ nonReentrant
- ♦ migration
  - Ⓜ onlyAllowed
  - Ⓜ nonReentrant
- ♦ recoverToken
  - Ⓜ onlyAllowed
  - Ⓜ nonReentrant
- ♦ allowContract
  - Ⓜ onlyAllowed
  - Ⓜ nonReentrant
- ♦ removeContract
  - Ⓜ onlyAllowed

- ♦ setMainPair
  - Ⓜ onlyOwner
- ♦ setBurnTax
  - Ⓜ onlyOwner
- ♦ setBackingTax
  - Ⓜ onlyOwner
- ♦ setLiquidityTax
  - Ⓜ onlyOwner
- ♦ setMarketingTax
  - Ⓜ onlyOwner
- ♦ setPlatformTax
  - Ⓜ onlyOwner
- ♦ setInvestmentFundTax
  - Ⓜ onlyOwner
- ♦ setLiquidityThreshold
  - Ⓜ onlyOwner

Comments:

The contract has multiple authorities for some functions. For example, some functions can be called by the owner as well as the contract addresses allowed by the owner. Moreover, the allowed contracts can also allow/disallow other contracts.
There are no Non-Owner functions in the contract.

Owner Privileges:

- Transfer of fees, boosting of liquidity
- Add/Remove Pair and Factory contracts
- Turn the fees on/off
- Migrate to a new contract and transfer all the funds either to a new contract or to a wallet also because there is no check against it while migration is in progress
- Allow/Disallow contracts to be in the onlyAllowed modifier
- Set and update tax but not more than 5%
- Set Liquidity Threshold without any limit or Range

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/interfaces/IPair.sol		1	28	5	3	1	13
contracts/interfaces/IFactory.sol		1	9	5	3	1	3
contracts/interfaces/IERC20.sol		1	6	5	3	1	3
contracts/dgnx/DGNXControllerV2.sol	1		746	680	563	33	430
Totals	1	3	789	695	572	36	449

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized 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

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Main	Missing Zero Check	143	Check that the address is not zero
#2	Main	Missing Events	569,597,143	Emit events for Critical parameter changes
#3	Main	Owner can drain tokens	569,597	Owner can pass the address of any arbitrary contract or wallet address in the migration functions to transfer all the funds.
#4	Main	Floating Pragma	-	The current pragma Solidity directive is "^0.8.4". Contracts should be deployed with the same compiler version and flag that they have been tested thoroughly. Locking the pragma

Issue	File	Type	Line	Description
				helps to ensure that contracts do not accidentally get deployed using other versions.

Informational issues

Issue	File	Type	Line	Description
#1	Main	Natspec Missing	–	If you start to comment your code then comment out properly and thoroughly.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/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.

05. October, 2022:

- There is still an owner (Owner still has not renounced ownership).
- The ownership can never be renounced from this contract because there is no function for that.
- The Owner can change multiple addresses and basically all the parameters in the contract hence, the contract is completely centralized and everything is controlled by the owner and some functions by the contracts authorized by the Owner.
- Read the 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-	Presence of unused variables	CWE-1164: Irrelevant Code	**PASSED**

1 3 1			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-	Write to Arbitrary	CWE-123: Write-what-where Condition	PASSED

[1 2 4](#)	Storage Location		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](#)	**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**

<u>S W C - 1 0 6</u>	Unprotected SELFDESTRUCT Instruction	[CWE-284: Improper Access Control](#)	**PASSED**
<u>S W C - 1 0 5</u>	Unprotected Ether Withdrawal	[CWE-284: Improper Access Control](#)	**PASSED**
<u>S W C - 1 0 4</u>	Unchecked Call Return Value	[CWE-252: Unchecked Return Value](#)	**PASSED**
<u>S W C - 1 0 3</u>	Floating Pragma	[CWE-664: Improper Control of a Resource Through its Lifetime](#)	**NOT PASSED**
<u>S W C - 1 0 2</u>	Outdated Compiler Version	[CWE-937: Using Components with Known Vulnerabilities](#)	**PASSED**
<u>S W C - 1 0 1</u>	Integer Overflow and Underflow	[CWE-682: Incorrect Calculation](#)	**PASSED**

[SWC-100](#)	Function Default Visibility	[CWE-710: Improper Adherence to Coding Standards](#)	**PASSED**