DIA DATA

Staking

SMART CONTRACT AUDIT

21.04.2021

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of D.I.A. e.V. . If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (19.04.2021)	Layout
0.4 (19.04.2021)	Automated Security Testing Manual Security Testing
0.5 (21.04.2021)	Verify Claims and Test Deployment
0.6 (21.04.2021)	Testing SWC Checks
0.9 (21.04.2021)	Summary and Recommendation
1.1 (21.04.2021)	Final document
1.2 (22.04.2021)	Added deployed contract

2. About the Project and Company

Company address:

D.I.A. e.V. (Association)
Baarerstrasse 10
6300 Zug
Switzerland

Website: https://diadata.org

Twitter: https://twitter.com/diadata_org

Medium: https://medium.com/@diadata_org

Telegram: https://t.me/DIAdata_org

LinkedIn: https://www.linkedin.com/company/diadata-org

GitHub: https://github.com/diadata-org/diadata

2.1 Project Overview

DIA (Decentralised Information Asset) is an open-source oracle platform that enables market actors to source, supply and share trustable data. DIA aims to be an ecosystem for open financial data in a financial smart contract ecosystem, to bring together data analysts, data providers and data users. In general, DIA provides a reliable and verifiable bridge between off-chain data from various sources and on-chain smart contracts that can be used to build a variety of financial DApps. DIA is the governance token of the platform. It is currently based on ERC-20 Ethereum protocol. The project was founded in 2018, while the token supply was made available to the public during the bonding curve sale from Aug. 3 through Aug. 17, 2020, where 10.2 million tokens were sold.

Who Are the Founders of DIA?
The DIA association was co-founded by a group of a dozen people, though Paul Claudius, Michael Weber and Samuel Brack are the leaders. Claudius is the face of the project and its lead advocate, sometimes also mentioned as a CBO. He has a masters degree in international management from ESCP Europe and a bachelors in business and economics from Passau University. Apart from working on DIA, he is also a co-founder and CEO of BlockState AG and c ventures. Before crypto, he had worked as director for a nutrition company called nu3. Weber is the project's CEO. He holds a asters in management from ESCP Business School and an equivalent to a bachelors in economics and physics from University of Cologne. He has worked in several banks and financial institutions before turning to crypto, where he founded such projects as Goodcoin, myLucy and BlockState. Samuel Brack serves DIA in the role of CTO. Like both Claudius and Weber, he shares the same position at BlockState. He has a masters degree in computer science from Humboldt University of Berlin, where as of January 2020, he is still studying for his PhD.

What Makes DIA Unique?
DIA aims to become the Wikipedia of financial data. It specifically addresses the problem of dated/unverified/hard to access data in the world of finance and crypto, especially DeFi, while proposing to solve it via system of financial incentives for users to keep the flow of open-source, validated data streams to the oracles up and running. The current design of oracles, DIA argues, is non-transparent, difficult to scale and vulnerable to attack. The DIA governance token will be used to fund data collection, data validation, voting on governance decisions and to incentivize the development of the platform. Users can stake DIA tokens to incentivise new data to appear on the platform, but access to historical data though DIA is free.

3. 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

4. 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.

4.1 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 Chainsulting 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 Chainsulting 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.

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

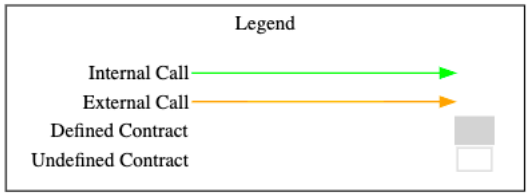
Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
@openzeppelin/contracts/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/SafeERC20.sol
@openzeppelin/contracts/utils/Address.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol

4.3 Tested Contract Files

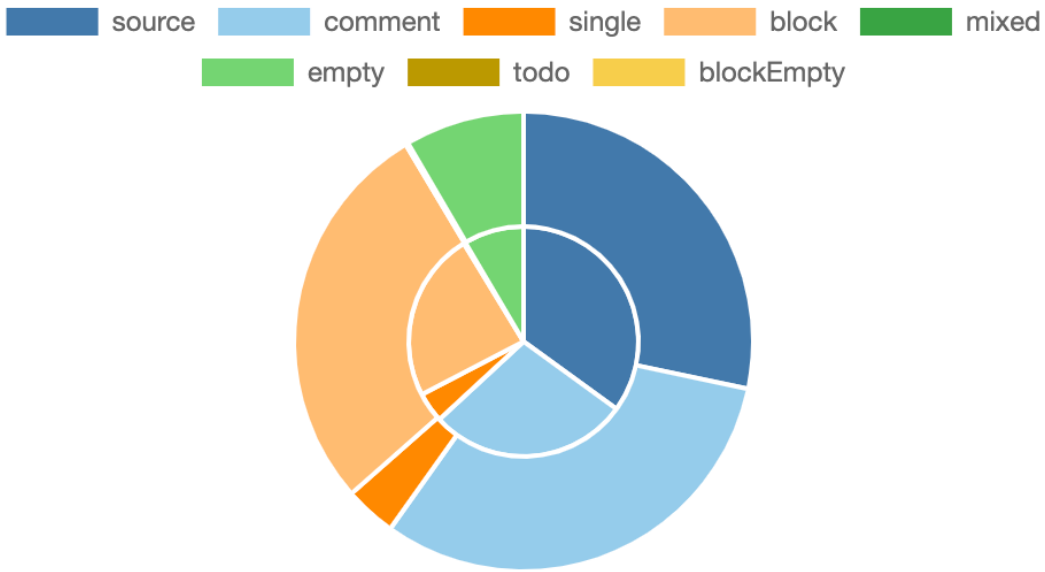
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
contracts/DIAStaking.sol	3e9030cc013ae167765462b82a1f00c3

4.4 Metrics / CallGraph



4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`0.8.3`			`yes` (2 asm blocks)	

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

🌐Public	💰Payable
21	0

External	Internal	Private	Pure	View
9	59	2	8	8

StateVariables

Total	🌐Public
9	5

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	DIAStaking.sol	6	1	673	606	297	312	207	💻📤🔆
📝📚🔍🎨	**Totals**	**6**	**1**	**673**	**606**	**297**	**312**	**207**	💻📤🔆

Legend: [📝]

- **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, ...)

5. Scope of Work

The DIA Data Team provided us with the file that needs to be tested. The scope of the audit is the DIA Data Staking contract.

Following contracts with the direct imports has been tested:

- DIAStaking.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- The users can lock at any time if the deposit deadline is not reached.
- The users can only withdraw locked amount and reward, after the locking period has ended
- Yield contract owner is not able to change beneficiary of a user lock (updateBeneficiary)
- The user is not able to withdraw reward and locked amount, before locking period ends / deposit deadline.
- Yield contact owner is not able to withdraw user locks
- Check the overall smart contract security and functions

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 Wrong import of OpenZeppelin library
Severity: LOW
Status: FIXED
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation, OpenZeppelin files are added directly into the code. This violates OpenZeppelin's MIT license, which requires the license and copyright notice to be included if its code is used.	`Address, SafeMath, IERC20 ..`	We highly recommend using npm (import "@openzeppelin/contracts/..) in order to guarantee that original OpenZeppelin contracts are used with no modifications. This also allows for any bug-fixes to be easily integrated into the codebase.

Moreover, updating code manually is error-prone.		

INFORMATIONAL ISSUES

5.1.2 SPDX license identifier
Severity: LOW
Status: FIXED
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
Warning: SPDX license identifier not provided in source file.	Line NA	Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

5.2. SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

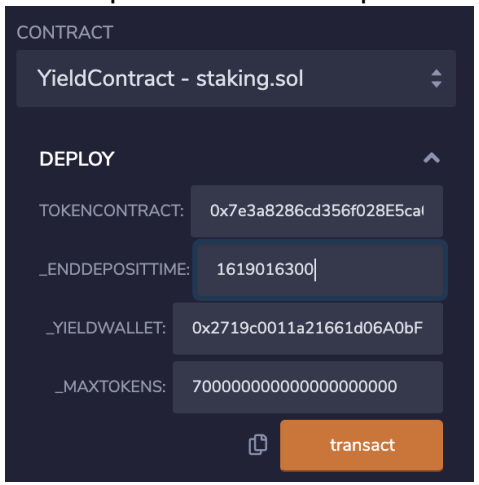
ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✅
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

6. Test Deployment

Account 1: 0xc25A908a52Eb3eaf22ba078d01470D544772a7EB (Contract Owner Yield)
Account 2: 0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC (Locker)
Account 3: 0x2719c0011a21661d06A0bF5f6e4A01a9Bb98D748 (DIA Token Owner and Yield Wallet)

6.1 Deployment of DIA Token
Tx: https://ropsten.etherscan.io/tx/0x0f2f5f5182c5772f518362a2ad19f2db1b0622c9bccfc6155ba0ffeeeae62056
Contract: https://ropsten.etherscan.io/address/0x7e3a8286cd356f028e5ca6b6ba6f74624d660c8b#code

6.2 Deployment of yield contract
(modified the period to 1 hour for testing)
End Deposit time: Wed Apr 21 2021 16:45:00 GMT+0200 (Central European Summer Time)



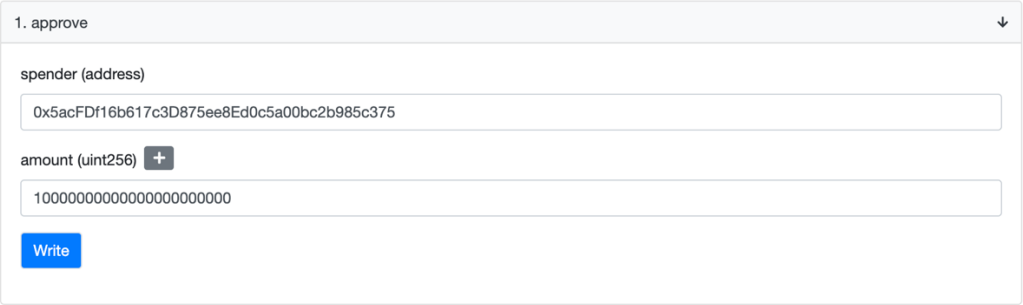
Tx: https://ropsten.etherscan.io/tx/0x3f9feca31ff7ec966338407b636747b46193d922725612c694b54151d47d386f
Contract_ https://ropsten.etherscan.io/address/0x5acfdf16b617c3d875ee8ed0c5a00bc2b985c375#code

6.3 Transfer 700 DIA Token to Account 2

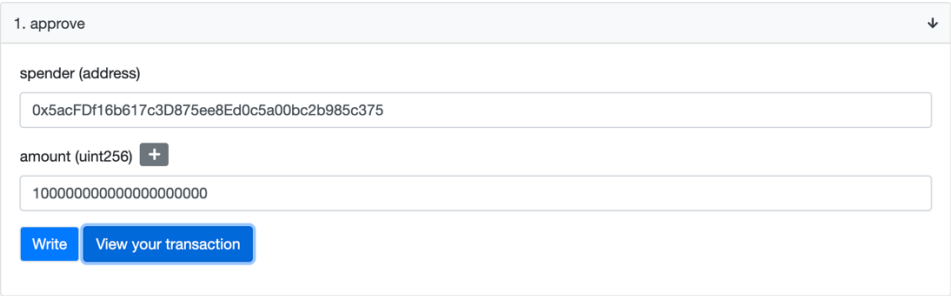
Tx: https://ropsten.etherscan.io/tx/0xfe6c42a34c12785febdd60d92a74a2d27322390ae19a90e6477bec512a97c373

6.4 Approvals

Approve yield contract as spender with the yield wallet (Account 3)
Tx: https://ropsten.etherscan.io/tx/0x6d95d34b34ceb48e87bea39ec10a0fb2c1bc8cf6abda887fcc952ede80c00130

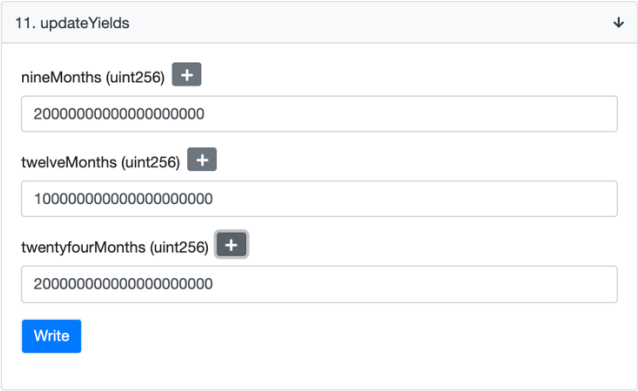


Approve Token from Account 2 with yield contract
Tx: https://ropsten.etherscan.io/tx/0x7a5ed4d2a56d37b56fac076f99314a6f6553138c52a945b63cacc4f931565a89



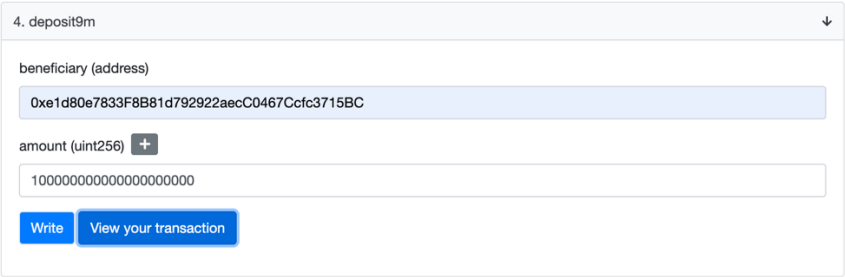
6.5 Update Yields

Tx: https://ropsten.etherscan.io/tx/0x928159246fa805b8bc8940c5f8ba152b63bb333d85701fd35bf88cdc71be8b38



6.6 Deposit 100 DIA Token

Deposit 100 DIA Token at yield contract and lock before end deposit time.



Tx: https://ropsten.etherscan.io/tx/0x45c0410715699ea5bdce68709764a75ab50e601380038e3853768aa4f4c5123e

Check, if successful

5. lockBoxStructs

<input> (uint256)

0

Query

beneficiary *address*, balance *uint256*, releaseTime *uint256*

[**lockBoxStructs(uint256)** method Response]
» **beneficiary** *address* : 0xe1d80e7833F8B81d792922aecC0467Ccfc3715BC
» **balance** *uint256* : 120000000000000000000
» **releaseTime** *uint256* : 1619019044

Wed Apr 21 2021 17:30:44 GMT+0200 (Central European Summer Time)

6.7 Yield Contract owner is not able to change beneficiary

Yield contract owner is not able to change beneficiary of a random lock (updateBeneficiary)
Result: Not possible
Tx: https://ropsten.etherscan.io/tx/0x9d9edb12814e099ca24b9da12bb81406a5d7b4a14dcf27f0895ddae9ed3f0119

Contract 0x5acfdf16b617c3d875ee8ed0c5a00bc2b985c375

Warning! Error encountered during contract execution [**Reverted**]

6.8 Withdraw before locking period ends

Check if withdraw reward and locked amount, before locking period ends / deposit deadline, is possible?
Result: Not possible
Tx: https://ropsten.etherscan.io/tx/0xe03bb7d5c0f70792d264e0e8b56f5b1533299277015fe667da5f1fbdedb57acf

Contract 0x5acfdf16b617c3d875ee8ed0c5a00bc2b985c375

Warning! Error encountered during contract execution [**Reverted**]

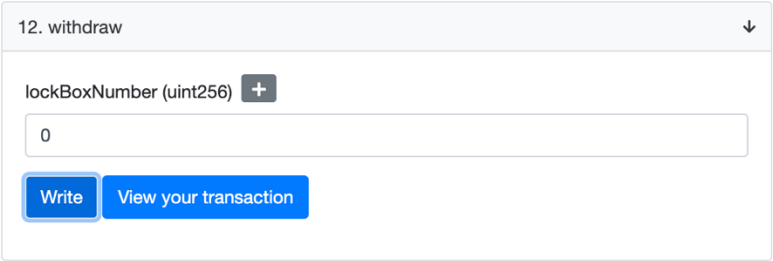
6.9 Contract owner is not able to withdraw user locks

Check if contract owner can withdraw user locks
Tx: https://ropsten.etherscan.io/tx/0xd3806b30aa16fd3e7eef74dc0e21a8e60beea9772cf10afa0c8f63592316ee86

Contract 0x5acfdf16b617c3d875ee8ed0c5a00bc2b985c375
└ Warning! Error encountered during contract execution [**Reverted**] ☹

6.10 Withdraw lock

Withdraw lock and yield after locking time is over.
Wed Apr 21 2021 17:30:44 GMT+0200 (Central European Summer Time)



Tx: https://ropsten.etherscan.io/tx/0x052b2a6c1fb90ddc0f8e01c7408a4fe7e209d7e86b5fb497c6a390384ef5bfa2

Tokens Transferred: **From** 0x5acfdf16b617c3… **To** 0xe1d80e7833f8b… **For** 120 DIAToken (DIA)

7. Verify claims

7.1 The users can lock at any time if the deposit deadline is not reached.
Status: tested and verified ✅

7.2 The users can only withdraw locked amount and reward, after the locking period has ended
Status: tested and verified ✅

7.3 Yield contract owner is not able to change beneficiary of a user lock (updateBeneficiary)
Status: tested and verified ✅

7.4 The user is not able to withdraw reward and locked amount, before locking period ends / deposit deadline.
Status: tested and verified ✅

7.5 Yield contract owner is not able to withdraw user locks
Status: tested and verified ✅

7.6 Check the overall smart contract security and functions
Status: tested and verified ✅

8. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found, after the manual and automated security testing. Only informational issues were found, to increase the code quality. Overall, everything was well documented and worked as it was supposed to be.

9. Deployed Smart Contract

VERIFIED

Smart Contract is deployed here:
https://etherscan.io/address/0x90Ef220F222e8c319504bdB510a2B739222a5f4f#code