Adoriasoft

ADORABLE SOFTWARE ON DISTRIBUTED LEDGER TECHNOLOGIES

A team of engineers with a primary focus
on blockchain and distributed ledger technologies

TABLE OF CONTENTS

INTRODUCTION TO THE AUDIT

Intro

Adoriasoft was contracted to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review conducted between August 29, 2021 – September 3, 2021.

Name	Smart Contract Code Review and Security Analysis Report for Cumulative Merkle Drop
Platform	Ethereum / Solidity
Repository	https://github.com/1inch/cumulative-merkle-drop
Commit	f2511f0a02ed2631d6c2630bd188a45281909614
Timeline	August 29, 2021 – September 3, 2021

Scope

The scope of the project is Merkle-tree-based token distribution smart contracts:

- CumulativeMerkleDrop;
- CumulativeMerkleDrop128;
- CumulativeMerkleDrop160.

which can be found on Github by the link below:

https://github.com/1inch/cumulative-merkle-drop/tree/master/contracts

Commit version: f2511f0a02ed2631d6c2630bd188a45281909614

We have scanned smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

SECURITY ASSESSMENT PRINCIPLES

Classification of issues

Severity Definitions	
Level	Description
Critical	Easy to use by 3rd parties. Can lead to assets and data loss. Needed to be fixed.
High	Hardly to use by 3rd parties. Also, can lead to assets and data loss. Needed to be fixed.
Medium	Can't lead to assets and data loss. Also, needed to be fixed.
Low, Comments	Can't lead to assets and data loss. Related to unused, outdated code, code style, etc.

OpenZeppelin library overview

The project uses OpenZeppelin library v 4.2.0, which is considered safe.

Ownable contract overview

Ownable contract is taken from OpenZeppelin library, which is considered as safe.

IERC20 interface overview

IERC20 interface is taken from OpenZeppelin library which is considered as safe.

SafeERC20 contract overview

SafeERC20 contract is taken from OpenZeppelin library, which is considered as safe.

CumulativeMerkleDrop contract overview

CumulativeMerkleDrop contract inherits Ownable, ICumulativeMerkleDrop contracts. It also uses SafeERC20 library for token operations.

CumulativeMerkleDrop contract constructor set:

- **token** to **token_**

CumulativeMerkleDrop contract has 9 functions:

claim, cumulativeClaimed, setMerkleRoot, merkleRoot, token, owner, renounceOwnership, transferOwnership, **_verifyAsm.**

- **claim** is an external function - transfers tokens from contract to receiver;
- **cumulativeClaimed** is a public function - returns how many tokens are already claimed;
- **setMerkleRoot** is an external function - sets new Merkle root;
- **merkleRoot** is a public function - returns the Merkle tree root;
- **token** is a public function - returns token address which will be claimed;
- **owner** is a public function - returns the address of owner;
- **renounceOwnership** is a public function - renounces the owner’s privileges;
- **transferOwnership** is a public function - transfers owner's privileges to given address;

- **_verifyAsm** is a private function - verifies the leaf for a given Merkle tree root.

CumulativeMerkleDrop128 Contract overview

CumulativeMerkleDrop128 contract inherits Ownable, ICumulativeMerkleDrop128 contracts. It also uses SafeERC20 library for token operations.

CumulativeMerkleDrop128 contract constructor set:

- **token** to **token_**

CumulativeMerkleDrop128 contract has 10 functions:

claim, cumulativeClaimed, setMerkleRoot, merkleRoot, token, owner, renounceOwnership, transferOwnership, **_verifyASM, _keccak128**

- **claim** is an external function - transfers tokens from contract to receiver;
- **cumulativeClaimed** is a public function - returns how many tokens are already claimed;
- **setMerkleRoot** is an external function - sets new Merkle root;
- **merkleRoot** is a public function - returns the Merkle tree root;
- **token** is a public function - returns token address which will be claimed;
- **owner** is a public function - returns the address of the owner;
- **renounceOwnership** is a public function - renounces owner’s privileges;
- **transferOwnership** is a public function - transfers owner's privileges to

given address;

- **_verifyAsm** is a private function - verifies the leaf from a given Merkle tree root;
- **_keccak128** is an internal function - gets a 128 bit hash from keccak256 hash.

CumulativeMerkleDrop160 Contract overview

CumulativeMerkleDrop160 contract inherits Ownable, ICumulativeMerkleDrop160 contracts. It also uses SafeERC20 library for token operations.

CumulativeMerkleDrop160 contract constructor set:

- **token** to **token_**

CumulativeMerkleDrop160 contract has 10 functions:

claim, cumulativeClaimed, setMerkleRoot, merkleRoot, token, owner, renounceOwnership, transferOwnership, **_verifyASM, _keccak160**

- **claim** is an external function - transfer tokens from contract to receiver;
- **cumulativeClaimed** is a public function - returns how many tokens are already claimed;
- **setMerkleRoot** is an external function - sets new Merkle root;
- **merkleRoot** is a public function - returns the Merkle tree root;
- **token** is a public function - returns token address which will be claimed;
- **owner** is a public function - returns the address of the owner;
- **renounceOwnership** is a public function - renounces owner’s

privileges;

- **transferOwnership** is a public function - transfers owner's privileges to given address;
- **_verifyAsm** is a private function - verifies the leaf from a given Merkle tree root;
- **_keccak160** is an internal function - gets a 160 bit hash from keccak256 hash.

DETECTED ISSUES

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low, Comments

1. **setMerkleRoot** function used in each contract allows the contract owner to set a new Merkle root for token distribution overwriting the old one. If the owner sets a wrong root (without some addresses or with smaller balances) the users won't be able to claim their tokens even if they could do it with the previous root. Also, if the root was changed, old proofs become incorrect and

users have to use new ones if they haven't claimed their tokens.

2. **claim** function used in **CumulativeMerkleDrop** contract takes a parameter **merkleProof** that is an array of arbitrary length. If this array is too long there may be gas limit issues, as there are no limitations on its size.

Informational statements

1. **setMerkleRoot** that controls the list of users for token distribution is callable only from one address. Therefore, the system depends heavily on this address. It's highly recommended to use a MultiSig wallet at this address.

2. We recommend using "mapping" to save Merkle roots, enabling users to claim their tokens even if the owner sets a wrong root. It will also work as history.

Conclusion and results

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the overview section of the report.

The audit report contains all found security recommendations in the reviewed code. Critical/High/Medium issues were not found. The overall quality of reviewed contracts is good and secured.

ABOUT ADORIASOFT

ADORIASOFT IS A BLOCKCHAIN DEVELOPMENT BOUTIQUE WITH CRYPTOGRAPHY EXPERTS AND SCIENTISTS ON BOARD

Adoriasoft has been working with blockchain and other distributed ledger technologies since 2015. Our approach to building blockchain networks and distributed apps is based on practical expertise in this area as well as on our constant research of new trends, innovations and methodologies appearing in the blockchain development sector.

Distributed ledger technologies are evolving constantly, and we follow the evolution closely to stay aware of the most effective implementations. Our blockchain projects are based on solid knowledge of math, cryptography, computer science, and business processes.

For every project, we assign experienced architects, encryption experts, developers competent in working with the required technology. We follow Agile and approaches allowing to test the software components more effectively at early stages and verify their proper integration. The product is thoroughly tested both during the development and before the launch, so that we always guarantee consistent performance, security, scalability, and perfect user experience.

In our work, we approach each project individually. We usually start with a discovery sprint discussing the business goals and requirements and selecting the most effective technology to implement them. We plan and design the project components and select the tools and frameworks with the focus on the advantages they are going to bring to the product.

Our solid knowledge of blockchain technologies combined with dedicated approach and attention to minor details allow us to build great distributed apps capable of bringing the business to a new level.

CONTACTS

sales@adoriasoft.com

+1 (650) 885-9777

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 1inch. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.