



**Sovryn**  
**Origins Platform**  
**SMART CONTRACT AUDIT**  
**04.10.2021**

**Made in Germany by Chainsulting.de**



## Table of contents

1. Disclaimer .....	3
2. About the Project and Company .....	4
2.1 Project Overview .....	5
3. Vulnerability & Risk Level.....	6
4. Auditing Strategy and Techniques Applied .....	7
4.1 Methodology.....	7
4.2 Used Code from other Frameworks/Smart Contracts .....	8
4.3 Tested Contract Files.....	9
4.4 Metrics / CallGraph .....	11
4.5 Metrics / Source Lines & Risk .....	12
4.6 Metrics / Capabilities.....	13
4.7 Metrics / Source Unites in Scope.....	14
5. Scope of Work.....	18
5.1 Manual and Automated Vulnerability Test .....	19
5.1.1 Wrong import of OpenZeppelin library.....	19
5.1.2 A floating pragma is set .....	20
5.2 SWC-Check.....	21
5.3. Unit Tests.....	24
6. Executive Summary .....	35
7. Deployed Smart Contract.....	35

## 1. Disclaimer

The audit makes no statements or warranties 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 1A1Z Limited. 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 (23.09.2021)	Layout
0.2 (24.09.2021)	Test Deployment
0.5 (25.09.2021)	Automated Security Testing Manual Security Testing
0.6 (27.09.2021)	Testing SWC Checks
0.7 (28.09.2021)	Verify Claims
0.9 (01.10.2021)	Summary and Recommendation
1.0 (04.10.2021)	Final document
1.1 (TBA)	Added deployed contract addresses

## 2. About the Project and Company

### Company address:

1A1Z Limited  
64 Southwark Bridge Road  
London SE1 0AS  
United Kingdom

**Website:** <https://www.sovryn.app>

**Twitter:** <https://twitter.com/SovrynBTC>

**Medium:** <https://medium.com/sovryn>

**Telegram:** <https://t.me/SovrynBitcoin>

**Discord:** <https://discord.gg/J22WS6z>

**Github:** <https://github.com/DistributedCollective>



## 2.1 Project Overview

Sovryn is a decentralized application allowing users to trade Bitcoin on Layer 2 in a permissionless, noncustodial, and censorship-resistant way. It allows users to leverage Bitcoin up to 5x, and earn interest, without sacrificing their security or privacy.

### **Decentralized Governance**

Decentralization is the key to user Sovereignty. By denying central points of control, it shifts the balance of power from centralized authorities, back to the users. Sovryn is being built with the goal of maximal decentralization, while still being able to provide ease-of-use and the ability to constantly improve. Sovryn will never have the same level of decentralization of Bitcoin but it's also not trying to be the next global reserve currency. We expect to continue engineering greater assurances of decentralization, as Sovryn evolves. In the early days, Sovryn will be decentralized by the following means:

**Governance Token** — The SOV token will be a form of decentralized equity, “Dequity”. It's holders will have the ability to manage the protocol by staking and voting. In doing so, they will earn the right to the revenue the protocol generates.

**Staking** — By staking SOV, holders will be able to actively participate in management of the protocol, changes to features and distribution of revenues and rewards.

**Distributed Smart Contract Keys** — A central point of control that afflicts almost all dapps is control of the smart contract keys. If one party controls those keys, that can make almost any change to the protocol, including stealing users funds. However, some degree of control is necessary, to allow for system and feature upgrades.

Sovryn will distribute control of smart contract keys and place them under the control of the SOV holders. No one party will be in control.

### 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
Address.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/utils/Address.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/utils/Address.sol</a>
Context.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/GSN/Context.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/GSN/Context.sol</a>
ERC20.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/token/ERC20/ERC20.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/token/ERC20/ERC20.sol</a>
ERC20Detailed.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/token/ERC20/ERC20Detailed.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/token/ERC20/ERC20Detailed.sol</a>
IERC20.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/token/ERC20/IERC20.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/token/ERC20/IERC20.sol</a>
Ownable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/access/Ownable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.0.0/contracts/access/Ownable.sol</a>
ReentrancyGuard.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/utils/ReentrancyGuard.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/utils/ReentrancyGuard.sol</a>
SafeERC20.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/token/ERC20/SafeERC20.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/token/ERC20/SafeERC20.sol</a>
SafeMath.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/math/SafeMath.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.5.1/contracts/math/SafeMath.sol</a>
MultiSigWallet.sol	<a href="https://github.com/gnosis/MultiSigWallet/blob/master/contracts/MultiSigWallet.sol">https://github.com/gnosis/MultiSigWallet/blob/master/contracts/MultiSigWallet.sol</a>



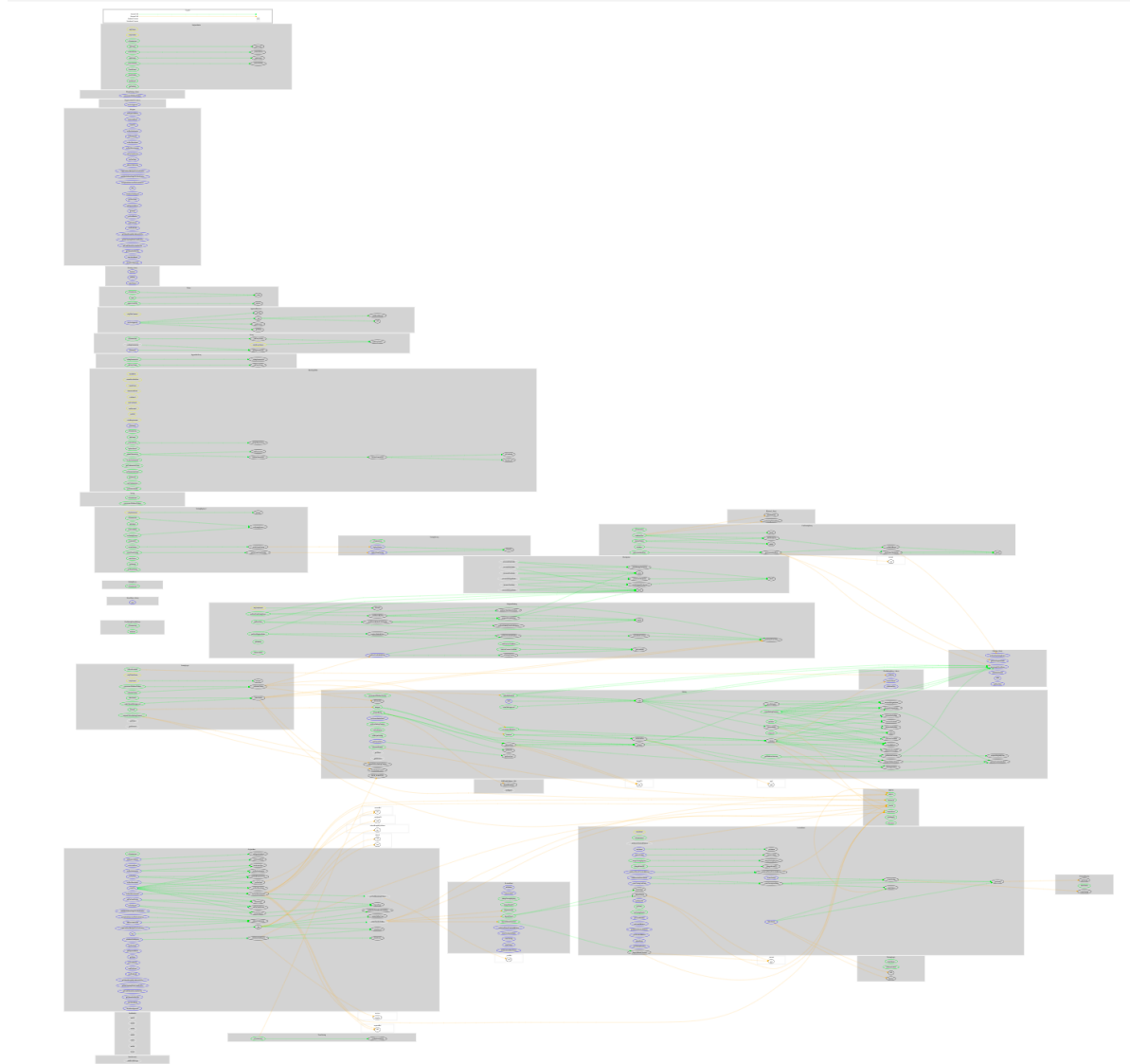
## 4.3 Tested Contract Files

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/Sovryn/Helper/ErrorDecoder.sol	0c966a3bd4709a7fcb525bd58c1deb57
contracts/Sovryn/Helper/SafeMath96.sol	55b605681a25b41460f77e46d087c9fd
contracts/Sovryn/RSK/RSKAddrValidator.sol	854a1213268d602c880c49ffb210fb8a
contracts/Sovryn/Governance/FeeSharingProxy.sol	9a6d8724763e55fadf9a2566d071a326
contracts/Sovryn/Governance/Staking/Staking.sol	27e8cee77c24a3c11ef9ede081c2460e
contracts/Sovryn/Governance/Staking/StakingProxy.sol	f7c4ada7b80a25817777b90c813415e8
contracts/Sovryn/Governance/Staking/StakingStorage.sol	27e558335bc3c2fd46852041413a5658
contracts/Sovryn/Governance/Staking/Checkpoints.sol	f5ef4394128de4f5742614ea8a43ec74
contracts/Sovryn/Governance/Staking/WeightedStaking.sol	1a83cd4a7622ceab96fab7397da66d75
contracts/Sovryn/Governance/Vesting/VestingRegistry3.sol	ce5d9f27beae6ec34e06d36d71b58757
contracts/Sovryn/Governance/Vesting/VestingFactory.sol	052179ddf063d2caf9240bc4db6f89e2
contracts/Sovryn/Governance/Vesting/Vesting.sol	ee7119132b4c0a1c61f3512fce5adb89
contracts/Sovryn/Governance/Vesting/VestingStorage.sol	dccd0cfbb7f94f9573c0c8da46c84547
contracts/Sovryn/Governance/Vesting/TeamVesting.sol	ab32b3a1bd6bb6d9d74bec757c54a9b2
contracts/Sovryn/Governance/Vesting/VestingLogic.sol	18650d36ebbcef1a19d47ddcf4a7f6c2
contracts/Sovryn/Proxy/UpgradableProxy.sol	3830a7cd6712c730ded89da05025e5d1
contracts/Sovryn/Proxy/Proxy.sol	dd5bd30a6497b11584e77d22ffe5deca
contracts/Sovryn/Token/ApprovalReceiver.sol	6bd46f036c08c73f9c99fb68a8838e91
contracts/Sovryn/Token/Token.sol	399466b6e8706c90958b8140ec44b66e
contracts/Interfaces/IVesting.sol	019eb0ffbd63ba6e8df719c6cd7a86b
contracts/Interfaces/IVestingFactory.sol	742c533d2c5c5f5bebe7341742e22aa1f
contracts/Interfaces/IVestingRegistry.sol	57dbec16cdceac339ec55ca2a081c37b
contracts/Interfaces/ILockedFund.sol	c97f378882bf3f38a84bccd7446018f2

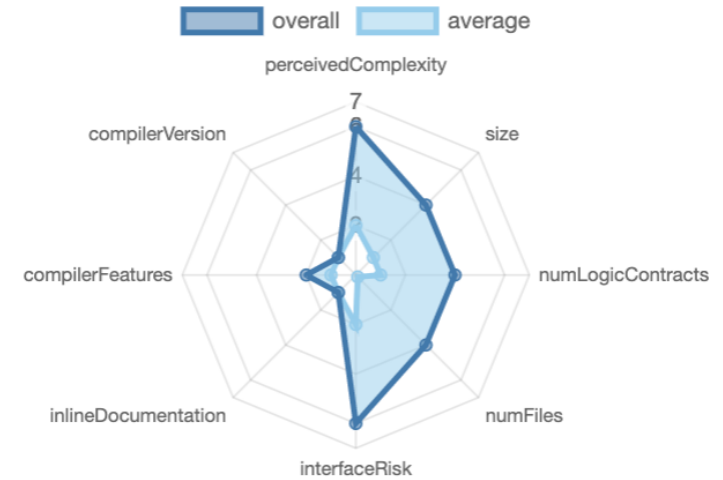
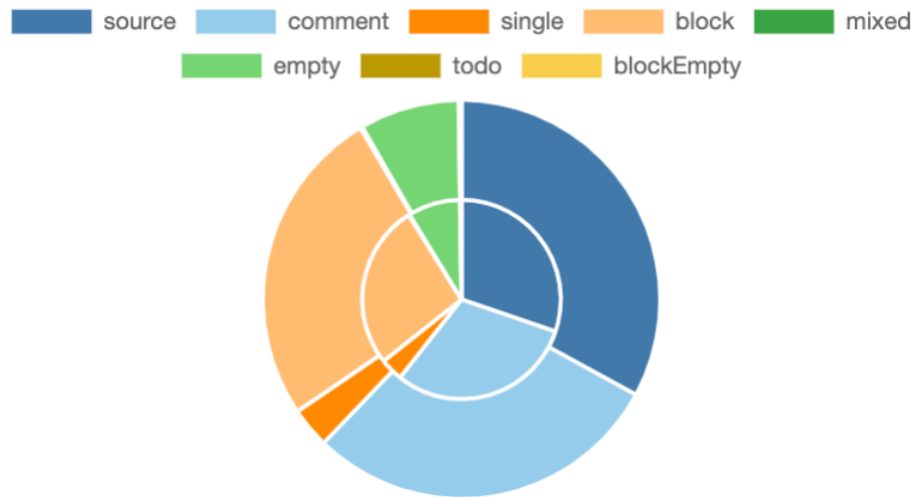
contracts/Interfaces/IFeeSharingProxy.sol	51bfb6237cde2c0eb92c946cd53e21a6
contracts/Interfaces/IStaking.sol	3ef5a02c10860c06ceca64a88c305dc8
contracts/Interfaces/IOrigins.sol	f01d2518a410e06a072d57397f4cd7fe
contracts/Interfaces/IApproveAndCall.sol	b4e5e35201b842cd74117fe268c2f61b
contracts/Interfaces/IVestingLogic.sol	8ea5cff9e992c03a4e99ae25cf61a5e4
contracts/Interfaces/IERC20.sol	c5bf1a5f6c8f911f1b4580b86dec5a6d
contracts/Interfaces/ITeamVesting.sol	a518984859d9474b6fddc1d9994f2535
contracts/OriginsEvents.sol	b6ef4b6a7a3889edfab76c57add28955
contracts/OriginsAdmin.sol	64e18923819e24fb8727bdb0aa421e59
contracts/OriginsStorage.sol	2f707d2be1810ba3b810ca7a876d3723
contracts/OriginsBase.sol	6c12de6398242b6ea024611cc9383f32
contracts/LockedFund.sol	7150f57ac1042e3c95cbea2da2ce5dda

## 4.4 Metrics / CallGraph













View full version: <https://chainsulting.de/wp-content/uploads/2021/10/solidity-metrics-sovryn-origins.html>

## 4.5 Metrics / Source Lines & Risk





## 4.6 Metrics / Capabilities


Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts	
<code>^0.5.17</code> <code>&gt;=0.5.0 &lt;0.6.0</code>	<code>ABIEncoderV2</code>	<code>yes</code>	<code>yes</code> (9 asm blocks)		
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECTransfer	 New/Create/Create2
<code>yes</code>		<code>yes</code>	<code>yes</code>	<code>yes</code>	<code>yes</code> → <code>NewContract:Vesting</code> → <code>NewContract:TeamVesting</code>

### Exposed Functions















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





















 <b>Public</b>	 <b>Payable</b>				
217	4				
<b>External</b>	<b>Internal</b>	<b>Private</b>	<b>Pure</b>	<b>View</b>	
103	290	0	13	88	















### StateVariables

<b>Total</b>	 <b>Public</b>
102	70









## 4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Sovryn/Helper/ErrorDecoder.sol	1	_____	53	53	18	34	27	_____
	contracts/Sovryn/Helper/SafeMath96.sol	1	_____	113	97	38	49	12	
	contracts/Sovryn/RSK/RSKAddrValidator.sol	1	_____	21	21	9	10	3	_____
	contracts/Sovryn/Mockup/FeeSharingProxyMockup.sol	1	_____	23	19	14	_____	10	_____
	contracts/Sovryn/Governance/FeeSharingProxy.sol	1	2	353	311	141	133	99	
	contracts/Sovryn/Governance/Staking/Staking.sol	1	_____	718	649	274	307	266	
	contracts/Sovryn/Governance/Staking/StakingProxy.sol	1	_____	22	22	9	11	8	_____
	contracts/Sovryn/Governance/Staking/StakingStorage.sol	1	_____	127	127	39	57	35	
	contracts/Sovryn/Governance/Staking/Checkpoints.sol	1	_____	211	181	85	72	28	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Sovryn/Governance/Staking/WeightedStaking.sol	1	_____	447	413	207	168	117	
	contracts/Sovryn/Governance/Vesting/VestingRegistry3.sol	1	_____	200	182	108	49	82	
	contracts/Sovryn/Governance/Vesting/VestingFactory.sol	1	_____	79	55	22	30	44	
	contracts/Sovryn/Governance/Vesting/Vesting.sol	1	_____	39	39	17	19	9	
	contracts/Sovryn/Governance/Vesting/VestingStorage.sol	1	_____	43	43	16	17	12	_____
	contracts/Sovryn/Governance/Vesting/TeamVesting.sol	1	_____	55	55	35	17	22	
	contracts/Sovryn/Governance/Vesting/VestingLogic.sol	1	_____	220	216	94	90	92	
	contracts/Sovryn/Multisig/MultiSigWallet.sol	1	_____	404	365	192	143	228	
	contracts/Sovryn/Proxy/UpgradableProxy.sol	1	_____	39	39	10	26	11	_____
	contracts/Sovryn/Proxy/Proxy.sol	1	_____	125	125	60	52	106	
	contracts/Sovryn/Token/ApprovalReceiver.sol	1	_____	105	100	50	40	55	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Sovryn/Token/Token.sol	1	_____	68	64	30	29	25	_____
	contracts/Interfaces/IVesting.sol	_____	1	16	11	3	7	7	_____
	contracts/Interfaces/IVestingFactory.sol	_____	1	30	11	3	7	5	_____
	contracts/Interfaces/IVestingRegistry.sol	1	_____	33	14	3	19	7	_____
	contracts/Interfaces/ILockedFund.sol	1	_____	116	15	3	70	25	_____
	contracts/Interfaces/IFeeSharingProxy.sol	_____	1	17	8	3	4	7	_____
	contracts/Interfaces/IStaking.sol	_____	1	39	8	3	4	13	_____
	contracts/Interfaces/IOrigins.sol	1	_____	291	17	3	176	58	
	contracts/Interfaces/IApproveAndCall.sol	_____	1	21	15	3	11	3	_____
	contracts/Interfaces/IVestingLogic.sol	1	_____	23	15	4	13	7	_____
	contracts/Interfaces/IERC20.sol	1	_____	31	13	6	4	16	_____
	contracts/Interfaces/ITeamVesting.sol	_____	1	11	10	3	6	3	_____
	contracts/OriginsEvents.sol	1	_____	173	173	45	110	3	_____



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/OriginsAdmin.sol	1	_____	228	228	93	103	71	_____
	contracts/OriginsStorage.sol	1	_____	142	142	63	68	14	_____
	contracts/OriginsBase.sol	1	_____	832	750	352	303	279	
	contracts/LockedFund.sol	1	_____	634	605	225	290	168	
	<b>Totals</b>	<b>31</b>	<b>8</b>	<b>6102</b>	<b>5211</b>	<b>2283</b>	<b>2548</b>	<b>1977</b>	

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, ...)

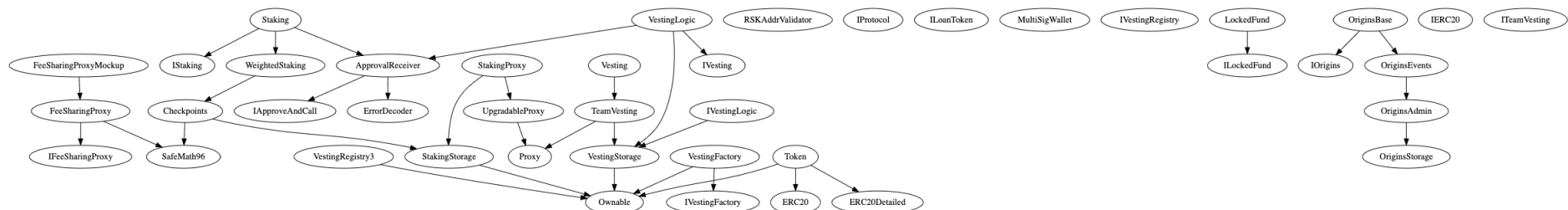
## 5. Scope of Work

The Sovryn Team provided us with the files that needs to be tested. The scope of the audit are the origins platform contracts.

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

- The smart contract is coded according to the newest standards and in a secure way

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: ACKNOWLEDGED

File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation, OpenZeppelin libraries are added directly into the code or added manually as file. This violates OpenZeppelin's MIT license, which requires the license and copyright notice to be included if its code is used. Moreover, updating code manually is error-prone.	SafeMath, Address, Context, ERC20, ERC20Detailed, IERC20, Ownable, ReentrancyGuard, SafeERC20	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.

## INFORMATIONAL ISSUES

5.1.2 A floating pragma is set

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: SWC-103

File(s) affected: ALL



Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is "^0.5.17". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.	Line 1: <code>pragma solidity ^0.5.17;</code>	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee.  i.e. Pragma solidity 0.5.17

## 5.2 SWC-Check

ID	Title	Relationships	Test Result
<a href="#">SWC-131</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	✓
<a href="#">SWC-130</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	✓
<a href="#">SWC-129</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	✓
<a href="#">SWC-128</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	✓
<a href="#">SWC-127</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	✓
<a href="#">SWC-125</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	✓
<a href="#">SWC-124</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	✓
<a href="#">SWC-123</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	✓
<a href="#">SWC-122</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-121</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	✓
<a href="#">SWC-120</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	✓
<a href="#">SWC-119</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-118</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	✓
<a href="#">SWC-117</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	✓
<a href="#">SWC-116</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-115</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-114</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	✓
<a href="#">SWC-113</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	✓
<a href="#">SWC-112</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-111</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-110</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	✓
<a href="#">SWC-109</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	✓
<a href="#">SWC-108</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-107</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	✓
<a href="#">SWC-106</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	✓
<a href="#">SWC-105</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	✓
<a href="#">SWC-104</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	✓
<a href="#">SWC-103</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	✗
<a href="#">SWC-102</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-101</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	
<a href="#">SWC-100</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	

### 5.3. Unit Tests

#### Contract: LockedFund (Admin Functions)

- ✓ Admin should be able to add another admin.
- ✓ Admin should not be able to add zero address as another admin.
- ✓ Admin should not be able to add another admin more than once.
- ✓ Admin should be able to remove an admin.
- ✓ Admin should not be able to call removeAdmin() with a normal user address.
- ✓ Admin should be able to change the vestingRegistry.
- ✓ Admin should not be able to change the vestingRegistry as a zero address.
- ✓ Admin should be able to change the waited timestamp.
- ✓ Admin should not be able to change the waited TS as zero.
- ✓ Admin should be able to deposit using depositVested().
- ✓ Admin should not be able to deposit using depositWaitedUnlocked() with invalid basis point.
- ✓ Admin should not be able to deposit with the duration as zero.
- ✓ Admin should not be able to deposit with the duration higher than max allowed.
- ✓ Admin should not be able to deposit with basis point higher than max allowed.

#### Contract: LockedFund (Creator Functions)

- ✓ Creator should not be able to create a lockedFund contract with zero as waited timestamp.
- ✓ Creator should not be able to create a lockedFund contract with zero address for token.
- ✓ Creator should not be able to create a lockedFund contract with zero address as vesting registry.
- ✓ Creator should not be able to create a lockedFund contract with invalid admin address.
- ✓ Creator should not be able to add another admin.



- ✓ Creator should not be able to remove an admin.
- ✓ Creator should not be able to change the vestingRegistry.
- ✓ Creator should not be able to change the waited timestamp.
- ✓ Creator should not be able to deposit using depositVested().

#### Contract: LockedFund (Events)

- ✓ Adding another admin should emit AdminAdded.
- ✓ Removing another admin should emit AdminRemoved.
- ✓ Changing the vestingRegistry should emit VestingRegistryUpdated.
- ✓ Changing the waited timestamp should emit WaitedTSUpdated.
- ✓ Depositing using depositVested() should emit VestedDeposited.
- ✓ Withdrawing waited unlocked balance using withdrawWaitedUnlockedBalance() should emit WithdrawnWaitedUnlockedBalance.
- ✓ Withdrawing waited unlocked balance to another wallet using withdrawWaitedUnlockedBalance() should emit WithdrawnWaitedUnlockedBalance.
- ✓ Creating vesting and staking vested balance using createVestingAndStake() should emit VestingCreated and TokenStaked.
- ✓ Creating vesting using createVesting() should emit VestingCreated.
- ✓ Staking vested balance using stakeTokens() should emit TokenStaked.
- ✓ Withdrawing waited unlocked balance, creating vesting and staking vested balance using withdrawAndStakeTokens() should emit WithdrawnWaitedUnlockedBalance, VestingCreated and TokenStaked.
- ✓ Withdrawing waited unlocked balance to any wallet, creating vesting and staking vested balance using withdrawAndStakeTokens() should emit WithdrawnWaitedUnlockedBalance, VestingCreated and TokenStaked.

#### Contract: LockedFund (State Change)

- ✓ Admin should be able to add another admin.
- ✓ Admin should be able to remove an admin.
- ✓ Admin should be able to change the vestingRegistry.
- ✓ Admin should be able to change the waited timestamp.
- ✓ Admin should be able to deposit using depositVested() and unlock as waited.
- ✓ Admin should be able to deposit using depositVested() and unlock as immediate.
- ✓ Admin should be able to deposit using depositVested() and unlock as none.
- ✓ Admin should be able to deposit using depositLocked().
- ✓ Admin should be able to deposit using depositWaitedUnlocked() with non zero basis point.
- ✓ Admin should be able to deposit using depositWaitedUnlocked() with zero basis point.

- ✓ User should be able to withdraw waited unlocked balance using `withdrawWaitedUnlockedBalance()`.
- ✓ User should be able to withdraw waited unlocked balance to any wallet using `withdrawWaitedUnlockedBalance()`.
- ✓ User should be able to create vesting and stake vested balance using `createVestingAndStake()`.
- ✓ User should be able to create vesting using `createVesting()`.
- ✓ User should be able to stake vested balance using `stakeTokens()`.
- ✓ User should be able to withdraw waited unlocked balance, create vesting and stake vested balance using `withdrawAndStakeTokens()`.
- ✓ User should be able to withdraw waited unlocked balance to any wallet, create vesting and stake vested balance using `withdrawAndStakeTokens()`.
- ✓ Admin should be able to get the cliff and duration of a user.

#### Contract: LockedFund (User Functions)

- ✓ User should not be able to add another admin.
- ✓ User should not be able to remove an admin.
- ✓ User should not be able to change the vestingRegistry.
- ✓ User should not be able to change the waited timestamp.
- ✓ User should not be able to deposit using `depositVested()`.
- ✓ User should be able to withdraw waited unlocked balance using `withdrawWaitedUnlockedBalance()`.
- ✓ User should be able to withdraw waited unlocked balance to any wallet using `withdrawWaitedUnlockedBalance()`.
- ✓ User should not be able to withdraw waited unlocked balance before waitedTimestamp using `withdrawWaitedUnlockedBalance()`.
- ✓ User should be able to create vesting and stake vested balance using `createVestingAndStake()`.
- ✓ User should be able to create vesting using `createVesting()`.
- ✓ User should be able to stake vested balance using `stakeTokens()`.
- ✓ User should not be able to stake vested balance using `stakeTokens()` if vesting is not created previously.
- ✓ User should be able to withdraw waited unlocked balance, create vesting and stake vested balance using `withdrawAndStakeTokens()`.
- ✓ User should be able to withdraw waited unlocked balance to any wallet, create vesting and stake vested balance using `withdrawAndStakeTokens()`.
- ✓ User should not be able to create vesting and stake vested balance using `createVestingAndStake()` if cliff and duration is not set.

**Contract: OriginsAdmin (Owner Functions)**

- ✓ Creator should not be able to create an instance with same owner address twice.
- ✓ Creator should not be able to add another owner.
- ✓ Creator should not be able to remove an owner.
- ✓ Creator should not be able to add a verifier.
- ✓ Creator should not be able to remove a verifier.

**Contract: OriginsAdmin (Owner Functions)**

- ✓ Adding another owner should emit OwnerAdded.
- ✓ Remove an owner should emit OwnerRemoved.
- ✓ Adding a verifier should emit VerifierAdded.
- ✓ Removing a verifier should emit VerifierRemoved.

**Contract: OriginsAdmin (Owner Functions)**

- ✓ Owner should be able to add another owner.
- ✓ Owner should not be able to add zero address as another owner.
- ✓ Owner should not be able to add another owner more than once.
- ✓ Owner should be able to remove an owner.
- ✓ Owner should not be able to call removeOwner() with a normal user address.
- ✓ Owner should be able to add a verifier.
- ✓ Owner should not be able to add zero address as a verifier.
- ✓ Owner should not be able to add another verifier more than once.
- ✓ Owner should be able to remove a verifier.
- ✓ Owner should not be able to call removeVerifier() with a normal user address.

**Contract: OriginsAdmin (Owner Functions)**

- ✓ Owner should be able to add another owner.
- ✓ Owner should be able to remove an owner.
- ✓ Owner should be able to add a verifier.
- ✓ Owner should be able to remove a verifier.
- ✓ Owner should be able to get owner list.
- ✓ Owner should be able to get verifier list.

**Contract: OriginsBase (Creator Functions)**

- ✓ Creator should not be able to create a originsBase with invalid token address.

- ✓ Creator should not be able to set deposit address.
- ✓ Creator should not be able to set Locked Fund Contract.
- ✓ Creator should not be able to add a new tier.
- ✓ Creator should not be able to set Tier Deposit Parameters.
- ✓ Creator should not be able to set Tier Token Limit Parameters.
- ✓ Creator should not be able to set Tier Token Amount Parameters.
- ✓ Creator should not be able to set Tier Vest or Lock Parameters.
- ✓ Creator should not be able to set Tier Time Parameters.
- ✓ Creator should not be able to verify a single address to a single tier.
- ✓ Creator should not be able to verify a single address to a multiple tier.
- ✓ Creator should not be able to verify a multiple address to a single tier.
- ✓ Creator should not be able to verify a multiple address to a multiple tier.

#### Contract: OriginsBase (Owner Functions)

- ✓ Owner should be able to set deposit address.
- ✓ Owner should not be able to add zero address as deposit address.
- ✓ Owner should be able to set Locked Fund Contract.
- ✓ Owner should not be able to add zero address as Locked Fund Contract.
- ✓ Owner should be able to add a new tier.
- ✓ Owner should be able to set Tier Verification Parameters.
- ✓ Owner should be able to set Tier Deposit Parameters.
- ✓ Owner should not be able to set Tier Deposit Parameters with deposit rate as zero.
- ✓ Owner should not be able to set Tier Deposit Parameters with deposit token as zero address if deposit type is Token.
- ✓ Owner should be able to set Tier Deposit Parameters with correct deposit token address if deposit type is Token.
- ✓ Owner should be able to set Tier Deposit Parameters with deposit token as zero address if deposit type is RBTC.
- ✓ Owner should be able to set Tier Token Limit Parameters.
- ✓ Owner should not be able to set Tier Token Limit Parameters with minimum is greater than maximum amount.
- ✓ Owner should be able to set Tier Token Amount Parameters where extra is provided.
- ✓ Owner should be able to set Tier Token Amount Parameters where extra is returned.
- ✓ Owner should not be able to set Tier Token Amount Parameters with remaining token as zero.

- ✓ Owner should not be able to set Tier Token Amount Parameters with max allowed is higher than remaining token.
- ✓ Owner should be able to set Tier Vest or Lock Parameters.
- ✓ Owner should not be able to set Tier Vest or Lock Parameters with cliff higher than duration.
- ✓ Owner should not be able to set Tier Vest or Lock Parameters with cliff higher than duration.
- ✓ Owner should be able to set Tier Time Parameters.
- ✓ Owner should not be able to set Tier Time Parameters with sale start timestamp is higher than sale end timestamp.
- ✓ Owner should not be able to set Tier Time Parameters with sale start timestamp is higher than sale end timestamp.
- ✓ Owner should be able to withdraw the sale deposit to deposit address.
- ✓ Owner should be able to withdraw the sale deposit to own address if deposit address is not set.
- ✓ Owner should be able to withdraw the sale deposit and remaining tokens to deposit address.

#### Contract: OriginsBase (State Functions)

- ✓ Setting a deposit address should update the state.
- ✓ Setting Locked Fund Contract should update the state.
- ✓ Adding a new tier should update the state correctly.
- ✓ Owner should be able to set Tier Verification Parameters.
- ✓ Owner should be able to set Tier Deposit Parameters.
- ✓ Owner should be able to set Tier Token Limit Parameters.
- ✓ Owner should be able to set Tier Token Amount Parameters.
- ✓ Owner should be able to set Tier Vest or Lock Parameters.
- ✓ Owner should be able to set Tier Time Parameters.
- ✓ Verifier should be able to verify a single address to a single tier.
- ✓ Verifier should be able to verify a single address to a multiple tier.
- ✓ Verifier should be able to verify a multiple address to a single tier.
- ✓ Verifier should be able to verify a multiple address to a multiple tier.
- ✓ User should be able to buy tokens.
- ✓ User should be able to buy tokens multiple times until max asset amount reaches.
- ✓ User should be refunded the extra after the max reaches.
- ✓ Owner should be able to withdraw the sale deposit to deposit address.

#### Contract: OriginsBase (User Functions)

- ✓ User should be able to buy tokens.



- ✓ User should be able to buy tokens multiple times until max asset amount reaches.
- ✓ User should be allowed to buy until the max reaches.
- ✓ User should be allowed to buy with Immediate Unlock.
- ✓ User should be allowed to buy with Waited Unlock.
- ✓ User should be allowed to buy with Locked Transfer.
- ✓ User should not be allowed to buy without setting Transfer Type.
- ✓ User should not be allowed to buy if sale end duration or TS not set.
- ✓ User should not be allowed to buy if sale start time is not set.
- ✓ User should not be allowed to buy if the token sale has ended.
- ✓ User should not be allowed to buy if sale end is not set.
- ✓ User should not be allowed to buy if total tokens are sold with Sale End as Duration.
- ✓ User should not be allowed to buy if total tokens are sold with Sale End as Until Supply.
- ✓ User should not be allowed to buy if Verification is not set.
- ✓ If verification is done by address, user should only be allowed if it is done by verified address.
- ✓ User should not be allowed to buy once the max reaches even if there is remaining tokens.
- ✓ User should not be allowed to buy tokens with zero deposit in RBTC
- ✓ User should not be allowed to buy with assets deposited less than minimum allowed.
- ✓ User should be able to get token address.
- ✓ User should be able to get Tokens Bought By Address On Tier.
- ✓ User should be able to get Participating Wallet Count Per Tier.
- ✓ User should be able to get Total Token Allocation Per Tier.
- ✓ User should be able to get Tokens Sold Per Tier.
- ✓ User should be able to check Sale Ended.

### Contract: OriginsBase (Verifier Functions)

- ✓ Verifier should not be able to set deposit address.
- ✓ Verifier should not be able to set Locked Fund Contract.
- ✓ Verifier should not be able to add a new tier.
- ✓ Verifier should not be able to set Tier Deposit Parameters.
- ✓ Verifier should not be able to set Tier Token Limit Parameters.
- ✓ Verifier should not be able to set Tier Token Amount Parameters.
- ✓ Verifier should not be able to set Tier Vest or Lock Parameters.
- ✓ Verifier should not be able to set Tier Time Parameters.
- ✓ Verifier should be able to verify a single address to a single tier.
- ✓ Verifier should not be able to verify a zero address to a single tier.
- ✓ Verifier should be able to verify a single address to a multiple tier.
- ✓ Verifier should be able to verify a multiple address to a single tier.
- ✓ Verifier should be able to verify a multiple address to a multiple tier.
- ✓ Verifier should not be able to verify a multiple address of length x to a multiple tier of length y, where  $x \neq y$ .

Solc version: 0.5.17		Optimizer enabled: true		Runs: 200	Block limit: 6800000 gas	
Methods						
		Min	Max	Avg		
LockedFund	addAdmin	47855	47879	47875	35	-
LockedFund	changeVestingRegistry	-	-	30545	9	-
LockedFund	changeWaitedTS	-	-	30231	4	-
LockedFund	createVesting	-	-	43979	3	-
LockedFund	createVestingAndStake	3369078	3373878	3372278	3	-
LockedFund	depositLocked	-	-	24726	1	-
LockedFund	depositVested	53919	150330	103383	29	-
LockedFund	depositWaitedUnlocked	52997	58291	55644	2	-
LockedFund	removeAdmin	-	-	25969	3	-
LockedFund	stakeTokens	1340642	1345442	1343842	3	-
LockedFund	withdrawAndStakeTokens	1347788	2816949	2325590	6	-
LockedFund	withdrawWaitedUnlockedBalance	43793	61128	58161	6	-
OriginsAdmin	addOwner	-	-	74955	3	-
OriginsAdmin	addVerifier	-	-	92077	4	-
OriginsAdmin	removeOwner	-	-	35540	3	-
OriginsAdmin	removeVerifier	-	-	33660	3	-
OriginsBase	addressVerification	-	-	48198	5	-
OriginsBase	addVerifier	-	-	92122	4	-
OriginsBase	buy	118883	251034	167281	32	-



OriginsBase	createTier	269023	310655	296404	67	-
OriginsBase	multipleAddressAndTierVerification	-	-	79076	2	-
OriginsBase	multipleAddressSingleTierVerification	78168	98068	88118	2	-
OriginsBase	setDepositAddress	-	-	31083	3	-
OriginsBase	setLockedFund	31082	48182	46673	34	-
OriginsBase	setTierDeposit	34293	37379	35065	4	-
OriginsBase	setTierTime	-	-	33714	2	-
OriginsBase	setTierTokenAmount	62837	82415	71874	3	-
OriginsBase	setTierTokenLimit	50491	53291	52358	3	-
OriginsBase	setTierVerification	-	-	30859	2	-
OriginsBase	setTierVestOrLock	-	-	55922	2	-
OriginsBase	singleAddressMultipleTierVerification	-	-	77736	2	-
OriginsBase	withdrawSaleDeposit	70963	194031	117013	4	-
StakingProxy	setImplementation	-	-	47953	10	-
Token	approve	26258	46158	45524	118	-
Token	mint	36385	70609	46634	117	-
VestingFactory	transferOwnership	-	-	28709	14	-
VestingRegistry3	addAdmin	47218	47230	47227	21	-

Deployments					
FeeSharingProxyMockup	1400804	1400816	1400815	20.6 %	–
LockedFund	1764379	1764391	1764390	25.9 %	–
OriginsAdmin	–	–	745710	11 %	–
OriginsBase	4000336	4024643	4023859	59.2 %	–
Staking	–	–	4830363	71 %	–
StakingProxy	991880	991892	991891	14.6 %	–
Token	–	–	1113260	16.4 %	–
VestingFactory	2054061	2054073	2054072	30.2 %	–
VestingLogic	–	–	1520139	22.4 %	–
VestingRegistry3	1058303	1058315	1058309	15.6 %	–

187 passing (25s)

🌟 Done in 71.75s.

## 6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the October 04, 2021.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical issues were found, after the manual and automated security testing and the claim have been successfully verified.

## 7. Deployed Smart Contract

PENDING

