



**Unicrypt Token Vesting**  
**SMART CONTRACT AUDIT**

07.07.2021

**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 .....	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
5.1.1 Contract code size exceeds .....	16
6. Test Deployment & Verify claims .....	17
7. Unit Tests.....	27
8. Executive Summary .....	32
9. Deployed Smart Contract.....	32



## 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 Unicrypt Network. 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 (11.05.2021)	Layout
0.5 (17.05.2021)	Verify Claims and Test Deployment
0.6 (14.05.2021)	Testing SWC Checks
0.8 (15.05.2021)	Automated Security Testing Manual Security Testing
0.9 (16.05.2021)	Summary and Recommendation
1.0 (18.05.2021)	Final Document
1.1 (07.07.2021)	Added deployed contract

## 2. About the Project and Company

### **Company address:**

SDD Tech OÜ  
Mustamäe tee 6b  
Tallinn Harjumaa 10616

**Website:** <https://unicrypt.network>

**Twitter:** [https://twitter.com/UNCX\\_token](https://twitter.com/UNCX_token)

**Telegram:** [https://t.me/uncx\\_token](https://t.me/uncx_token)

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

## 2.1 Project Overview

UniCrypt is a decentralized services provider which offers several ways for DeFi projects to build community trust and keep users safe. Famously, UniCrypt created the first-ever liquidity locking smart contracts for Uniswap on Ethereum, known as Proof-of-Liquidity or POL. From there the project continued to develop new features, combining liquidity locking with a decentralized launchpad.

**Liquidity Lockers:** these are smart contracts that enable teams to publicly lock liquidity on Uniswap or other AMMs for a predetermined period. Essentially, it's a guarantee to investors that the project developers can't drain the pool of all the funds. A key innovation is UniCrypt's lockers will be able to migrate liquidity to Uniswap V3 when the time comes.

**FaaS:** This is a yield farming-as-a-service protocol that enables the creation of a farm for any token. Launch a farm in a couple clicks using the UI, all automatic with no coding necessary.

**Launchpad:** Perhaps the most interesting service, a 100% decentralized and automated presale platform that is connected to the liquidity lockers. Once the presale ends a portion of the raised funds (between 30% to 100%) will create the DEX pair on a supported AMM and the liquidity will be locked.

### 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/contracts/token/ERC20/IERC20.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/token/ERC20/IERC20.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/token/ERC20/IERC20.sol</a>
@openzeppelin/contracts/security/ReentrancyGuard.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/security/ReentrancyGuard.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/security/ReentrancyGuard.sol</a>
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/utils/structs/EnumerableSet.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/utils/structs/EnumerableSet.sol</a>
@openzeppelin/contracts/access/Ownable.sol	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/access/Ownable.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.0.0/contracts/access/Ownable.sol</a>
FullMath.sol	<a href="https://gist.github.com/paulrberg/439ebe860cd2f9893852e2cab5655b65">https://gist.github.com/paulrberg/439ebe860cd2f9893852e2cab5655b65</a>

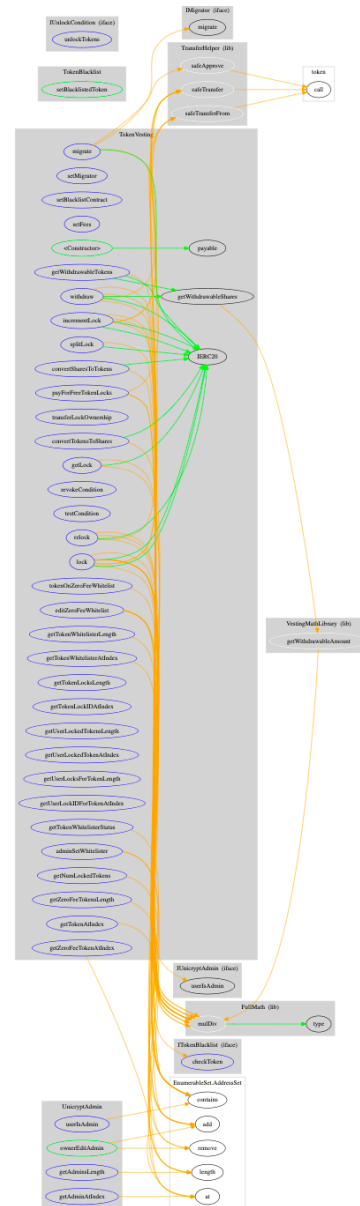
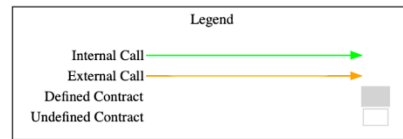


## 4.3 Tested Contract Files

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/FullMath.sol	5ed71d7eb53f69e5ac501115175e9f32
contracts/TokenBlacklist.sol	a9cbc652a84205606737fef1f2a174e7
contracts/TokenVesting.sol	29bb065fd83692340350fb160cad2229
contracts/TransferHelper.sol	7a76ffd13dfed9a198911fc8a2d7d739
contracts/UnicryptAdmin.sol	9f3ec91b4a5e3d8bc059da6c3c77ba46
contracts/VestingMathLibrary.sol	ce5557f42b5165053a0664c93b25180b

## 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
<code>^0.8.0</code> <code>&gt;=0.6.0</code>			<code>yes</code>	<code>yes</code> (7 asm blocks)	
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRecover	 New/Create/Create2
<code>yes</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.

 Public	 Payable				
46	1				
External	Internal	Private	Pure	View	
43	25	0	1	30	

### StateVariables

Total	 Public
15	8

## 4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/UnicryptAdmin.sol	1	1	42	39	24	6	22	
	contracts/FullMath.sol	1	_____	110	106	50	55	99	
	contracts/TransferHelper.sol	1	_____	21	21	15	2	13	_____
	contracts/TokenVesting.sol	1	3	578	569	368	145	291	
	contracts/TokenBlacklist.sol	1	1	26	23	11	7	12	_____
	contracts/VestingMathLibrary.sol	1	1	41	38	23	11	13	
	<b>Totals</b>	<b>6</b>	<b>6</b>	<b>818</b>	<b>796</b>	<b>491</b>	<b>226</b>	<b>450</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 Unicrypt Team provided us with the files that needs to be tested. The scope of the audit is the Token Vesting contract.

Following contracts with the direct imports has been tested:

- TokenVesting.sol
- FullMath.sol

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

- Works for rebasing and highly deflationary tokens, basically any token that works in a Uniswap pool
- Allows a locker to transfer ownership, increment a lock amount, increment the lock date, revoke premature locking conditions
- Allows for pre-mature locking conditions which allow funds to be prematurely withdrawn when IUnlockCondition.unlockTokens() returns true
- Typical safety whereby only a lock owner can withdraw the lock or transfer it, increment the lock date
- Linear scaling locks allow a user to withdraw an amount of tokens each block scaling linearly from the startEmission date to the endEmission date
- Unicrypt (Deployer) is not able to withdraw locked tokens
- 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**

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

## INFORMATIONAL ISSUES

### 5.1.1 Contract code size exceeds

Severity: INFORMATIONAL

File(s) affected: TokenVesting.sol

Attack / Description	Code Snippet	Result/Recommendation
Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off revert strings, or using libraries. --> TokenVesting.sol:26:1:   26   contract TokenVesting is Ownable, ReentrancyGuard {   ^ (Relevant source part starts here and spans across multiple lines).	NA	Enable optimization for deployment



## 6. Test Deployment & Verify claims

### 6.1 Deployment of UnicryptAdmin

Tx: <https://kovan.etherscan.io/tx/0xf2a7fa11549374bebd610cd2718f00c6c7cc28cf68f0fcde5f853575045ba732>

Contract: <https://kovan.etherscan.io/address/0x9fb5557f644a45e8ae37a6bdf7c39878701e8df8#code>

### 6.2 Deployment of TokenVesting

Tx: <https://kovan.etherscan.io/tx/0x2a75131b8c0a9e1a036f95dadedc0c261c81a375776097a81be2bad03ff38ed5>

Contract: <https://kovan.etherscan.io/address/0x887f0582ef6db1aec2c18afe77989e059a0ed8a5#code>

Set BlacklistContract

Contract: <https://kovan.etherscan.io/address/0x5a1a99d852d36a8e5f30a340f3d37d97cf7dbe34#code>

Tx: <https://kovan.etherscan.io/tx/0x071f65dcac95d1b550b473ebc42a4c9485a82b89e9c051f9d9b5023f06b09107>

Set Fees


Tx: <https://kovan.etherscan.io/tx/0x0359f148cb9813402faceaa2beef46540ceb4ec162ec8ad558075b47b98bc965>

### 6.3 Deployment of TokenBlacklist

Tx: <https://kovan.etherscan.io/tx/0x9d0fdc18aa46377019cda45b49847a1f30fc904cc7a58f9cea44e64b8973535f>

Contract: <https://kovan.etherscan.io/address/0x5a1a99d852d36a8e5f30a340f3d37d97cf7dbe34#code>

### 6.4 Works for rebasing and highly deflationary tokens, basically any token that works in a Uniswap pool

Status: tested and verified 

#### 6.4.1 Deployment of a self-destructive token (BOMB)

Tx: <https://ropsten.etherscan.io/tx/0xe26ac0f67537e7fc4b8cea949d3c9fa7471a76f5700c583419c56775a4187538>

Contract: <https://ropsten.etherscan.io/address/0x22c9fa11a2fe9c32d86403f18671235f74761f03#code>

#### 6.4.2 Approve 1000 BOMB Token with TokenVesting contract

Tx: <https://ropsten.etherscan.io/tx/0x3ad54396f01e2034080a4808db2f164c8c33638650902a6dc8b0f19672195b76>

### 6.4.3 Lock 1000 BOMB Token

Tx: <https://ropsten.etherscan.io/tx/0xc73cc777c125d319dd3a03bebf188c5b1e1fd43237265b43b4dab38aabaf83e>

[illegible]

#### 6.4.4 Withdraw Lock 990 BOMB Token

Tx: <https://ropsten.etherscan.io/tx/0xa4cc78e367c0fa487add0aeedc262173122d7b232d8a930b24ea3a0ec6d3bb6e>

Received 980 BOMB and 1% was burned doing the withdraw (No dust at the TokenVesting Contract)



16. withdraw ↓

\_lockID (uint256) +

0


\_amount (uint256) +

990

Write

View your transaction

## 6.5 Allows a locker to transfer ownership, increment a lock amount, increment the lock date, revoke premature locking conditions

**Status:** tested and verified 

### 6.5.1 Deployment of ERC20 Token

Tx: <https://kovan.etherscan.io/tx/0x1ddf07aa4348c9c51763407bccd693c8672cfc321868cd3451fea7d334106746>

Contract: <https://kovan.etherscan.io/address/0x64604a303c9dc1c92bab12c325db9bb29a75ba5f#code>

### 6.4.2 Approve 1000 DEMO Token with TokenVesting contract

Tx: <https://kovan.etherscan.io/tx/0xf5b7718bc33196ec1dfbd4fa8abc2fd259cbc8c41fdebc2d96088a4882ae11a2>

### 6.5.3 Locked 200 DEMO Token under Lock ID 4 (Lock Type 1)

Lock Type 1: when startEmission == 0 the lock is considered lockType 1. This is a normal lock whereby tokens can be withdrawn on the due date (endEmission).

```
_lock_params [["0x8CC424176Dd31802996989f006972d2B8F76Cb01", '20000000000000000000', 0, 1621253700, "0x0000000000000000000000000000000000000000"]]
```

Tx: <https://kovan.etherscan.io/tx/0xb205e01b5bd85b7d1bb6d0ee427855dab8678e3c64bcbee0149621dacb0e68ec>

Withdraw Tx: <https://kovan.etherscan.io/tx/0xb2f85febff61486b15e722bdaca082a332d250e4a559dfeca1ff9c2e3cf5cd61>

### 6.5.4 Locked 200 DEMO Token under Lock ID 6 (Lock Type 2)

Lock Type 2: when startEmission != 0. Lock tokens over a period, with an amount withdrawable every block. This scales linearly over time from startEmission -> endEmission. e.g. If the lock period is 100 seconds, 50 seconds after the startEmission you can withdraw 50% of the lock. Instead of making 10 locks for 10 months to withdraw tokens at the end of each month, you can now make 1 linear scaling lock with a period of 10 months and withdraw the relative share every block.

```
_lock_params  
[["0x8CC424176Dd31802996989f006972d2B8F76Cb01", '20000000000000000000', 600, 1621279800, "0x0000000000000000000000000000000000000000"]]
```

startEmission: 10min (600)

endEmission: 21:30 (1621279800)

[ **LOCKS(uint256)** method Response ]

» **tokenAddress** *address* : [0x64604a303C9dC1c92BaB12c325DB9bb29a75BA5f](#)  
» **sharesDeposited** *uint256* : 20000000000000000000  
» **sharesWithdrawn** *uint256* : 0  
» **startEmission** *uint256* : 600  
» **endEmission** *uint256* : 1621279800  
» **lockID** *uint256* : 6  
» **owner** *address* : [0x8CC424176Dd31802996989f006972d2B8F76Cb01](#)  
» **condition** *address* : 0x00

4. lock ↓

\_token (address)

0x64604a303C9dC1c92BaB12c325DB9bb29a75BA5f

\_lock\_params (tuple[])

[["0x8CC424176Dd31802996989f006972d2B8F76Cb01", '20000000000000000000', 600, 1621279800, "0x00"]]

Write

View your transaction

Tx: <https://kovan.etherscan.io/tx/0x682592e8b8aaf5a65962970d256aececd45d0106906b9043c9f83d6da561607f>

6.5.4 Transfer Ownership of Lock ID 6 -> to ID 7

Tx: <https://kovan.etherscan.io/tx/0xb55d64eafb56a5cacb5f99b4821e20473b613bb58b5de0a98256afb381915cb2>

6.5.5 Increment a lock amount of Lock ID 7 (+100 DEMO Token)

Tx: <https://kovan.etherscan.io/tx/0xab2f184c9764ab97b9c3fae0413980ddcf228af34063bca58135b85ad259ac73>

6.5.6 Increment the lock date of Lock ID 7 to 21:45

Tx: <https://kovan.etherscan.io/tx/0x5c2e1cebe257ea5862b4d51fff8d56e7b3488911b04806c87dc6fe605d55232b>

[ LOCKS(uint256) method Response ]

» tokenAddress address: 0x64604a303C9dC1c92BaB12c325DB9bb29a75BA5f

» sharesDeposited uint256: 30000000000000000000

» sharesWithdrawn uint256: 10000000000000000000

» startEmission uint256: 600

» endEmission uint256: 1621280700

» lockID uint256: 7

» owner address: 0x0D36bc31E54C41E4BC6FBd1cEB7124A8aBFa5a32

» condition address: 0x00

6.5.7 Withdraw Lock ID 7 after 21:45

Tx: <https://kovan.etherscan.io/tx/0xb0c9eb9ac85e113a142a68e314ade5c521f3db3e5d3573c2939479f7a2cced60>


6.5.8 Token got blacklisted while locking period

Lock Tx: <https://kovan.etherscan.io/tx/0x47053f6c8365c9df0ab54bfaf7505a883e2af45e0c1b24f06aba741d9b3d7b6e>

Blacklist Tx: <https://kovan.etherscan.io/tx/0x55b9aa1f59b991d665478286b51d38852824b2dc4d18470c579e2793885f9045>

Withdraw Tx: <https://kovan.etherscan.io/tx/0xfe3558d5f27e676d5a535cfacc0ec070231bdee1f8782d51ae447e1e2baefde>

## 6.6 Allows for pre-mature locking conditions which allow funds to be prematurely withdrawn when IUnlockCondition.unlockTokens() returns true

Status: tested and verified 

### 6.6.1 Deployment Conditions Light

Tx: <https://kovan.etherscan.io/tx/0x583e1afb08572b903834201c10a43bd0c288e5adad3d371bf49bd4f91d5f24fb>

Contract: <https://kovan.etherscan.io/address/0x57b4b77e99365860513a72e84704eb183ae120da#code>

unlockTokens = False

### 6.6.2 Locked 200 DEMO Token under Lock ID 9 (CUSTOM PREMATURE UNLOCKING CONDITIONS)

All locks support premature unlocking conditions. A premature unlock condition can be anything that implements the IUnlockCondition interface. If IUnlockCondition(address).unlockTokens() returns true, the lock withdraw date is overridden and the entire lock value can be withdrawn. The key here is this is for premature unlocks, locks always fall back to the endEmission date even if unlockTokens() returns false, and are therefore always withdrawable in full by the unlockDate.

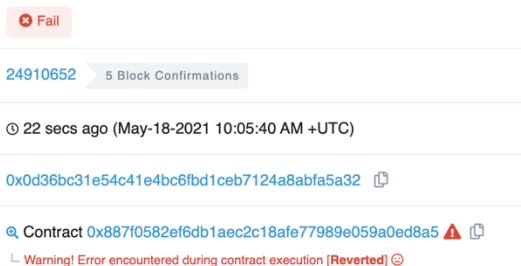
\_lock\_params

```
[[["0x0D36bc31E54C41E4BC6FBd1cEB7124A8aBFa5a32", '20000000000000000000', 0, 11621287600, "0x57b4b77E99365860513a72E84704eb183AE120DA"]]]
```


startEmission: 0

endEmission: 23:40 (11621287600)

Tx: <https://kovan.etherscan.io/tx/0x9b97ebc24afa4b7203488eeb071d986d42c9fdb44e101249f37c943dfc4ef4a6>



## 6.7 Typical safety whereby only a lock owner can withdraw the lock or transfer it, increment the lock date

Status: tested and verified 

### 6.7.1 Withdraw of Lock ID 7 with the old owner

Tx: <https://kovan.etherscan.io/tx/0x74d66f9562a5da264da01dbe6f1a3e0d378e330b3b40003eae6e58ba80a62904>

✖ Fail with error 'OWNER'

### 6.7.2 Change Increment lock date (not owner)

Tx: <https://kovan.etherscan.io/tx/0xdc39ca8f02b6079365050fc890b332f535e5cac7e6d39f65403e7b81a00172b9>


✖ Fail with error 'TransferHelper: TRANSFER\_FROM\_FAILED'

### 6.7.3 Change ownership (not owner)

Tx: <https://kovan.etherscan.io/tx/0x256bbb85a2e673663dbb6e6c706def6d55fe2801cb13fd6a53ff3f8ea64ef408>

✖ Fail with error 'SELF'

## 6.8 Linear scaling locks allow a user to withdraw an amount of tokens each block scaling linearly from the startEmission date to the endEmission date

Status: tested and verified 



21:43

23. getWithdrawableTokens ↓

\_lockID (uint256)

7

Query

↳ uint256

[ getWithdrawableTokens(uint256) method Response ]  
» uint256 : 199999987418971310135

21:44

23. getWithdrawableTokens ↓

\_lockID (uint256)

7

Query

↳ uint256

[ getWithdrawableTokens(uint256) method Response ]  
» uint256 : 199999996300841876533



21:45 (unlock date)

23. getWithdrawableTokens ↓

\_lockID (uint256)

7

Query

└─ uint256


[ getWithdrawableTokens(uint256) method Response ]

➤ uint256 : 200000000001621279200

6.8.1 Withdraw Lock ID 7 after 21:45

Tx: <https://kovan.etherscan.io/tx/0xb0c9eb9ac85e113a142a68e314ade5c521f3db3e5d3573c2939479f7a2cced60>

**6.9 Unicrypt (Deployer) is not able to withdraw locked tokens**

**Status:** tested and verified 

6.9.1 Withdraw of Lock ID 7 with deployer


Tx: <https://kovan.etherscan.io/tx/0xf0ab0b6aca6d47838ca8c2c3d0345c731a3a7d4e027b4db37cedf7b13020ace8>

 Fail with error 'OWNER'

Contract [0x887f0582ef6db1aec2c18afe77989e059a0ed8a5](#)  

└─ Warning! Error encountered during contract execution [Reverted] 

**6.10 The smart contract is coded according to the newest standards and in a secure way.**

**Status:** tested and verified 

## 7. Unit Tests

### Token Vesting

- ✓ Should test contract deployment (214ms)
- ✓ Should test setting the Migrator contract (89ms)
- ✓ Should prevent bob from setting migrator contract (77ms)
- ✓ Should test setting the BlackList contract (64ms)
- ✓ Should prevent bob from setting BlackList contract (41ms)
- ✓ Should change Fee parameters (80ms)
- [ ✓ Should prevent bob from setting Fees

### Token Vesting

- ✓ Should test contract deployment (86ms)
- ✓ Should test setting the Token whitelisters
- ✓ Should prevent bob from setting a Token whitelister
- ✓ Should test removing a Token whitelisters
- ✓ Should pay for free token locks
- ✓ Fail to pay for a token already listed for free locking
- ✓ editZeroFeeWhitelist test

### Token Vesting

- ✓ Should test contract deployment (129ms)
- ✓ Should test a small Lock (85ms)
- ✓ Should test Lock withdrawl (61ms)
- ✓ Should test a LARGE Lock deposit and withdrawl that has phantom overflow (148ms)

### Token Vesting

- ✓ Generate tokens (87ms)
- ✓ Should test contract deployment (69ms)
- ✓ Test a lock Type 1 for alice with fee to fee address (48ms)
- ✓ Test a lock Type 1 for alice with fee to fee address
- ✓ Test a lock Type 1 for alice with fee to fee address (49ms)

### Token Vesting

- ✓ Should test contract deployment (131ms)
- ✓ Should test multiple locks with rounding up (81ms)

```
{
  lockID: 0,
  address: '0x95401dc811bb5740090279Ba06cfA8fcF6113778',
  tokensDeposited: '990',
  tokensWithdrawn: '0',
  sharesDeposited: '990',
  sharesWithdrawn: '0',
  startEmission: 0,
  endEmission: 1621187038,
  owner: '0x70997970C51812dc3A010C7d01b50e0d17dc79C8',
  condition: '0x0000000000000000000000000000000000000000'
}
{
  lockID: 1,
  address: '0x95401dc811bb5740090279Ba06cfA8fcF6113778',
  tokensDeposited: '198',
  tokensWithdrawn: '0',
  sharesDeposited: '198',
  sharesWithdrawn: '0',
  startEmission: 0,
  endEmission: 1621187038,
  owner: '0x70997970C51812dc3A010C7d01b50e0d17dc79C8',
  condition: '0x0000000000000000000000000000000000000000'
}
```

```
{
  lockID: 2,
  address: '0x95401dc811bb5740090279Ba06cfA8fcF6113778',
  tokensDeposited: '198',
  tokensWithdrawn: '0',
  sharesDeposited: '198',
  sharesWithdrawn: '0',
  startEmission: 0,
  endEmission: 1621187038,
  owner: '0x70997970C51812dc3A010C7d01b50e0d17dc79C8',
  condition: '0x0000000000000000000000000000000000000000000000000000000000000000'
}

✓ Log info
✓ Test withdrawl
```

#### Token Vesting

- ✓ Should test contract deployment (117ms)
- ✓ Should test a small Lock (59ms)
- ✓ Should test an increment lock
- ✓ Should allow another user / contract to increment another users lock

#### Token Vesting

- ✓ Should test contract deployment (66ms)
- ✓ Should test a Multi Lock which results in dust tokens being stuck (94ms)
- ✓ Should test withdrawl with no stuck dust tokens
- ✓ Should test a small lock of 200 units (45ms)

#### Token Vesting

- ✓ Should test contract deployment (69ms)
- ✓ Should test a Multi Lock (140ms)
- ✓ Should test Multi Lock withdrawls (111ms)

#### Token Vesting

- ✓ Should test contract deployment (68ms)
- ✓ Should test a Multi Lock Type 2 (125ms)
- ✓ Should test Multi Lock withdrawls Type 2 withdrawls (131ms)

#### Token Vesting

- ✓ Generate tokens (45ms)
- ✓ Should test contract deployment (69ms)
- ✓ Test a lock Type 1 for alice with fee to fee address (82ms)
- ✓ Test a withdraw for alice and fail for bob (56ms)
- ✓ Test a ownership transfer to bob (65ms)
- ✓ Generate tokens (39ms)
- ✓ Test a lock Type 1 Again
- ✓ Multi Lock (100ms)

#### Token Vesting

- ✓ Should test contract deployment (130ms)
- ✓ Should test a small Lock (56ms)
- ✓ Should rebase and affect the shares
- ✓ Should test Lock withdrawl
- ✓ Should test a Large Lock (44ms)

#### Token Vesting

- ✓ Should test contract deployment (63ms)

```
{
  lockID: 0,
  address: '0x46b142DD1E924FAB83eCc3c08e4D46E82f005e0E',
  tokensDeposited: '0',
  tokensWithdrawn: '0',
  sharesDeposited: '200',
  sharesWithdrawn: '200',
  startEmission: 0,
  endEmission: 10,
  owner: '0x70997970C51812dc3A010C7d01b50e0d17dc79C8',
  condition: '0x0000000000000000000000000000000000000000'
}
```

✓ Should test a small lock of 200 units, with two withdrawals, proving dust clearance only works on dust values (118ms)

#### Token Vesting

TIME: 1621188390

- ✓ Generate tokens (45ms)
- ✓ Should test contract deployment (68ms)
- ✓ Test a lock Type 1 for alice with fee to fee address (60ms)
- ✓ Test a withdraw for alice and fail for bob (55ms)
- ✓ Test a ownership transfer to bob (62ms)
- ✓ Generate tokens
- ✓ Large Lock (39ms)
- ✓ Tiny Lock (73ms)
- ✓ Multi Lock (118ms)
- ✓ Linear scaling locks (129ms)
- ✓ Lock with startEmission >= endEmission (83ms)
- ✓ Conditional Lock with premature withdrawl (147ms)
- ✓ Conditional Lock with failing condition and revoke withdrawl (213ms)
- ✓ Test Migration (89ms)

70 passing (6s)

## 8. 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, our auditors highly enjoyed the good documentation and unit tests. It implemented widely used and reviewed contracts from OpenZeppelin, these is greatly benefiting the security of the contract.

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.

## 9. Deployed Smart Contract

### VERIFIED

Ethereum

<https://etherscan.io/address/0xDbA68f07d1b7Ca219f78ae8582C213d975c25cAf#code>

xDAI

<https://blockscout.com/xdai/mainnet/address/0x647586E6B815D2C4a2A30dA12931c69bAda541AB/contracts>

BSC

<https://bscscan.com/address/0xeaEd594B5926A7D5FBBC61985390BaAf936a6b8d#code>

Polygon

<https://polygonscan.com/address/0x2621816bE08E4279Cf881bc640bE4089BfAf491a#code>





