



**Oiler Vesting & Token**  
**SMART CONTRACT AUDIT**

05.05.2021

**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 / CallGrap.....	10
4.6 Metrics / Capabilities.....	12
4.7 Metrics / Source Unites in Scope.....	13
5. Scope of Work.....	14
5.1 Manual and Automated Vulnerability Test .....	15
5.2. SWC Attacks & Special Checks .....	16
6. Test Deployment & Verify claims .....	20
7. Executive Summary .....	26
8. Deployed Smart Contract.....	26



## 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 Oiler DeFi. 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 (15.03.2021)	Layout
0.5 (16.03.2021)	Verify Claims and Test Deployment
0.6 (17.03.2021)	Testing SWC Checks
0.8 (17.03.2021)	Automated Security Testing Manual Security Testing
0.9 (18.03.2021)	Summary and Recommendation
1.0 (20.03.2021)	Final document
1.1 (05.05.2021)	Added deployed contract

## 2. About the Project and Company

### Company address:

Oiler DeFi c/o International Corporation Services Ltd  
PO Box 472, Harbour Place 2nd  
Floor103 South Church Street  
Grand Cayman  
KY1-1106, Cayman Islands

Website: <https://oiler.network>

Twitter: <https://twitter.com/OilerNetwork>

Telegram: <https://t.me/oilernetwork>

Telegram: [https://t.me/oiler\\_official](https://t.me/oiler_official)

Medium: <https://medium.com/oiler-network>

LinkedIn: <https://www.linkedin.com/company/oiler-network/>

## 2.1 Project Overview

Oiler is a protocol for blockchain native derivatives.

Before 2019/2020, it was practically impossible to deliver blockchain native derivatives. Without stablecoins and AMMs (automatic market makers), it was not possible to provide a reliable pricing solution. What has changed? Stablecoins introduced a non-volatile on-chain base for pricing (a USD peg) and AMMs introduced a pricing discovery mechanism; on-chain and with high volumes. It has also been proven that if the markets are efficient then the AMMs are efficient too and arbitrageurs will set the price right.

In order to settle derivatives on-chain nowadays, we need to ensure that the payout can be calculated entirely on-chain. At Oiler, they not only assume that they will not take off-chain data but also that there is no oracle hidden behind the layers of on-chain data sources that the smart contracts use. It means that the prices of the underlying instruments should not be derived from a protocol that uses off-chain oracles. Moreover, it is desirable to avoid any on-chain oracles like Uniswap since they can always be manipulated within a flash loan based attack. The last requirement is much stronger but still holds for the initial set of Oiler products.

DeFi users are most likely to use Oiler if they already have exposure to blockchain protocol parameters — their operations rely on big shifts in the network behavior, manifesting via big hashrate changes, massive gas price movements and protocol behavior changes. Who would that be?

- Exchanges that cover the highly volatile on-chain assets withdrawal costs
- Miners having exposure to shifting block reward and transaction fees
- Institutions with exposures to many blockchains and a need for hedging the protocol-level risks

Oiler will continue bringing products — both new underlying blockchain parameters and new instrument types that will be related to the category of blockchain protocol trading. They will continue working on pricing models without oracles and will look at the instruments related to both Ethereum 1.0 and Ethereum 2.0 (and their coexistence).

read more at <https://docs.oiler.network/oiler-network/>

### 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	Link / Source
openzeppelin-solidity/contracts/math/SafeMath.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/math/SafeMath.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/math/SafeMath.sol</a>
openzeppelin-solidity/contracts/ownership/Ownable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/access/Ownable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/access/Ownable.sol</a>
openzeppelin-solidity/contracts/token/ERC20/SafeERC20.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/token/ERC20/SafeERC20.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/token/ERC20/SafeERC20.sol</a>
openzeppelin-solidity/contracts/utils/Address.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/utils/Address.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v2.3.0/contracts/utils/Address.sol</a>

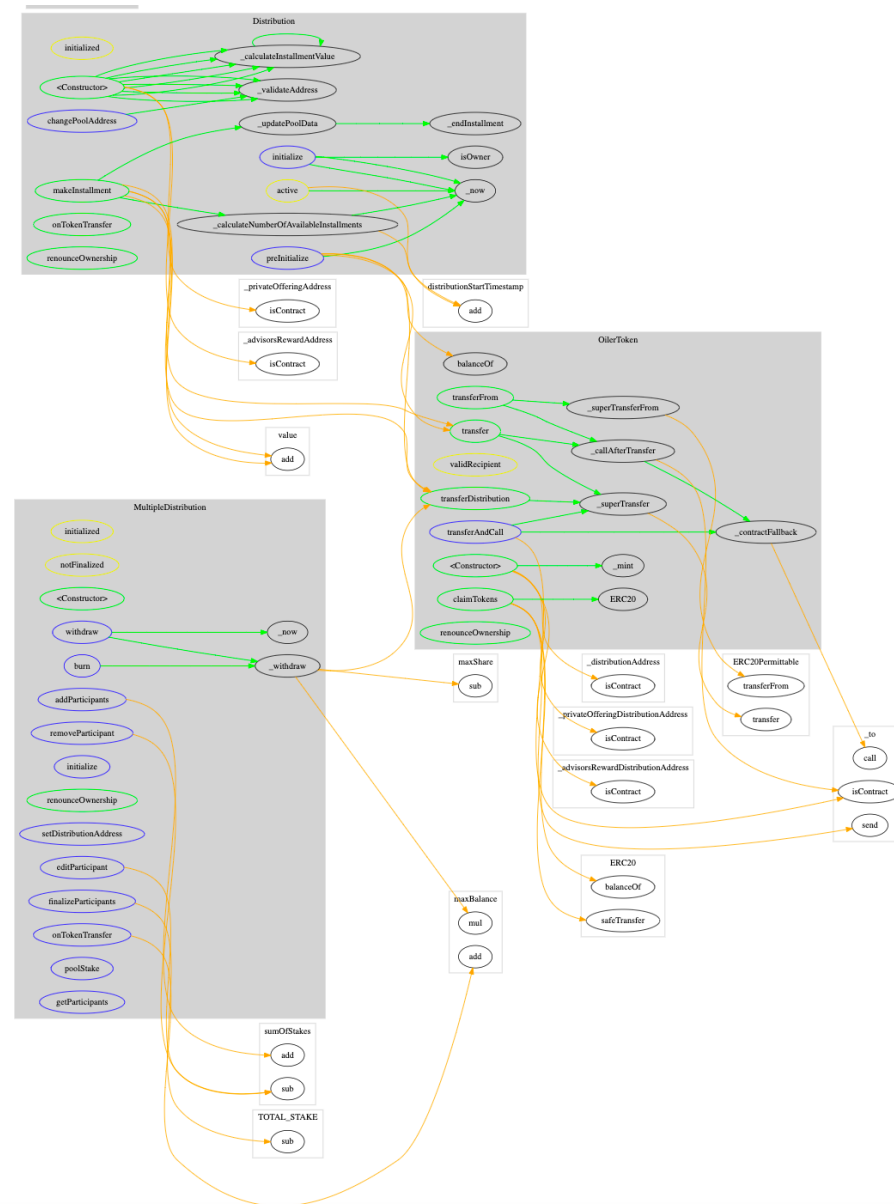
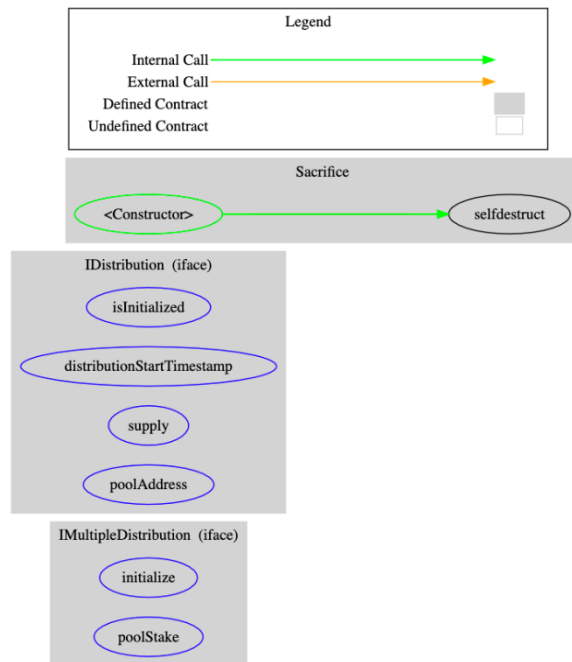


## 4.3 Tested Contract Files

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/Token/OilerToken.sol	c8d112c25fb40e11cb64b03b715a3d93
contracts/Token/Sacrifice.sol	2814f402b4d68f22f0dbddad8155ca73
contracts/Distribution.sol	3410c541a188145ac2c34baadeba8404
contracts/IDistribution.sol	24db0703c98418e0eed0be35ce1cb538
contracts/IMultipleDistribution.sol	386bbd0c6745f7490287c9798eb661d6
contracts/MultipleDistribution.sol	94edcdfbc073350de047d5837354089e

## 4.4 Metrics / CallGrap





## 4.5 Metrics / Source Lines















## 4.6 Metrics / Capabilities

Solidity Versions observed		 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<div>0.5.12</div>			<div>yes</div>	**** (0 asm blocks)	<div>yes</div>
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
<div>yes</div>					

 Public	 Payable				
34	1				
External	Internal	Private	Pure	View	
21	44	1	2	12	

## 4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Distribution.sol	1	_____	363	352	222	104	147	
	contracts/MultipleDistribution.sol	1	_____	289	275	158	73	141	_____
	contracts/IMultipleDistribution.sol	_____	1	6	4	3	_____	5	
	contracts/IDistribution.sol	_____	1	8	4	3	_____	9	_____
	contracts/Token/OilerToken.sol	1	_____	232	223	118	87	101	
	contracts/Token/Sacrifice.sol	1	_____	8	8	6	_____	7	
	<b>Totals</b>	<b>4</b>	<b>2</b>	<b>906</b>	<b>866</b>	<b>510</b>	<b>264</b>	<b>410</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

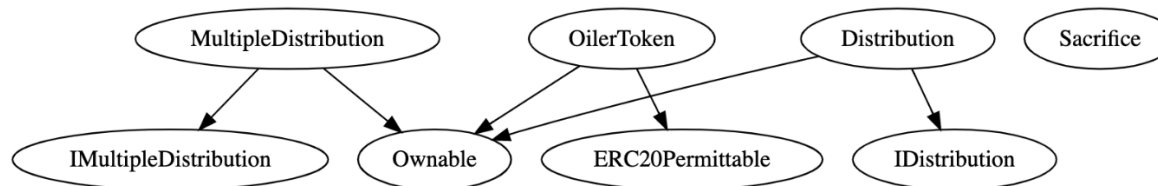
The Oiler Team provided us with the files that needs to be tested. The scope of the audit is the Oiler Vesting & Token contract. Following contracts with the direct imports has been tested:

- Distribution.sol
- OilerToken.sol

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

- Oiler Token must behave like a safe and regular ERC20 token - no malicious minting, burning, stealing, locking, etc.
- Distribution contract must distribute funds to the specified addresses in a specified timeline, and in no other way. This means that OilerTokens stored in Distribution must remain there safe, and can only be withdrawn when the timeline specified unlocking allows it to - some portion after the cliff, and daily afterwards - to the corresponding pools, and there is no way for anyone to access these funds via other means.
- The only exception to the above “cliff+daily afterwards” rule are LIQUIDITY\_FUND - which is unlocked and sent entirely during preInitialization, and PRIVATE\_OFFERING which unlocks 25% of funds during preInitialization, but they can't be withdrawn for 5 days (more on that below)
- Each MultipleDistribution POOL must allow its recorded participants (and only them) to withdraw their part of already unlocked stake, and have no other malicious options of accessing these funds. At the same time there shouldn't be a situation when the funds suddenly become locked forever.
- PRIVATE\_OFFERING MultipleDistribution pool has an additional limitation mentioned in pt.3 above - withdrawals are locked for 5 days since distribution start date (counting from Initialization)

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**

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





## 5.2. SWC Attacks & Special Checks

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>	



ID	Title	Relationships	Test Result
<a href="#">SWC-122</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	✓
<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>	✓

ID	Title	Relationships	Test Result
<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>	✓
<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>	✓

ID	Title	Relationships	Test Result
<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>	
<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>	

## 6. Test Deployment & Verify claims

### 6.1 Deploy PrivateOfferingDistribution (MultipleDistribution contract)

Tx: <https://kovan.etherscan.io/tx/0x714c3a8e752a15bad41399583952b5eb6780d3c480e060b68ade866851823c29>

Contract: <https://kovan.etherscan.io/address/0x97af0b4f96b4277ac445ab611392bf54a10d80d7>

### 6.2 Deploy AdvisorRewardDistribution (MultipleDistribution contract)

Tx: <https://kovan.etherscan.io/tx/0x0e449cd2dc3adb75f402729ca551e865291ba9bd0005f93f4284c1472dcb3bf4>

Contract: <https://kovan.etherscan.io/address/0x823Bffa5c837f0bb09c6D9e9904ba836394b895b>

### 6.3 Deploy Distribution contract

Constructor variables:


Variable	Value
ECOSYSTEM_FUND_ADDRESS	0x9D0783F0a6BDfA9B40c9335D3A96da2750f45C1e
TEAM_FUND_ADDRESS	0x7A6F5917703202D909190b354fCA57b2905187b9
PRIVATE_OFFERING_ADDRESS	0x97Af0B4f96B4277AC445Ab611392bf54A10d80d7
ADVISOR_REWARD_ADDRESS	0x823BFfa5c837f0bB09C6D9e9904ba836394b895b
FOUNDATION_REWARD_ADDRESS	0x32179D90B7e51F55bc3B7844CfCcFBE89ec4f0EC
LIQUIDITY_FUND_ADDRESS	0x01ba52C313F609683AF5dc5Af6fdB7A378A2253d

Tx: <https://kovan.etherscan.io/tx/0xc0e238ac5811d872392d63e235810c609aee051607c31866f7de8b7c0cedaad9>

Contract: <https://kovan.etherscan.io/address/0x3857d26289f00f3e42ef71b5594229158bb30b35>

## 6.4 Deploy Oiler Token

All funds (100 000 000 Tokens) are sent to distribution contract.

**Status:** tested and verified 

Tx: <https://kovan.etherscan.io/tx/0x6dc4a0057031a299cd230f559fe9e8ad98179da898e3d6e4d29f59905dd1610a>



Contract: <https://kovan.etherscan.io/address/0x7efaea31c85ae7947fe54e6d8f83de9a8f8a3dcc>

## 6.5 Pre-initialize Distribution contract

**Status:** tested and verified 


Funds are distributed as expected. Distribution contract sends 10 000 000 Tokens to Liquidity fund address and 3 405 275.5 tokens to private funding address. The remaining tokens stay safe in Distribution contract until they get unlocked.

Tx: <https://kovan.etherscan.io/tx/0x89af9c5415054efe4ca888501d005f3ce10868e53254ac8daa56031136c19948>

Rank	Address	Quantity	Percentage
1	 0x3857d26289f00f3e42ef71b5594229158bb30b35	86,594,724.5	86.5947%
2	0x01ba52c313f609683af5dc5af6fdb7a378a2253d	10,000,000	10.0000%
3	 0x97af0b4f96b4277ac445ab611392bf54a10d80d7	3,405,275.5	3.4053%

## 6.6 Initialize Distribution contract


The initialization sets the start time for the distribution to the current timestamp and can only be called after preinitialization.

**Status:** tested and verified 

Tx: <https://kovan.etherscan.io/tx/0xb86631a932d85111acfe1c2c0ad32b6c5d0ff41138b9815511ac8196a424337b>


## 6.7 Withdrawing funds from private funds pool

Withdrawing funds from private funds pool is only possible for whitelisted users after the initial locking period.

**Status:** tested and verified 

### 6.7.1 Withdraw funds from private funds pool during locking period

Transaction fails as expected.

 Fail with error 'Private Offering withdrawals are locked during LBP event'

Tx: <https://kovan.etherscan.io/tx/0x41dca479235882ab55660c0cb0f5ae0a76636cddcafd590a051d9d3fd2985136>

### 6.7.2 Withdraw funds after initial locking from private funds pool with participant


Funds are sent successfully from private funds pool to participant.

► **From** 0x97af0b4f96b427... **To** 0x488025158db88... **For** 19,230.75  Molier (MOL)

Tx: <https://kovan.etherscan.io/tx/0xaffaceeb19a360d70a2b1091843f4a53040ee926d6f70173f60cf0ac748c8fb0>

### 6.7.3 Withdraw funds after locking period from private funds pool with not whitelisted participant


Transaction fails as expected.

 Fail with error 'you are not a participant'

Tx: <https://kovan.etherscan.io/tx/0x3d59ba4c7a16e4d2344379fc69644dea7671bb626b6f9e7c55baef7416443d20>

## 6.8 Withdrawing funds from advisor funds pool

Withdrawing funds from advisor funds pool is only possible for whitelisted users after the cliff.

**Status:** tested and verified 

- 6.8.1 Withdraw funds of advisor reward pool before cliff.  
Transaction fails as expected. Withdraws can only be done after cliff.

 Fail with error 'no tokens available to withdraw'


Tx: <https://kovan.etherscan.io/tx/0x897de6189bc5ba56a1dc6cb8a734c6d89c70b8fe658895e4966937f91db74ce5>

- 6.8.2 Make installments for advisor funds pool before cliff  
Transaction fails as expected. Installments can only be made after the cliff.

 Fail with error 'installments are not active for this pool'

Tx: <https://kovan.etherscan.io/tx/0x7b82d7c0b228c84251a43b3e37601f3fa4d80250320ed453b8b268b972a82de9>

- 6.8.3 Make installments for advisor funds pool after cliff  
Funds are sent successfully from distribution to advisor fund pool.

► From 0x3857d26289f00f... To 0x823bffa5c837f0... For 1,070,333.4  Molier (MOL)

Tx: <https://kovan.etherscan.io/tx/0x07728afaf0a30fdde534eef2704860824fa619e831ccf2dfc755067c0303a8af>

- 6.8.4 Withdraw funds from advisor reward pool after locking period with whitelisted participant  
Funds are successfully sent to the caller of the function, if he is a participant of the pool.

► From 0x823bffa5c837f0... To 0x488025158db88... For 239,000  Molier (MOL)

Tx: <https://kovan.etherscan.io/tx/0x25be61552bb2f51621632d9fbc1ad74f29af6c2e5a440aba9feed9cc9ed9cc1a>

- 6.8.5 Withdraw funds from advisor reward pool after locking period with not whitelisted participant  
Transaction fails as expected. Caller has to be a participant.

✖ Fail with error 'you are not a participant'

Tx: <https://kovan.etherscan.io/tx/0x04da78f6ed7f6a4b9f7031c992506d3576d6af0b59b3cc98922ac25adcae08f1>

## 6.9 Installments for Foundation only available after cliff for foundation

The foundation can only claim its tokens after the cliff for the foundation.

**Status:** tested and verified 

- 6.9.1 Make installment for foundation before cliff  
Transaction fails as expected. Installments are only available after the cliff.

✖ Fail with error 'installments are not active for this pool'

Tx: <https://kovan.etherscan.io/tx/0x9e33106231c601b50aca93e2a95ebd977b35dab6ce1ef3ba0121b47ca37c2273>

- 6.9.2 Make installment for foundation after cliff  
Funds are sent successfully from distribution to the foundation address.

► From 0x3857d26289f00f... To 0x32179d90b7e51... For 7,705,446.2  Molier (MOL)

Tx: <https://kovan.etherscan.io/tx/0xc174f8f2ca9ddcab4feabf8fa856937f7ff25c741368c1ad937836df74e0c6dd>

- 6.9.3 Try to do installment multiple times  
Transaction fails as expected. Make installment can only be called once.


✖ Fail with error 'no installments available'

Tx: <https://kovan.etherscan.io/tx/0xf190dbdb620acfda4e55859b63730c61bb0f58d9517e51d12ff167e568a69ccf>



## 6.10 Installments for Team only available after cliff for team

The team can only claim its tokens after the cliff for the team.

**Status:** tested and verified 

- 6.10.1 Make installment for team before cliff  
Transaction fails as expected. Installments are only available after the cliff.

✖ Fail with error 'installments are not active for this pool'

Tx: <https://kovan.etherscan.io/tx/0xe2cd93f4cec1bde4510433494d84f1ceee9488f1e56463f4b0f89b5c94ea78c1>

- 6.10.2 Make installment for team after cliff  
Funds are sent successfully from distribution to the team address.

► From 0x3857d26289f00f... To 0x7a6f591770320... For 1,250,000  Molier (MOL)

Tx: <https://kovan.etherscan.io/tx/0x78b410f648eeaea631cdc7de7744374a1c2b89bd5b49f4760e2c621d9105aa55>

- 6.10.3 Try to do installment multiple times  
Transaction fails as expected. Make installment can only be called once.

✖ Fail with error 'no installments available'

Tx: <https://kovan.etherscan.io/tx/0xd09226995ab8b90f65d1cd5359f5e5e407eb30c2ad6ea165c732a549dbe37b20>

## 7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The overall code quality of the project is very good, and the simplicity greatly benefits the overall security. It implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations. Our auditors highly enjoyed the good documentation and unit tests, one of the few contracts that have been reviewed at chainsulting, with no issues and consistently good code quality.

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 issues were found after the manual and automated security testing.

## 8. Deployed Smart Contract

### VERIFIED

Private Offering Distribution (MultipleDistribution.sol)

<https://etherscan.io/address/0x1d041E3A90da2240Ba298B85bcb6c32275e42B99#code>

Advisors Reward Distribution (MultipleDistribution.sol)

<https://etherscan.io/address/0xa27342A82b0bfeE68e366Fd84FC517BDDEab4aE4#code>

Distribution (Distribution.sol)

<https://etherscan.io/address/0x5A3E535C93558bD89287Aa4ef3752FD726517673#code>

Oiler Token

<https://etherscan.io/address/0x0275E1001e293C46CFE158B3702AADe0B99f88a5#code>

