SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

DegenX Liquidity Backing

Audit

Security Assessment
16. May, 2023

For

SolidProof_io

@solidproof_io

Disclaimer 3
Description 5
Project Engagement 5
Logo 5
Contract Links 5
Methodology 7
Used Code from other Frameworks/Smart Contracts (direct imports) 8
Tested Contract Files 9
Source Lines 11
Risk Level 11
Capabilities 12
Inheritance Graph 13
CallGraph 14
Scope of Work/Verify Claims 15
Modifiers and public functions 17
Source Units in Scope 19
Critical issues 20
High issues 20
Medium issues 20
Low issues 20
Informational issues 20
Audit Comments 20
SWC Attacks 21

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	17. April 2023 - 19. April 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	20. April 2023	• Reaudit

Network
Avalanche

Website
https://dgnx.finance/

Telegram
https://t.me/DegenXportal

Twitter
https://twitter.com/degenecosystem

Discord
https://discord.com/invite/pyaZqZrS

Facebook
https://www.facebook.com/profile.php?id=100078427221036

Reddit
https://www.reddit.com/user/degentrader_sd

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.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 15th of April 2023, **DGNX 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 Links

v1.0

https://github.com/DEGENTOKENTEAM/liquidity-backing

Commit: bc7dc39

v1.1

https://github.com/DEGENTOKENTEAM/liquidity-backing

Commit: d6b60eaf1cd791bc4d370e5b2a2c09ddd4886b10

Note- The FeeManager contract was removed from the repo in v1.1

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.

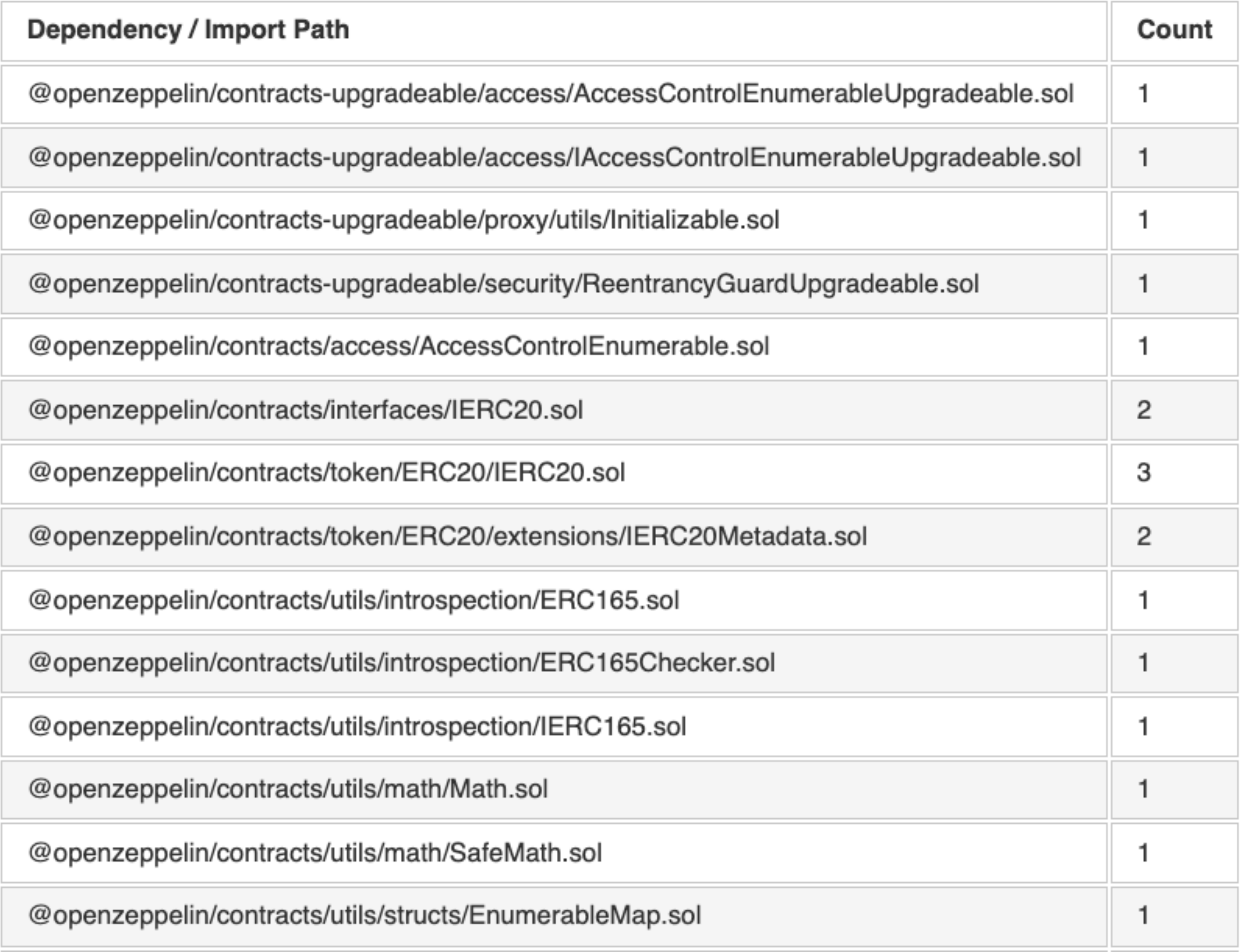
2. Testing and automated analysis that includes the following:
   i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii) Symbolic execution, which is analysing a program to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



Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/AccessControlEnumerableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/access/IAccessControlEnumerableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts/access/AccessControlEnumerable.sol	1
@openzeppelin/contracts/interfaces/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	3
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	2
@openzeppelin/contracts/utils/introspection/ERC165.sol	1
@openzeppelin/contracts/utils/introspection/ERC165Checker.sol	1
@openzeppelin/contracts/utils/introspection/IERC165.sol	1
@openzeppelin/contracts/utils/math/Math.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
@openzeppelin/contracts/utils/structs/EnumerableMap.sol	1

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.

v1.1

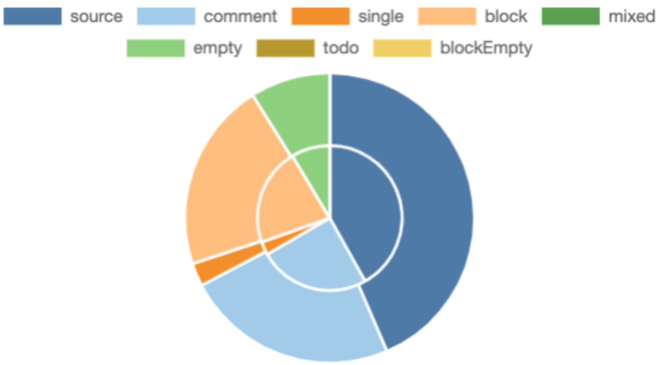
File Name	SHA-1 Hash
contracts/interfaces/IController.sol	a5c0529e0c3b62d9c80ef33977f032b36379b453
contracts/interfaces/IPair.sol	d199631b71eaceb9187c70db145c04a24289c1ba
contracts/interfaces/ISwapper.sol	a0b310eb465ae65b960ddba5a41a8555ba219660
contracts/interfaces/IL1Wrapper.sol	28df279aac3158f75e858c0ecb14e509b0aa9740
contracts/interfaces/IRouter.sol	79936d719724f0172750cb610dc3b97b8c78b767
contracts/interfaces/IFeeManager.sol	6ef41e4f01e2337d0ed9c8a586f1732b71030fc2
contracts/interfaces/IStrategy.sol	69e1ed9c9224649cb5772781b14d42e5e5cc2837
contracts/interfaces/IFactory.sol	d3ae4693af2ce2df195a75798f2cba7a10669249
contracts/interfaces/IVault.sol	c46fbdd3439b396041849c5874648940f88cf5a7
contracts/interfaces/IAggregator.sol	3894a8a876ea1698902253b7f6a0b6c7b7b07eb2
contracts/Aggregator.sol	82833f7c05f26387712f638fbb093352721775e1

contracts/vaults/VaultBase.sol	b553942e9ef9b6389171a6dedbc6015c3aa66fe4
contracts/vaults/ERC20Vault.sol	90b2c54af8adb5244b5dcc171116c30de3c2ac40
contracts/BaseRoles.sol	c63ddcc68f38ff0a29e5e62511d60bf1ace52a19
contracts/Base.sol	dc7a2274d39fb70413b5b0d2106899738177a234
contracts/libraries/Helper.sol	9848856737ebb1f55486edec6c3691be8b37b759
contracts/libraries/SwapLibrary.sol	90a8bef242a21810c560d056f02e113bab97393f
contracts/strategies/StrategyTokenToWAVAX.sol	a1fc2fe931d480b817f0a5fec31ea34cf7601bce
contracts/strategies/StrategyBase.sol	afb5eaa612ecd3842d60760a320cfe9d32cbaec1
contracts/strategies/StrategyTokenToToken.sol	566f77c3f150aa63307924df490706209026934d
contracts/BaseUpgradeable.sol	a2bbd7f9aefa2c68506c0ac30f9e8a338c83d89f
contracts/Controller.sol	ec1959b719a9f78925d019590815d4c507abfb89
contracts/swappers/SwapperWAVAX.sol	f17de88c1baa09b356ab31d60da20c1f8e7b05d1
contracts/swappers/SwapperBase.sol	f32e169d593eaa392ed15c7b4434be0329cfc2e8
contracts/swappers/SwapperProxy.sol	5f834205894551c8493c9799eb70a94bac183169

Metrics

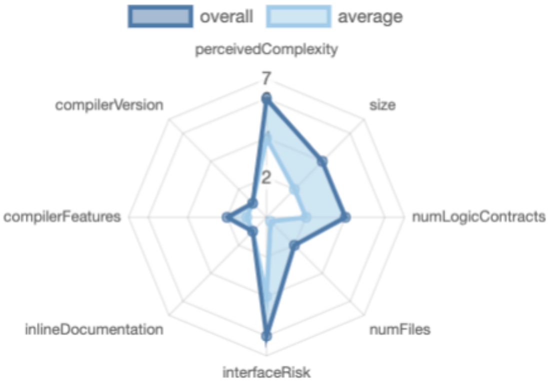
Source Lines

v1.0



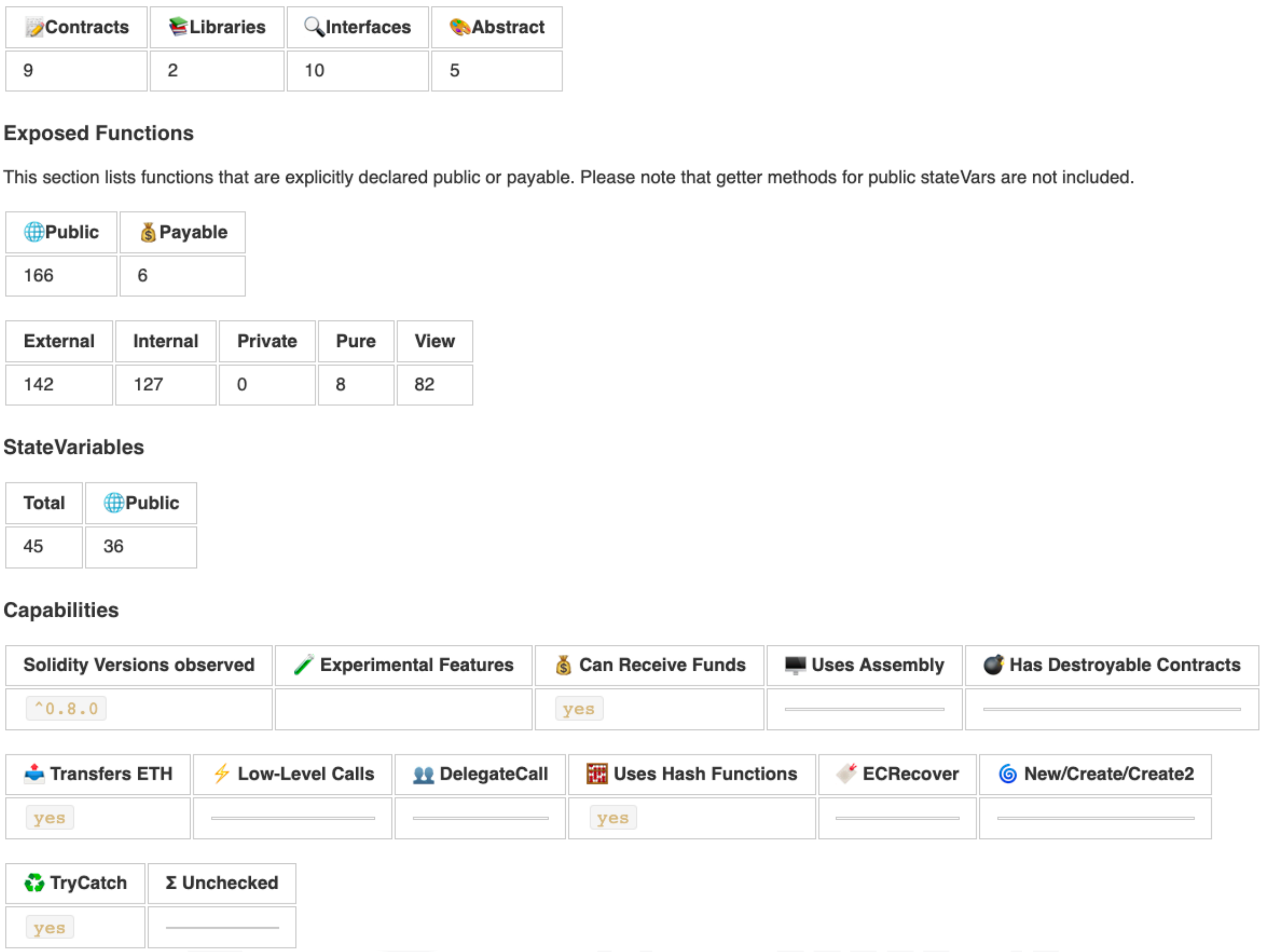
Risk Level

v1.0



Capabilities

Components



📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
9	2	10	5

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
166	6

External	Internal	Private	Pure	View
142	127	0	8	82

StateVariables

Total	🌐Public
45	36

Capabilities

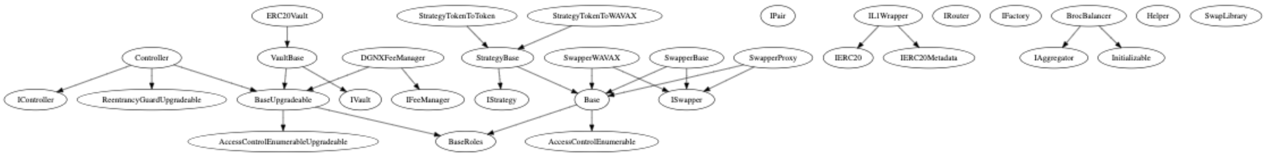
Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.0`		`yes`		

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🏷 ECRecover	🌀 New/Create/Create2
`yes`			`yes`		

♻️ TryCatch	Σ Unchecked
`yes`	

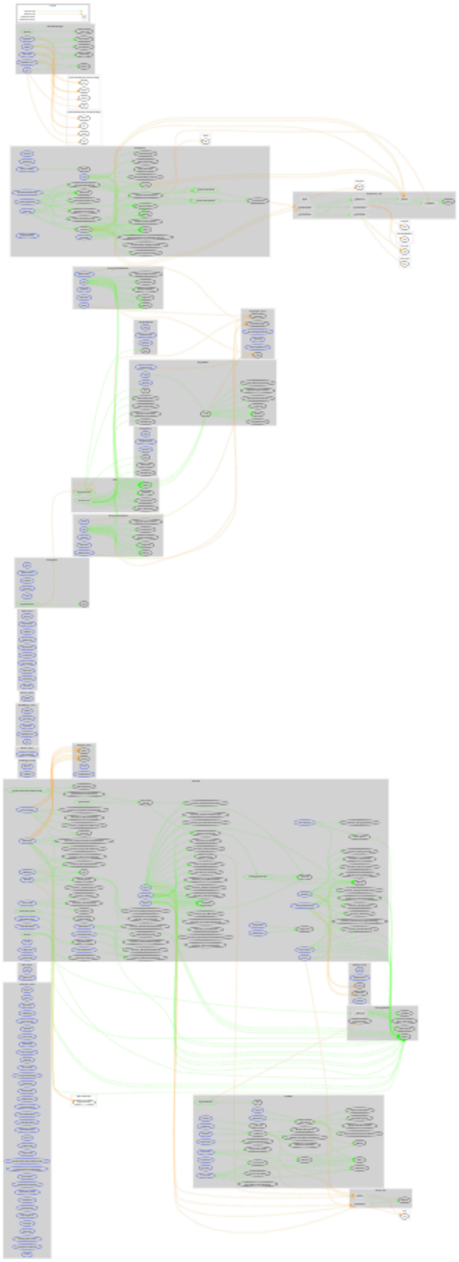
Inheritance Graph

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. Overall checkup (Smart Contract Security)

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

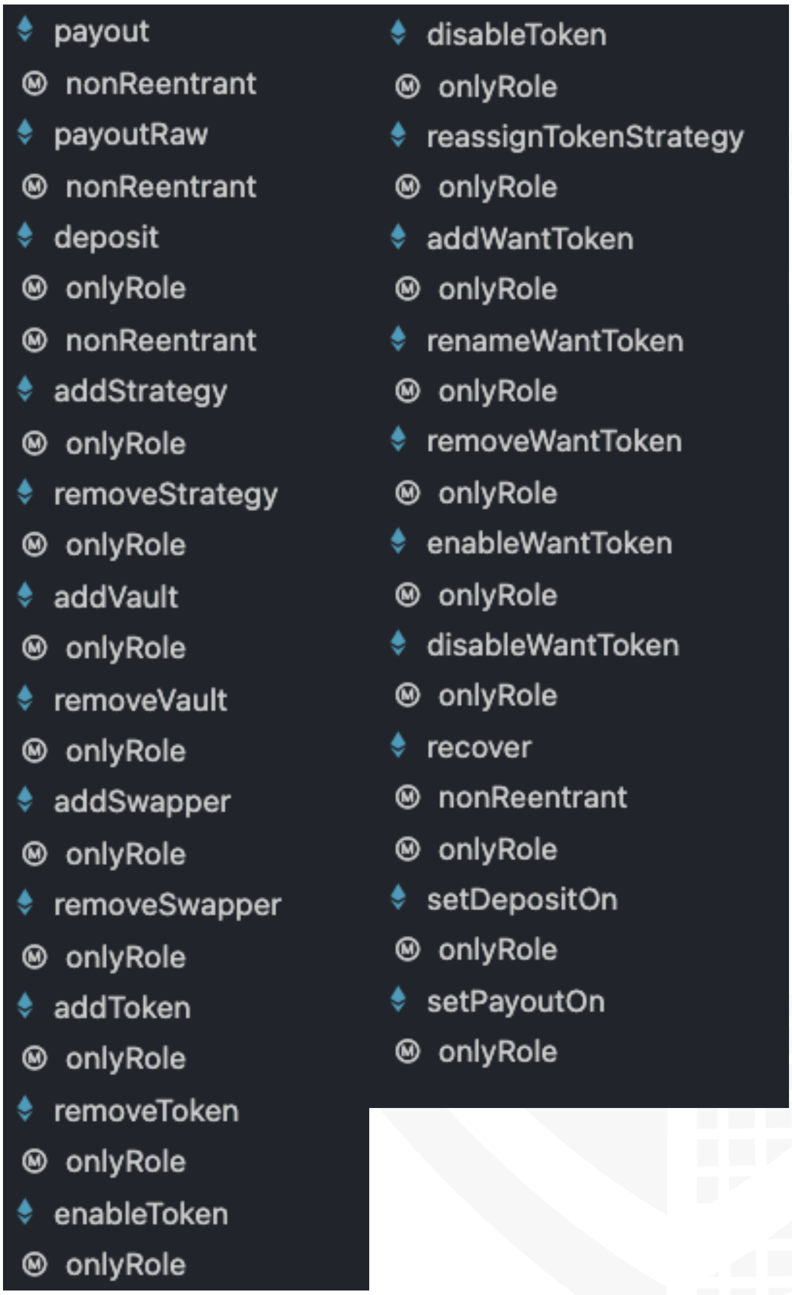
Legend

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

Modifiers and public functions

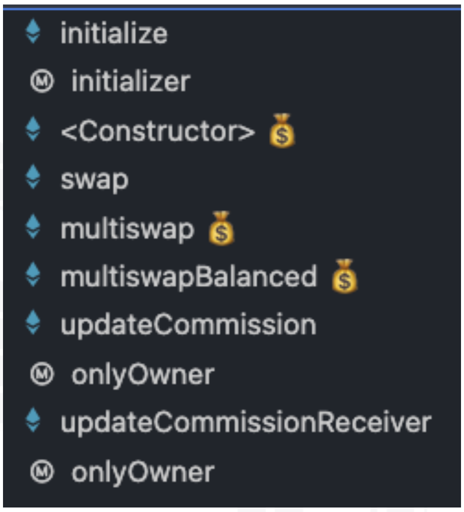
v1.1

Controller



♦ payout
Ⓜ nonReentrant
♦ payoutRaw
Ⓜ nonReentrant
♦ deposit
Ⓜ onlyRole
Ⓜ nonReentrant
♦ addStrategy
Ⓜ onlyRole
♦ removeStrategy
Ⓜ onlyRole
♦ addVault
Ⓜ onlyRole
♦ removeVault
Ⓜ onlyRole
♦ addSwapper
Ⓜ onlyRole
♦ removeSwapper
Ⓜ onlyRole
♦ addToken
Ⓜ onlyRole
♦ removeToken
Ⓜ onlyRole
♦ enableToken
Ⓜ onlyRole
♦ disableToken
Ⓜ onlyRole
♦ reassignTokenStrategy
Ⓜ onlyRole
♦ addWantToken
Ⓜ onlyRole
♦ renameWantToken
Ⓜ onlyRole
♦ removeWantToken
Ⓜ onlyRole
♦ enableWantToken
Ⓜ onlyRole
♦ disableWantToken
Ⓜ onlyRole
♦ recover
Ⓜ nonReentrant
Ⓜ onlyRole
♦ setDepositOn
Ⓜ onlyRole
♦ setPayoutOn
Ⓜ onlyRole

Aggregator



♦ initialize
Ⓜ initializer
♦ <Constructor> 💰
♦ swap
♦ multiswap 💰
♦ multiswapBalanced 💰
♦ updateCommission
Ⓜ onlyOwner
♦ updateCommissionReceiver
Ⓜ onlyOwner

Ownership/Authority Privileges

The address/wallet with the “ROLE_GOVERNANCE” have the following privileges

- Controller.sol
  - *Deposit a token in the vault*
  - *Add/Change/Remove strategy, vault, and swapper addresses*

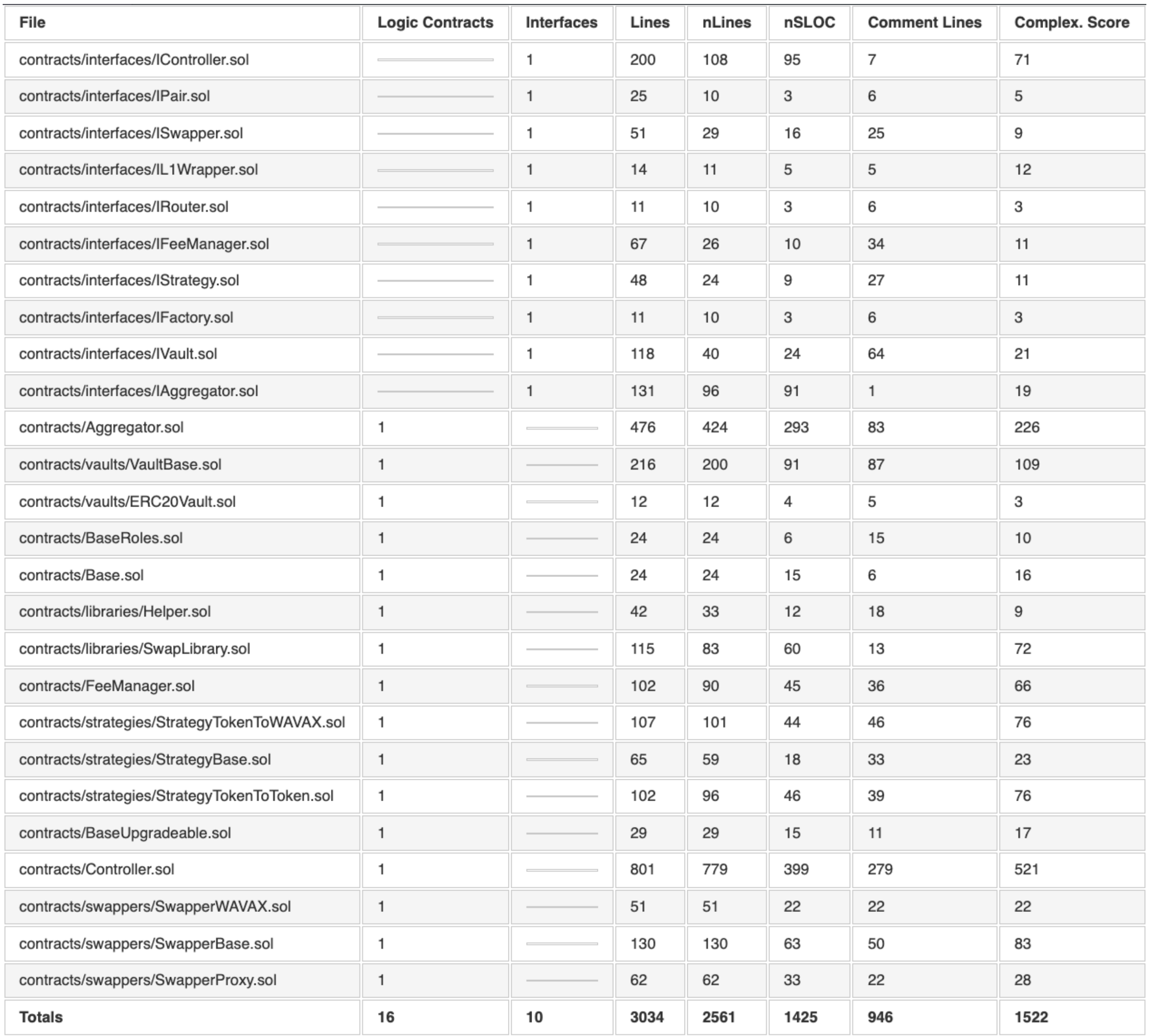
  - *Add/Remove a new token address that can be deposited into the vault*
  - *Enable/Disable a token for being deposited*
  - *Reassign a token to another strategy for swapping*
  - *Add/Change/Remove Payout token*
  - *Enable/Disable payout tokens*
  - *Recover/Withdraw any token from the contract's balance*
  - *Recover/Withdraw unregistered tokens from the current vault contract's balance*
  - *Enable/Disable deposit and Payout*

- Aggregator.sol
  - *Update commission receiver address*
  - *Update commission up to 50% (recommended limit is 25 so do your own research here)*

The write functions in the Swapper and Vault contracts can only be called by the 'Controller contract'
Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope

v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/interfaces/IController.sol		1	200	108	95	7	71
contracts/interfaces/IPair.sol		1	25	10	3	6	5
contracts/interfaces/ISwapper.sol		1	51	29	16	25	9
contracts/interfaces/IL1Wrapper.sol		1	14	11	5	5	12
contracts/interfaces/IRouter.sol		1	11	10	3	6	3
contracts/interfaces/IFeeManager.sol		1	67	26	10	34	11
contracts/interfaces/IStrategy.sol		1	48	24	9	27	11
contracts/interfaces/IFactory.sol		1	11	10	3	6	3
contracts/interfaces/IVault.sol		1	118	40	24	64	21
contracts/interfaces/IAggregator.sol		1	131	96	91	1	19
contracts/Aggregator.sol	1		476	424	293	83	226
contracts/vaults/VaultBase.sol	1		216	200	91	87	109
contracts/vaults/ERC20Vault.sol	1		12	12	4	5	3
contracts/BaseRoles.sol	1		24	24	6	15	10
contracts/Base.sol	1		24	24	15	6	16
contracts/libraries/Helper.sol	1		42	33	12	18	9
contracts/libraries/SwapLibrary.sol	1		115	83	60	13	72
contracts/FeeManager.sol	1		102	90	45	36	66
contracts/strategies/StrategyTokenToWAVAX.sol	1		107	101	44	46	76
contracts/strategies/StrategyBase.sol	1		65	59	18	33	23
contracts/strategies/StrategyTokenToToken.sol	1		102	96	46	39	76
contracts/BaseUpgradeable.sol	1		29	29	15	11	17
contracts/Controller.sol	1		801	779	399	279	521
contracts/swappers/SwapperWAVAX.sol	1		51	51	22	22	22
contracts/swappers/SwapperBase.sol	1		130	130	63	50	83
contracts/swappers/SwapperProxy.sol	1		62	62	33	22	28
Totals	**16**	**10**	**3034**	**2561**	**1425**	**946**	**1522**

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

No medium issues

Low issues

No low issues

Informational issues

No informational issues

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.

16. May 2023:

- Read whole report and modifiers section for more information
- Some of the contracts are upgradeable and they can be deployed again by the owner with new privileges and functionality
- The paidOut in the controller contract is checked for only 1 which means if there is an array which has only 1 paid out it will forever be true because it is not checking for every element in the iteration.
- Do your own research here

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	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	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	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