



SOLIDProof
Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

Darwin Protocol

Audit

Security Assessment
12. April, 2023

For



**DARWIN
PROTOCOL**



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	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	23
Source Units in Scope	25
Critical issues	26
High issues	26
Medium issues	26
Low issues	26
Informational issues	26
Audit Comments	26
SWC Attacks	27

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 — 31. March 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	12. April 2023	<ul style="list-style-type: none">• Reaudit

Network

Arbitrum

Website

<https://darwinprotocol.io/>

Telegram

<https://t.me/DarwinProtocol>

Twitter

https://twitter.com/Darwin_Protocol

Facebook

<https://www.facebook.com/darwinprotocol>

LinkedIn

<https://www.linkedin.com/company/darwin-tech-ltd/>

Description

Darwin Protocol is the introduction of redistributing funds during a trade without taking a portion of the user's tokens to do so. This unique contract taxes the liquidity pool of the token on the decentralized exchange (DEX) during swaps instead of users, allowing it to collect both \$DARWIN and ETH! An ERC20 token \$DARWIN is built on the Arbitrum One Chain with a launch supply of 150 Million tokens (increasing to 1 billion through staking and farming). It is decentralized with the token contract and the liquidity pool 'LP' tokens being owned and controlled by a DAO, DarwinCommunity

Project Engagement

During the Date of 28 March 2023, **Darwin Protocol 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://github.com/Darwin-Coin/darwin-token-contracts>
- **Commit:** 0b21c24

v1.0

- <https://github.com/Darwin-Coin/darwin-token-contracts>
- **Commit:** 6b91adf5bc634d736b0bcebd129478e0a84fcc33

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 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/AccessControlUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	1
@openzeppelin/contracts/access/AccessControl.sol	1
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/security/ReentrancyGuard.sol	4
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC721/ERC721.sol	3
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	2
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.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.

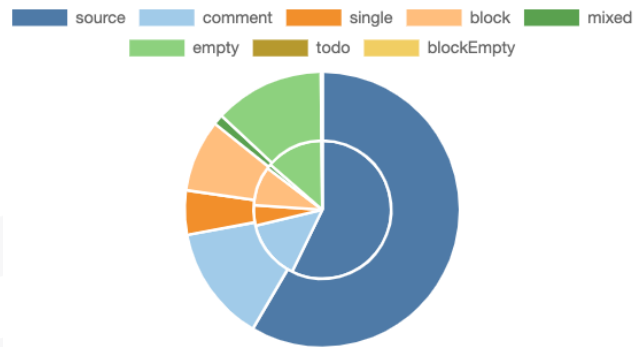
v1.0

File Name	SHA-1 Hash
contracts/Openzeppelin/ERC20Upgradeable.sol	d9c748e8b426aa2043f140d6bc120c2a1d449a00
contracts/DarwinBurner.sol	3a907cbb79b8a3ab1cb4d4657949fafa235cbb04
contracts/Darwin.sol	80fd0c8f0d8826991855135ad6f86abc9e5d2339
contracts/EvoturesNFT.sol	258f9989439bf4437d1bbbcc6de37365e6591875
contracts/DarwinVester5.sol	bcb5f7aa43a47c0488e4067cac7e58deeeceaf8
contracts/DarwinVester7.sol	fd56c41f308cc55af54edf49cb6d78b2ad5d0315
contracts/LootboxTicket.sol	1efccc7972647fff5b9a5e0e1f8c12933cf64865
contracts/DarwinStaking.sol	2d050685c7bd42d0351209e6dc4d9895871109ee
contracts/StakedDarwin.sol	5cd080ec8e0a530071a725c9f44123d52dbf0c5c
contracts/DarwinCommunity.sol	f831e5a2e8b94f5f1ec4be3aab0208fc6553d7cd
contracts/interface/IDarwinVester.sol	34ee9422aeb4d9185150ca4f9ea115687162fb0b

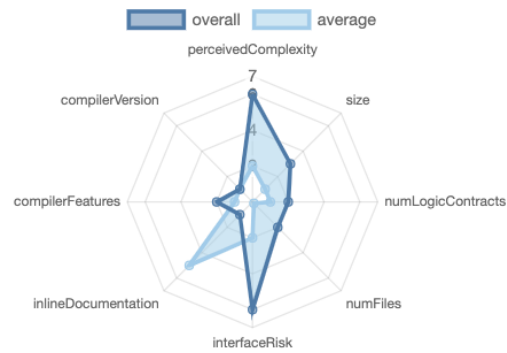
contracts/interface/ IEvoturesNFT.sol	f46b0d12889df0e8f3a2dc1bd246f1 220e9c8fd0
contracts/interface/IDarwin.sol	bf0688808abb5cd7a92fbed9bc920 e2ba7b495b2
contracts/interface/ IDarwinStaking.sol	b15f65f405180068ed0855d73a88b 14b442e5f67
contracts/interface/ IStakedDarwin.sol	86dbba7cfd30d1212acde4711452 3a892e9615fc
contracts/interface/IERC20.sol	c8b0080678fdb94c4aa4efd5b7d43 0c888b258f6
contracts/interface/ ILootboxTicket.sol	4fa753857d1a54c39a49b737dcc2 42efdfee5b4f
contracts/interface/ IDarwinCommunity.sol	53280f7845ab4732d15c65be75c6 5c0c45dbde63

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
10	0	10	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
150	1







External	Internal	Private	Pure	View
127	170	10	8	47


StateVariables

Total	 Public
102	73

Capabilities

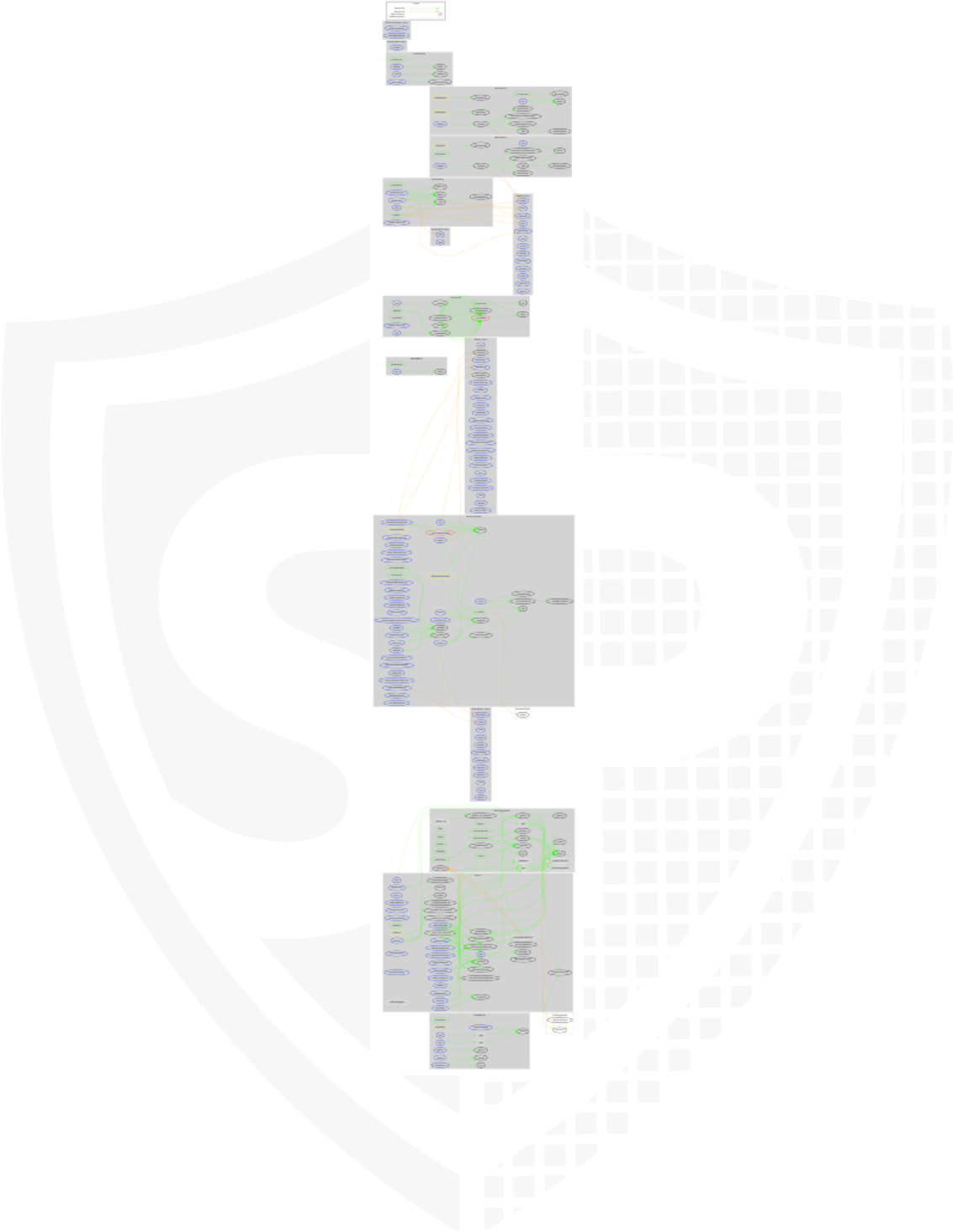
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<div><div>^0.8.0</div><div>0.8.14</div><div>^0.8.14</div></div>		<div>yes</div>	<div>yes</div> <div>(4 asm blocks)</div>	

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECREcover	 New/Create/Create2
<div>yes</div>			<div>yes</div>		<div>yes</div> <div>→ <code>AssemblyCall:Name:create2</code></div>

 TryCatch	 Σ Unchecked
	<div>yes</div>

Inheritance Graph





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. Authorities cannot mint any new tokens
4. Authority cannot burn or lock user funds
5. Community cannot pause the contract
6. Authority cannot blacklist/antisnipe addresses
7. Overall checkup (Smart Contract Security)



Is contract an upgradeable

Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

- A new version of the contracts can be deployed by the address/wallet that have the UPGRADER_ROLE privileges which can change any limit and may introduce new privileges in the updated contract.
 - Be aware of this and do your own research for the contract which the proxy contract is pointing to

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	✓	✓	✓

Authority cannot mint any new tokens

Name	Exist	Tested	Status
Authority cannot mint	✓	✓	✗
Max / Total Supply	1.000.000.000		

Comments:

v1.0

- Addresses with Minter role can mint tokens but not more than max supply
- The deployer address of the "EvoturesNFT" contract can mint any amount of tokens with any rarity in the loot box contract
- darwinStaking address in the "StakedDarwin" contract which will be set by the deployer, can mint unlimited tokens

Authority cannot burn or lock user funds

Name	Exist	Tested	Status
Authority cannot lock	—	—	—
Authority cannot burn	✓	✓	✗

Comments:

v1.0

- darwinStaking address in the "StakedDarwin" contract which will be set by the deployer, can burn tokens from any arbitrary address.

Community cannot pause the contract

Name	Exist	Tested	Status
Authority can pause	✓	✓	✗



Authority can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Authority can blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.0

- MAINTENANCE_ROLE wallet is able to whitelist addresses. And the addresses not in the whitelist, won't be able to transfer tokens if the contract is paused by the "SECURITY_ROLE" address.



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

Darwin

```
◆ setDarwinSwapFactory
Ⓜ onlyRole
◆ setDarwinStaking
Ⓜ onlyRole
◆ setMasterChef
Ⓜ onlyRole
◆ registerDarwinSwapPair
Ⓜ onlyRole
◆ setLive
Ⓜ onlyRole
◆ setPauseWhitelist
Ⓜ onlyRole
◆ setPresaleAddress
Ⓜ onlyRole
◆ emergencyPause
Ⓜ onlyRole
◆ emergencyUnPause
Ⓜ onlyRole
◆ distributeRewards
◆ bulkTransfer
◆ mint
Ⓜ onlyRole
◆ burn
◆ setMinter
Ⓜ onlyRole
◆ setMaintenance
Ⓜ onlyRole
◆ setSecurity
Ⓜ onlyRole
◆ setUpgrader
Ⓜ onlyRole
◆ setReceiveRewards
Ⓜ onlyRole
◆ communityPause
Ⓜ onlyRole
◆ communityUnPause
Ⓜ onlyRole
```

DarwinCommunity

```
◆ init
◆ emitInitialFundsEvents
Ⓜ onlyRole
◆ setDarwinAddress
Ⓜ onlyDarwinCommunity
◆ setStakedDarwinAddress
Ⓜ onlyDarwinCommunity
◆ deactivateFundCandidate
Ⓜ onlyDarwinCommunity
◆ newFundCandidate
Ⓜ onlyDarwinCommunity
◆ distributeCommunityFund
Ⓜ onlyRole
◆ propose
Ⓜ onlyRole
◆ cancel
Ⓜ isProposalIdValid
◆ castVote
◆ execute 💰
Ⓜ onlyRole
◆ setProposalMaxOperations
Ⓜ onlyDarwinCommunity
◆ setMinVotingDelay
Ⓜ onlyDarwinCommunity
◆ setMinVotingPeriod
Ⓜ onlyDarwinCommunity
◆ setMaxVotingPeriod
Ⓜ onlyDarwinCommunity
◆ setGracePeriod
Ⓜ onlyDarwinCommunity
◆ setProposalMinVotesCountForAction
Ⓜ onlyDarwinCommunity
◆ setOwner
Ⓜ onlyDarwinCommunity
◆ setAdmin
Ⓜ onlyDarwinCommunity
◆ setSeniorProposer
Ⓜ onlyDarwinCommunity
◆ setProposer
Ⓜ onlyDarwinCommunity
◆ withdrawStakedDarwin
Ⓜ nonReentrant
```

Comments

- [DarwinCommunity.sol \(onlyDarwinCommunity, Admin, and OWNER\)](#)
 - Set Darwin address
 - Add new fund candidate and deactivate a fund candidate
 - Distribute community fund
 - Admin address can execute proposal
 - Set the following variables to any arbitrary values
 - Max proposal operations

- Minimum voting delay, and period
 - Maximum voting period
 - Grace Period
 - Minimum proposal votes for action
 - Grant OWNER, ADMIN, and Proposer roles to wallets
-
- Darwin.sol
 - MAINTENANCE_ROLE
 - Set Darwin Swap Factory, and Staking address. The staking address will be granted MINTER_ROLE, and can only be set once
 - Set MasterChef address but only once
 - Set presale address, only after the contract is live
 - MAINTENANCE_ROLE
 - Pause and Unpause the contract.
 - MINTER_ROLE wallets/addresses can mint tokens, but not more than the Maximum Supply
 - COMMUNITY_ROLE
 - Set Minter address, and Security
 - Set upgrader address
 - Set receive address
 - Pause/Unpause contract
 - Include/Exclude addresses from rewards
 - Note that the “Community Address” will be set at the time of deployment by the deployer
 - UPGRADER_ROLE wallets/addresses can authorise upgrade
 - There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
 - Be aware of this

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/Openzeppelin/ERC20Upgradeable.sol	1	————	416	393	129	220	91
contracts/DarwinBurner.sol	1	1	23	19	14	1	15
contracts/Darwin.sol	1	————	400	374	290	31	256
contracts/EvoturesNFT.sol	1	————	163	158	128	12	102
contracts/DarwinVester5.sol	1	————	120	120	96	8	64
contracts/DarwinVester7.sol	1	————	112	112	89	8	57
contracts/LootboxTicket.sol	1	————	58	58	46	2	35
contracts/DarwinStaking.sol	1	————	115	110	86	3	72
contracts/StakedDarwin.sol	1	————	84	84	67	1	46
contracts/DarwinCommunity.sol	1	1	630	581	419	55	424
contracts/interface/IDarwinVester.sol	————	1	34	34	20	10	1
contracts/interface/IEvoturesNFT.sol	————	1	33	31	25	10	5
contracts/interface/IDarwin.sol	————	1	77	19	10	26	41
contracts/interface/IDarwinStaking.sol	————	1	17	17	13	1	1
contracts/interface/IStakedDarwin.sol	————	1	23	7	5	————	27
contracts/interface/IERC20.sol	————	1	22	7	5	————	27
contracts/interface/ILootboxTicket.sol	————	1	15	14	10	1	3
contracts/interface/IDarwinCommunity.sol	————	1	84	82	67	6	5
Totals	10	10	2426	2220	1519	395	1272

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

Issue	File	Type	Line	Description
#1	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.

12. April 2023:

- A new version of the contracts can be deployed by the address/wallet that have the UPGRADER_ROLE privileges which can change any limit and may introduce new privileges in the updated contract.
- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

*Solid
Proofed*

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**


MADE IN GERMANY