Smart contract audit report for Lightstreams.network

25/04/2019

commit hash: 0b45a1b7d005fe17aa6f8c708ce58e376c084547

Lightstreams audit

1. Executive summary

Lightstreams Network audit was conducted between 12/04/2019 and 25/04/2019. Our job was to check the token vesting and distribution mechanisms which were based on 4 Solidity files.

Overall the smart contracts were created in a secure way, no major bugs were found by our team. There were some drawbacks with tests created and not all business requirements were tested. We managed to find a few minor bugs connected with current business requirements. The code itself was clearly written although it took a while to go through it. Contracts have some methods that were created in advance and not used in a current state of requirements which may obscure readability. The vestings programmatic abstraction is complicated because it requires to send confusing arguments like startDate, which equals to endDate and lock period = 0 just to make it functional. We created a small list of suggestions in order to improve current code quality and maintainability and optimize it.

The audit was performed on the latest commit from the lightstreams-network official Github repository (commit hash: 0b45a1b7d005fe17aa6f8c708ce58e376c084547).

To sum up, we are happy to say that the part of Lightstreams Network code that our team has tested is free of potential critical errors and can be safely used in the network.

For the sake of clarity we introduced the following issues symbols:

- ✓ works fine!
- ● works fine although modifications are recommended
- ✗ a valid check is missing (according to the business requirements)

The following, clickable table of contents represents a list of all the issues found.

In case of any questions regarding this audit please contact us at www.blockhunters.io.

2. Table of contents

1. EXECUTIVE SUMMARY 2

2. TABLE OF CONTENTS 3

3. SECURITY AUDIT 4

3.1. DISTRIBUTION 4

3.1.1. Known ETH Vulnerabilities 4

- ✓ Reentrancy attack 4
- ✓ Integer over / underflow 4
- ✓ Timestamp 4
- ✓ Front-running attack 4
- ✓ Insufficient gas griefing 4

3.1.2. State variables and methods 4

3.2. VESTING 8

3.2.1. Known ETH Vulnerabilities 8

- ✓ Reentrancy attack 8
- ✓ Integer over / underflow 8
- ✓ Timestamp 8
- ✓ Front-running attack 8
- ✓ Insufficient gas griefing 8

3.2.2. State variables and methods 8

3.3. AUTOMATED TOOLS TESTS 11

3.3.1. Mythril 11

3.3.2. Utils 12

4. BUSINESS REQUIREMENTS AUDIT 14

4.1. PROJECT VESTING 14

4.2. PRIVATE SALE VESTING 17

4.3. PUBLIC SALE 20

4.4. CREATING OF VESTING 21

4.5. REVOKING OF VESTING 23

4.6. UPDATE VESTING BENEFICIARY 24

5. SUGGESTIONS 25

3. Security Audit

3.1. Distribution

3.1.1. Known ETH Vulnerabilities

✓ Reentrancy attack

The methods are safe against the reentrancy attack. The state is being changed before the transaction / vesting is started.

✓ Integer over / underflow

The project uses the SafeMath library which prevents from the over / underflow.

✓ Timestamp

An attack is strictly related to the network. Since PHT is built on it’s own network - it depends on the creators.

✓ Front-running attack

Doesn’t occur. The issue exists on the decentralized markets where the transactions order matters.

✓ Insufficient gas griefing

No sub-calls available in the given contracts.

3.1.2. State variables and methods

Distribution.sol	**Status**	**Information**
State Variables		
enum Category {TEAM, SEED_CONTRIBUTORS, CONSULTANTS, OTHER, FUTURE_OFFERING}	OK	

uint256 private constant decimalFactor	OK
uint256 public PROJECT_INITIAL_SUPPLY	OK
uint256 public AVAILABLE_TEAM_SUPPLY	OK
uint256 public AVAILABLE_SEED_CONTRIBUTORS_SUPPLY	OK
uint256 public AVAILABLE_CONSULTANTS_SUPPLY	OK
uint256 public AVAILABLE_OTHER_SUPPLY	OK
uint256 public AVAILABLE_FUTURE_OFFERING_SUPPLY	OK
uint256 public PROJECT_AVAILABLE_TOTAL_SUPPLY	OK
uint256 public SALE_INITIAL_SUPPLY	OK
uint256 public SALE_AVAILABLE_TOTAL_SUPPLY	OK
uint256 private constant BONUS_MIN	OK
uint256 private constant BONUS_MAX	OK

uint256 public openingTime	OK
Methods	
constructor(uint256 _openingTime) Ownable() public	OK
function scheduleProjectVesting(address payable _beneficiary, Category _category) onlyOwner public payable	OK
function schedulePrivateSaleVesting(address _beneficiary, uint256 _bonus) onlyOwner public payable	OK
function transferToPublicSale(address payable _beneficiary) onlyOwner public payable	OK

function _validateScheduleProjectVesting(
  address _beneficiary,
  uint256 _amount,
  uint256 _categorySupply
)
internal view
	OK

function _validatePrivateSaleVesting(
  address _beneficiary,
  uint256 _amount,
  uint256 _bonus
)
internal view
	OK

function _validatePublicSaleTransfer(address _beneficiary, uint256 _amount) internal view	OK

function projectSupplyDistributed() public view returns (uint256)	OK
function saleSupplyDistributed() public view returns (uint256)	OK

3.2. Vesting

3.2.1. Known ETH Vulnerabilities

✓ Reentrancy attack

No possible reentrancy attack. A status is being updated before transaction is performed, thus contract is secured.

✓ Integer over / underflow

The project uses the SafeMath library which prevents from the over / underflow.

✓ Timestamp

An attack is strictly related to the network. Since PHT is built on it’s own network - it depends on the creators.

✓ Front-running attack

Doesn’t occur. The issue exists on the decentralized markets where the transactions order matters.

✓ Insufficient gas griefing

Using address.transfer() function puts on at risk of a 'gas griefing' attack. If the fallback function of the recipient contains instructions that use more than 2100 gas, then the whole transaction will revert. This vulnerability is not present in the Vesting smart contracts, as neither non-owner users are now allowed to pass their own recipient addresses nor the withdrawals are processed in a deterministic loop.

3.2.2. State variables and methods

Vesting.sol	**Status**	**Information**
State Variables		
uint256 public revokedAmount;	OK	

struct VestingSchedule {
uint256 startTimestamp;
uint256 endTimestamp;
uint256 lockPeriod;
uint256 balanceInitial;
uint256 balanceClaimed;
uint256 balanceRemaining;
uint256 bonusInitial;
uint256 bonusClaimed;
uint256 bonusRemaining;
bool revocable;
bool revoked;
}
	OK

mapping (address => VestingSchedule) public vestings	OK
Methods	
function setVestingSchedule(address _beneficiary, uint256 _amount, uint256 _bonus, uint256 _startTimestamp, uint256 _endTimestamp,	OK

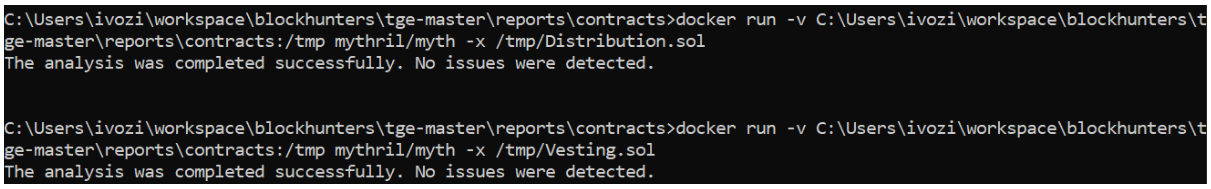
uint256 _lockPeriod, bool _revocable) internal	
function withdraw(address payable _beneficiary) public	OK
function revokeVestingSchedule(address payable _beneficiary) onlyOwner public	OK
function updateVestingBeneficiary(address _beneficiary, address _nextBeneficiary) onlyOwner public	OK
function transferRevokedTokens(address payable _recipient, uint256 _amount) public onlyOwner	OK
function _calculateTotalAmountVested(uint256 _startTimestamp, uint256 _endTimestamp, uint256 _balanceInitial) internal view returns (uint256 _amountVested)	OK
function _calculateBalanceWithdrawal(uint256 _startTimestamp, uint256 _endTimestamp, uint256 _lockPeriod, uint256 _balanceInitial, uint256 _balanceRemaining, uint256	OK

_balanceClaimed) internal view returns(uint256 _amountReleasable)		
function _calculateBonusWithdrawal(uint256 _startTimestamp, uint _endTimestamp, uint256 _lockPeriod, uint256 _balanceInitial, uint256 _bonusRemaining) internal view returns(uint256 _amountWithdrawable)	OK – fixed (Warning)	A method contained a bug, but fix was applied by subtracting bonusClaimed from amountWithdrawablePerLockPeriod. (In case of withdraw - It allows user to withdraw more bonus amount than it should.)

3.3. Automated tools tests

3.3.1. Mythril

We've used one of the most popular automated to on both Distribution.sol and Vesting.sol files. It didn't return any issue and not even a notice. The code semantincs is correct.

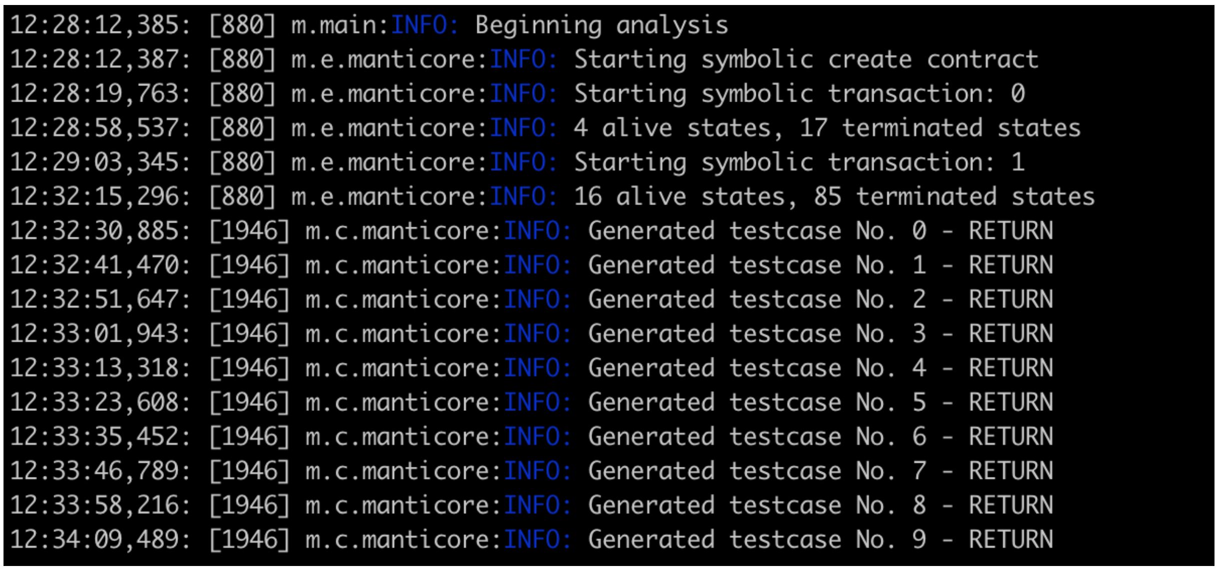


```
C:\Users\ivozi\workspace\blockhunters\tge-master\reports\contracts>docker run -v C:\Users\ivozi\workspace\blockhunters\t
ge-master\reports\contracts:/tmp mythril/myth -x /tmp/Distribution.sol
The analysis was completed successfully. No issues were detected.


C:\Users\ivozi\workspace\blockhunters\tge-master\reports\contracts>docker run -v C:\Users\ivozi\workspace\blockhunters\t
ge-master\reports\contracts:/tmp mythril/myth -x /tmp/Vesting.sol
The analysis was completed successfully. No issues were detected.
```

3.3.2. Manticore

Manticore is yet another automated testing tool for smart contract which operates on a most known attacks and also creates test cases with random generated input. Both Distribution and Vesting were tested. More than 90 test cases were created in a process of validation. None of them returned any warning, nor error.



```
12:28:12,385: [880] m.main:INFO: Beginning analysis
12:28:12,387: [880] m.e.manticore:INFO: Starting symbolic create contract
12:28:19,763: [880] m.e.manticore:INFO: Starting symbolic transaction: 0
12:28:58,537: [880] m.e.manticore:INFO: 4 alive states, 17 terminated states
12:29:03,345: [880] m.e.manticore:INFO: Starting symbolic transaction: 1
12:32:15,296: [880] m.e.manticore:INFO: 16 alive states, 85 terminated states
12:32:30,885: [1946] m.c.manticore:INFO: Generated testcase No. 0 - RETURN
12:32:41,470: [1946] m.c.manticore:INFO: Generated testcase No. 1 - RETURN
12:32:51,647: [1946] m.c.manticore:INFO: Generated testcase No. 2 - RETURN
12:33:01,943: [1946] m.c.manticore:INFO: Generated testcase No. 3 - RETURN
12:33:13,318: [1946] m.c.manticore:INFO: Generated testcase No. 4 - RETURN
12:33:23,608: [1946] m.c.manticore:INFO: Generated testcase No. 5 - RETURN
12:33:35,452: [1946] m.c.manticore:INFO: Generated testcase No. 6 - RETURN
12:33:46,789: [1946] m.c.manticore:INFO: Generated testcase No. 7 - RETURN
12:33:58,216: [1946] m.c.manticore:INFO: Generated testcase No. 8 - RETURN
12:34:09,489: [1946] m.c.manticore:INFO: Generated testcase No. 9 - RETURN
```

3.3.3. Utils

Ownable.sol	Status	Information
State Variables		
address private _owner	OK	
Methods		
constructor () internal	OK	
function owner() public view returns (address)	OK	
modifier onlyOwner()	OK	
function isOwner() public view returns (bool)	OK	

function transferOwnership(address newOwner) public onlyOwner	OK	
function _transferOwnership(address newOwner) internal	OK	

SafeMath.sol library	Status	Information
Methods		
function mul(uint256 a, uint256 b) internal pure returns (uint256)	OK	
function div(uint256 a, uint256 b) internal pure returns (uint256)	OK	
function sub(uint256 a, uint256 b) internal pure returns (uint256)	OK	
function add(uint256 a, uint256 b) internal pure returns (uint256)	OK	
function mod(uint256 a, uint256 b) internal pure returns (uint256)	OK	

4. Business requirements audit

4.1. Project vesting

New vesting, the allocation is only possible if the corresponding category pool cap was not reached yet.	OK	
01_distribution.('The owner can not create an allocation from the team supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded, real reason not checked)
01_distribution.('The owner can not create an allocation from the consultant supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded, real reason not checked)
01_distribution.('The owner can not create an allocation from the other supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded, real reason not checked)
01_distribution.('The owner can not create an allocation from the future offering supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded, real reason not checked)
01_distribution.('The owner can not create an allocation from public sale supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded, real reason not checked)
02_teamMember.('The owner can not create an allocation from the team supply greater than the amount allocated to it')	OK	
Distribution period already started (the start time is decided on contract deployment)	OK	

01_distribution.('The owner can not create an allocation before allocation distribution starts')	OK	
Only contract owner can schedule new vesting	OK	
01_distribution.('Only the owner can create an allocation ')	OK	
02_teamMember.('The owner can create an allocation from the team supply')	OK	
The beneficiary cannot have any other vesting already scheduled	OK	
No tests	Minor	No tests concerning the given statement
Available pools caps were updated	OK	
03_seedContributor.('The owner can create an allocation from the seed contributors supply')	OK	
Only beneficiary is allowed call withdraw() method to release his tokens	OK	
02_teamMember.('Only beneficiary itself can release its vested amount')	OK	
02_teamMember.('The team member can release two more vested periods')	OK	
Vesting is created with the total amount send on the transaction message	OK	
02_teamMember.('The owner can create another allocation from a new team member')	OK	

Team member vestings		
Tokens are vested in 24 periods of 30 days in equal proportions	Minor	The business requirement not met – last period vested amount is higher if vesting amount is uneven.
Only owner can revoke vesting	OK	
02_teamMember.(''The owner can revoke a team member vesting'')	Minor	Only checked if an owner can, not if somebody else can't. Tests should be more specific for all conditions
The first vesting period is due on the very first day	OK	
02_teamMember.('The team member can release first vested amount on very first day')	OK	
Seed contributor vestings		
Tokens are vested in 5 periods of 30 days in equal proportions	Minor	If the total vesting is not divisible by five, the remainder will be included in the last period withdrawal, making the last one not equal to the other.
Vesting cannot be revoked	OK	
03_seedContributor.('The owner cannot revoke a seed contributor vesting')	OK	
The first vesting period is due on the very first day	OK	
03_seedContributor.('The seed contributor can release first vested amount')	OK	

Others, Consultant, Future offering vesting - The beneficiary receives the tokens directly without vesting schedules	OK

4.2. Private sale vesting

New vesting, the allocation is only possible if sale pool cap was not reached yet	OK	
01_distribution.(‘ The owner can not create an allocation from public sale supply greater than the amount allocated to it’)	OK	
Distribution period already started (the start time is decided on contract deployment)	OK	
No tests	Minor	No tests concerning the given statement
Only contract owner can schedule new vesting	OK	
No tests	Minor	No tests concerning the given statement
Vesting is performed in 5 periods of 30 days each for the total amount, and 2 periods of 30 days for the bonus amount. Each period corresponds to 20% of the total amount excluding the bonus amount.	OK	
04_privateSaleContributor.('The private sale contributor can release 20% of vested bonus')	OK	
04_privateSaleContributor.('The private sale contributor can release remaining vested bonus')	OK	

04_privateSaleContributor.('The private sale contributor can release first vested amount')	OK	
04_privateSaleContributor.('The private sale contributor can release their vested amount but not bonus')	OK	
The first vesting period is due on the very first day	OK	
04_privateSaleContributor.('The private sale contributor can release first vested amount')	OK	
Bonus can represent a maximum of 35% of the amount allocated to the beneficiary	OK	
04_privateSaleContributor.('The owner cannot create an allocation for private contributor with more than 45% bonus')	OK	
04_privateSaleContributor.('The owner can create an allocation for private contributors from sale supply with vesting an with bonus')	OK	
Maximum bonus level of exactly 35%	OK	
The beneficiary can not have any other vesting already scheduled	OK	
No tests	Minor	No tests concerning the given statement
Sale cap was updated	OK	
04_privateSaleContributor.('The owner can create an allocation for private contributors from sale supply with vesting an without bonus')	OK	
04_privateSaleContributor.('The owner can create an allocation for private	OK	

contributors from sale supply with vesting an with bonus')		
Vesting schedule amount corresponds to the total amount on the transaction message minus the bonus.	OK	
04_privateSaleContributor.('The owner can create an allocation for private contributors from sale supply with vesting an without bonus')	OK	
Only beneficiary is allowed to call withdraw() method to release his tokens	OK	
No test	Minor	No tests concerning the given statement
Vesting cannot be revoked	OK	
04_privateSaleContributor.('The owner cannot revoke a private contributor vesting')	OK	

4.3. Public sale

Only possible if sale pool cap was not reached yet	OK	
No tests	Minor	No tests concerning the given statement
Distribution period already started (the start time is decided on contract deployment)	OK	
No tests	Minor	No tests concerning the given statement
Only the contract owner can distribute tokens to public sale contributors	OK	
No tests	Minor	No tests concerning the given statement
All tokens including are sent immediately to the beneficiary without getting locked in the smart contract	Minor	Not tested and not true - public vesting is locked in the smart contract - it has to be withdrawn manually.
05_publicSaleContributor.('The owner can create distribute tokens to public sale contributor without vesting')	OK	
Sale cap was updated	OK	
05_publicSaleContributor.('The owner can create distribute tokens to public sale contributor who lost his PK without vesting')	OK	

The beneficiary doesn’t have to call withdraw() or interact anyhow else with the contract	Minor	Not tested and not true - public vesting is locked in the smart contract - it has to be withdrawn manually.
05_publicSaleContributor.('The owner can create distribute tokens to public sale contributor without vesting')	OK	Testing against contract balance instead of beneficiary balance
04_privateSaleContributor.('The private sale contributor can release first vested amount')	OK	
Bonus can represent a maximum of 35% of the amount allocated to the beneficiary	OK	
No tests	Minor	No tests concerning the given statement
Vesting is created with the total amount send on the transaction message	OK	
05_publicSaleContributor.('The owner can create distribute tokens to public sale contributor without vesting')	OK	
Vesting cannot be revoked	OK	
No Tests	Minor	No tests concerning the given statement

4.4. Creating of vesting

Vesting cannot be created before the starting date	Minor	Only project vestings are tested

01_distribution.('The owner can not create an allocation before allocation distribution starts')	OK	
01_distribution.('The owner cannot create distribute tokens to public sale before allocation distribution starts')	OK	
Creation of vesting schedule should come along with the total amount vested as tx value.	OK	
Vesting schedule cannot be created if there is not available tokens on the corresponding pool	OK	
01_distribution.('The owner can not create an allocation from the seed contributor supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded). Real reason not checked
01_distribution.('The owner can not create an allocation from the team supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded). Real reason not checked
01_distribution.('The owner can not create an allocation from the consultant supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded). Real reason not checked

01_distribution.('The owner can not create an allocation from the other supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded). Real reason not checked
01_distribution.('The owner can not create an allocation from the future offering supply greater than the amount allocated to it')	Minor	Invalid test, 2 conditions not met (multiple vestings for one beneficiary and pool cap exceeded). Real reason not checked
01_distribution.('The owner can not create an allocation from public sale supply greater than the amount allocated to it')	Minor	Invalid test, 4 conditions not met. Tests should be more specific.

4.5. Revoking of vesting

Revoking action is performed only on the amount tokens of tokens which were not vested, even if they were not claimed by its beneficiary.	OK	
02_teamMember.('The owner can revoke a team member vesting')	OK	
Revoked tokens are stored in the smart contract and they can be transferred only by the owner to any beneficiary account at any time.	OK	
02_teamMember.("The owner can transfer revoked tokens to any address")	OK	

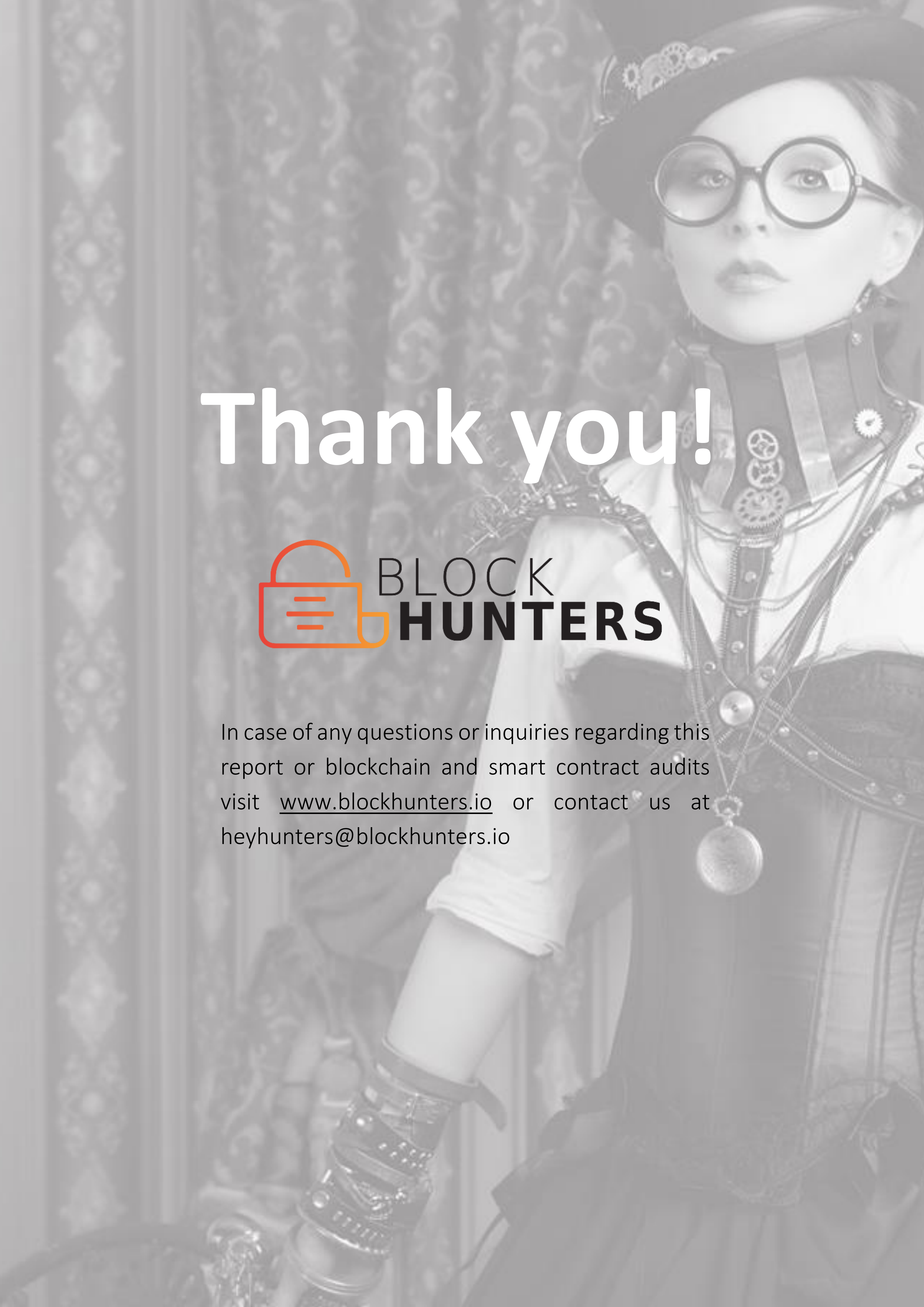
Not enough tests	Minor	Not tested if someone else can’t transfer

4.6. Update vesting beneficiary

Any vesting beneficiary address can be updated in meanwhile no tokens were withdrawn for the vesting	OK	
03_seedContributor.('The owner cannot use beneficiary with another vesting schedule to update of a seed contributor vesting')	OK	
03_seedContributor('The owner can updated beneficiary of a seed contributor vesting')	OK	
Not enough tests	Minor	No check if all the fields of the updated vesting are correct
The new beneficiary address cannot have already a scheduled vesting	OK	
02_teamMember.js.('The owner cannot use beneficiary with another vesting schedule to update of a seed contributor vesting')	OK	

5. Suggestions

- enum Categories can be changed to a structure with all the necessary fields with one boolean added that checks whether to create a scheduled vesting or to directly transfer funds to the beneficiary. It would require an additional mapping (mapping (address => CategoryStructure) public categories). Thanks to it we can define all required categories instead of making multiple “if’s” in a scheduleProjectVesting method.
- if would be a good practice to check for the address(0) in updateVestingBeneficiary method. It doesn’t generate any problem, because it can be changed again and again by the owner, yet still it is a good practice.
- Most of the tests have more than one possible revert and the test itself doesn’t check which one occurs.
- Lack of consistent usage of SafeMath methods such as div.
- _calculateBalanceWithdrawal - if (now >= _endTimestamp) can be placed on top of the function
- _validateSaleVesting bonusMin is always zero, the multiplication is redundant (for the current state of the contract)
- LogInt event should be removed if used only for debugging



Thank you!

BLOCK HUNTERS

In case of any questions or inquiries regarding this report or blockchain and smart contract audits visit www.blockhunters.io or contact us at heyhunters@blockhunters.io