Blaize.Security

July 19th, 2022 / V. 1.0

LIVE TREE
SMART CONTRACT AUDIT

TABLE OF CONTENTS

AUDIT RATING

LiveTree contract's source code was taken from the repository provided by the LiveTree team.

SCORE **9.4**/10

The scope of the project is **LiveTree** set of contracts**:**

1/ SedcSalesAndSwapOldTokens.sol

2/ SEDCToken.sol

Contracts were delivered from the repository with contract and tests.

Repository

https://github.com/livetreeShare/BlaizeAudit

SEDCToken.sol primary SHA256:

- 1ef5e6bbf0623338f6e6c874a924bfb2dfdbfe163be6bd1d5c71e956047fabe4

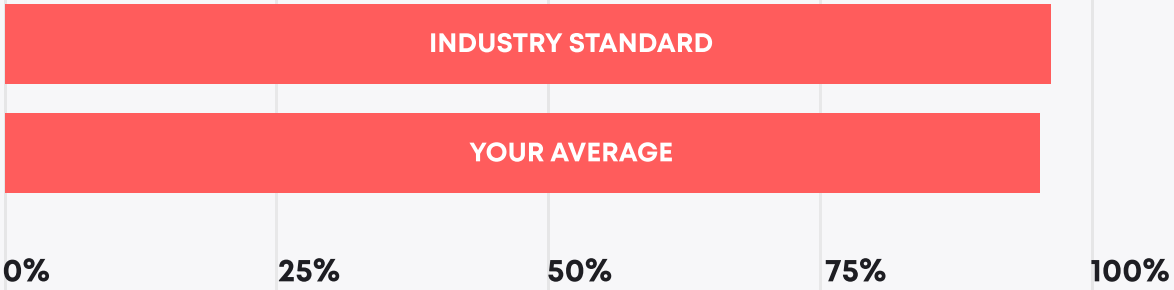
SedcSalesAndSwapOldTokens.sol final SHA256:

- 6401c30945bf241faa925a2f3f76b7fd773568f1dbf43431bf32deec182a5f1a

TECHNICAL SUMMARY

In this report, we consider the security of the contracts for LiveTree protocol. Our task is to find and describe security issues in the smart contracts of the platform. This report presents the findings of the security audit of **LiveTree** smart contracts conducted between **February 07th, 2022 - April 14th, 2022** (the first iteration) and **June 23rd, 2022 - July 15th, 2022** (the second iteration)

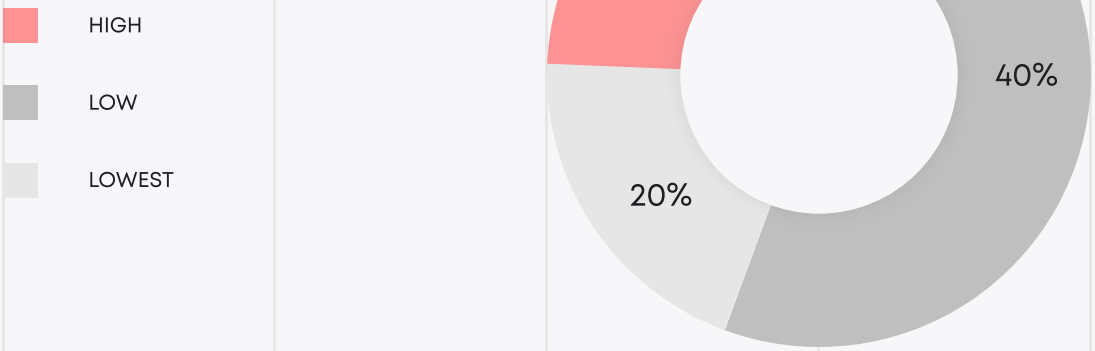
Testable code



The testable code is 94.65%, which corresponds to the industry standard of 95%.

The scope of the audit includes the unit test coverage, that bases on the smart contracts code, documentation and requirements presented by the LiveTree team. Coverage is calculated based on the set of Truffle framework tests and scripts from additional testing strategies. Though, in order to ensure a security of the contract Blaize.Security team recommends the LiveTree team put in place a bug bounty program to encourage further and active analysis of the smart contracts.

THE GRAPH OF VULNERABILITIES DISTRIBUTION:



The table below shows the number of found issues and their severity. A total of 5 problems were found. 3 issues were fixed or verified by the LiveTree team.

	FOUND	FIXED/VERIFIED
Critical	0	0
High	1	1
Medium	0	0
Low	3	2
Lowest	1	0

SEVERITY DEFINITION

Critical

A system contains several issues ranked as very serious and dangerous for users and the secure work of the system. Needs immediate improvements and further checking.

High

A system contains a couple of serious issues, which lead to unreliable work of the system and migh cause a huge information or financial leak. Needs immediate improvements and further checking.

Medium

A system contains issues which may lead to mediumfinancial loss or users' private information leak. Needs immediate improvements and further checking.

Low

A system contains several risks ranked as relatively small with the low impact on the users' information and financial security. Needs improvements.

Lowest

A system does not contain any issue critical to the secure work of the system, yet is relevant for best

AUDITING STRATEGYAND TECHNIQUES APPLIED \ PROCEDURE

We have scanned this smart contract for commonly known and more specific vulnerabilities:

- Unsafe type inference;
- Timestamp Dependence;
- Reentrancy;
- Implicit visibility level;
- Gas Limit and Loops;
- Transaction-Ordering Dependence;
- Unchecked external call - Unchecked math;
- DoS with Block Gas Limit;
- DoS with (unexpected) Throw;
- Byte array vulnerabilities;
- Malicious libraries;
- Style guide violation;
- ERC20 API violation;
- Uninitialized state/storage/ local variables;
- Compile version not fixed.

Procedure

In our report we checked the contract with the following parameters:

- Whether the contract is secure;
- Whether the contract corresponds to the documentation;
- Whether the contract meets best practices in efficient use of gas, code readability;

Automated analysis:

Scanning contract by several public available automated analysis tools such as Mythril, Solhint, Slither and Smartdec. Manual verification of all the issues found with tools.

Manual audit:

Manual analysis of smart contracts for security vulnerabilities. Checking smart contract logic and comparing it with the one described in the documentation.

EXECUTIVE SUMMARY

The protocol represents a standard ERC20 token contract with additional features and an exchange contract for swapping old token to new one.

There were no critical issues found during audit. There was one high issue, connected to the correctness of ETH transfer. Other issues were connected to the additional variables validations, unnecessary SafeMath usage and improving of code readability.

High issues and low issues, related to parameters validation, were successfully fixed by Live Tree team. The overall code security is high enough.

	RATING
Security	9.5
Gas usage and logic optimization	9
Code quality	9.5
Test coverage	9.8
Total	9.4

COMPLETE ANALYSIS

HIGH	✔ Resolved

Verify that the owner is able to receive Ether.

SedcSalesAndSwapOldTokens.sol: line 322.
Function “transfer” which is used to transfer Ether to the owner, sends Ether with fixed amount of gas(2300 gas units), which might not be enough in case owner is a smart-contract with additional logic in function “receive”.

Recommendation:

Verify that the owner is able to receive Ether with 2300 gas units. In case, owner is a smart-contract, it is recommended to use function “call” for transferring Ether. Example:
(bool result,) = payable(owner).call{value: address(this).balance}();
require(result);

LOW	✔ Resolved

Verify “bonusSwapLockRate” before assigning.

SedcSalesAndSwapOldTokens.sol: line 397.
Values from array _amount should be validated not to be greater than swapLockRate. This is essential due to subtractions in lines 425 and 464). Value is marked as low because only admin can set these values, however such security checks are still important for contract correct operation.

Recommendation:

Validate that values from array _amount are less than swapLockRate.

LOW	✔ Resolved

Wrong variable is passed to event.

SedcSalesAndSwapOldTokens.sol: function setSwapRate(). Function changes variable “swapRate”, however a variable “distRate” is passed to event(Line 342).

Recommendation:

Pass variable “swapRate” to event.

LOW	✔ Unresolved

SafeMath usage can be omitted.

SedcSalesAndSwapOldTokens.sol: lines 249-250, 424-425, 444, 463-464 Contracts set utilizes Solidity 0.8.x, which has built in support of overflow and underflow warnings, thus SafeMath library can be omitted for gas savings and code simplification.

Recommendation:

Remove SafeMath library.

LOWEST	✔ Unresolved

Unclear values.

SedcSalesAndSwapOldTokens.sol: lines 170-176, 207-223, 249-250 Using big values may be unclear while reading code and in might be easy to make a mistake in such values. In order to provide a better code readability, it is recommended to user ether notation. For example, variable “distRate” (Line 171) can be assigned to value “250000 ether” instead of “250000000000000000000000”.

Recommendation:

Use ether notation for big values.

	SedcSalesAndSwapOldTokens.sol	SEDCToken.sol
✓ Re-entrancy	Pass	Pass
✓ Access Management Hierarchy	Pass	Pass
✓ Arithmetic Over/Under Flows	Pass	Pass
✓ Delegatecall Unexpected Ether	Pass	Pass
✓ Default Public Visibility	Pass	Pass
✓ Hidden Malicious Code	Pass	Pass
✓ Entropy Illusion (Lack of Randomness)	Pass	Pass
✓ External Contract Referencing	Pass	Pass
✓ Short Address/ Parameter Attack	Pass	Pass
✓ Unchecked CALL Return Values	Pass	Pass
✓ Race Conditions / Front Running	Pass	Pass
✓ General Denial Of Service (DOS)	Pass	Pass
✓ Uninitialized Storage Pointers	Pass	Pass
✓ Floating Points and Precision	Pass	Pass
✓ Tx.Origin Authentication	Pass	Pass
✓ Signatures Replay	Pass	Pass
✓ Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES BY LIVETREE

Contract: SedcSalesAndSwapOldTokens

Sales part

- ✓ should fail if an address not whitelisted tries to sends eth (78ms)
- ✓ contract should receive ether (660ms)
- ✓ should return expected value for 1 ETH (436ms)
- ✓ should return presale distribution value for 1 eth (472ms)
- ✓ should return mainsale distribution value for 1 eth (384ms)
- ✓ owner should withdraw eth from contract (407ms)
- ✓ only owner should withdraw eth from contract (417ms)
- ✓ should transfer 50% locked tokens (361ms)
- ✓ should fail if 0% is set as distribution Lock rate (131ms)

Swap part

- ✓ should return exact value of SED token sent (126ms)
- ✓ should whitelist addresses (68ms)
- ✓ should approve tokens for swap contract (96ms)
- ✓ should make swap contract manager
- ✓ should subtract token sent from owner (52ms)
- ✓ should set sedc and sed contract in swap contract as new and old tokens (51ms)
- ✓ should set bonus rate for swap (49ms)
- ✓ should return swap value for presale (348ms)
- ✓ should return swap value for mainsale (249ms)
- ✓ should return swap value for marketing (241ms)
- ✓ should fail if an address not whitelisted tries to swap tokens (166ms)

- ✓ should transfer 50% of locked amount to another address (179ms)
- ✓ should fail if 0% is set for swap Lock rate (141ms)

Contract: SEDCToken

Token contract

- ✓ Should return the token name
- ✓ Should transfer tokens to another address
- ✓ Should approve spender address (65ms)
- ✓ should set msg.sender as owner
- ✓ should mint token

27 passing (19s)

TEST COVERAGE RESULTS

BY LIVETREE TEAM

FILE	% STMTS	% BRANCH	% FUNCS
SedcSalesAndSwapOldTokens.sol	44.92	36.59	50
SEDCToken.sol	47.89	34.62	45.45
All files	**46.41**	**35.61**	**47.73**

CODE COVERAGE AND TEST RESULTS FOR ALL FILES BY BLAIZE SECURITY TEAM

Contract: SedcSalesAndSwapOldTokens

- ✓ Should delete addresses from whitelist (101ms)
- ✓ Shouldn't add addresses to whitelist when addresses count bigger than 255 (39ms)
- ✓ Shouldn't delete addresses from whitelist when addresses count bigger than 255
- ✓ Should get addresses and count (56ms)
- ✓ Shouldn't get addresses from when from bigger than addresses length
- ✓ Should get addresses from (54ms)
- ✓ Should get contract token balance
- ✓ Shouldn't set 0 dist rate
- ✓ Should set swap rate
- ✓ Shouldn't set 0 swap rate
- ✓ Shouldn't set 0 swap lock and dist lock
- ✓ Should set swap lock and dist lock (45ms)
- ✓ Should set bonus dist rate (76ms)
- ✓ Should set bonus swap rate (94ms)
- ✓ Should set bonus dist lock (77ms)
- ✓ Should set bonus swap lock (74ms)
- ✓ Shouldn't set bonus (dist rate, swap rate, dist lock, swap lock) when addresses count != amount count (54ms)
- ✓ Shouldn't set bonus (dist rate, swap rate, dist lock, swap lock) when addresses count bigger than 255 (146ms)
- ✓ Should withdraw tokens from contract by owner (93ms)
- ✓ Shouldn't delete address from whitelist when addresses count bigger than 255 (298ms)

- ✓ Should swap token when token balance bigger than allowance (232ms)
- ✓ Should not swap token when SWAP contract have not manager role (271ms)
- ✓ Should not swap token when SWAP contract have not approve for SEDC token transfer (250ms)
- ✓ Should not swap token when SWAP contract have not approve for SED token transfer (217ms)
- ✓ Should get swap token amount (176ms)
- ✓ Shouldn't get swap token amount when sender is not whitelisted (68ms)
- ✓ Should receive approval (218ms)
- ✓ Should not receive approval when SWAP contract have not approve for SED token transfer (248ms)
- ✓ Should not receive approval when SWAP contract have not approve for SEDC token transfer (191ms)
- ✓ Should not receive approval when argument contract is not old token (146ms)
- ✓ Should not receive approval when address is not whitelisted (105ms)
- ✓ Should receive approval when allowance bigger than token balance (219ms)
- ✓ Should not receive approval when SWAP contract have not manager role (178ms)
- ✓ Should distribute token when ether value >= 30e18 (297ms)
- ✓ Should distribute token when ether value <= 1e18 (263ms)
- ✓ Should distribute token when ether value >= 1e18 (271ms)
- ✓ Should distribute token when ether value >= 3e18 (258ms)
- ✓ Should distribute token when ether value >= 5e18 (265ms)
- ✓ Should distribute token when ether value >= 7e18 (261ms)

- ✓ Should distribute token when ether value >= 25e18 (268ms)
- ✓ Should not distribute token when SWAP contract have not manager role (106ms)
- ✓ Should not distribute token when SWAP contract have not approve for SEDC token transfer (105ms)

Contract: SEDCToken

- ✓ Should not transfer token when active off
- ✓ Should return balance of address when active off for admin (47ms)
- ✓ Should not return balance of address when active off for user (65ms)
- ✓ Should return addresses amount (43ms)
- ✓ Should not transfer from when allowance smaller than amount (59ms)
- ✓ Should not transfer from when balance smaller than amount (51ms)
- ✓ Should increase approval (401ms)
- ✓ Should decrease approval (172ms)
- ✓ Should decrease approval to zero when decrease value bigger than approve (121ms)
- ✓ Should increase total supply via mint (93ms)
- ✓ Should decrease total supply via burn (91ms)
- ✓ Should not burn when burn value bigger than balance (42ms)
- ✓ Should redeem to owner (131ms)
- ✓ Should transfer when lock is off (76ms)
- ✓ Should not transfer when (sender locked amount * scaleLock) bigger than ((sender balance - transfer amount) * 1000000) (326ms)
- ✓ Should not transfer with 0 amount (55ms)
- ✓ Only owner can set active (40ms)
- ✓ Only admin can set locked amounts (385ms)
- ✓ Should not set locked amounts when addresses amount =! value amounts (49ms)
- ✓ Should get locked amounts (101ms)

90 passing (48s)

TEST COVERAGE RESULTS

BY BLAIZE SECURITY TEAM

FILE	% STMTS	% BRANCH	% FUNCS
SedcSalesAndSwapOldTokens.sol	95.93	92.68	96.15
SEDCToken.sol	90.14	88.46	90.91
All files	**94.65**	**93.62**	**93.53**

DISCLAIMER

The information presented in this report is an intellectual property of the customer including all presented documentation, code databases, labels, titles, ways of usage as well as the information about potential vulnerabilities and methods of their exploitation. This audit report does not give any warranties on the absolute security of the code. Blaize.Security is not responsible for how you use this product and does not constitute any investment advice.

Blaize.Security does not provide any warranty that the working product will be compatible with any software, system, protocol or service and operate without interruption. We do not claim the investigated product is able to meet your or anyone else requirements and be fully secure, complete, accurate and free of any errors and code inconsistency.

We are not responsible for all subsequent changes, deletions and relocations of the code within the contracts that are the subjects of this report.

You should perceive Blaize.Security as a tool which helps to investigate and detect the weaknesses and vulnerable parts that may accelerate the technology improvements and faster error elimination.