soken

SMART CONTRACT SECURITY AUDIT

Leandro Lopes

Scan and check this report was posted at Soken Github

May, 2022

Website: soken.io

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws. We took into consideration smart contract based algorithms, as well. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Soken and its affiliates shall not be held responsible to you or anyone else, nor shall Soken provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Soken excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Soken disclaims all responsibility and responsibilities and no claim against Soken is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Manual analysis of smart contracts:
- Deploying smart contracts on any of the network(Ropsten/Rinkeby) using Remix IDE
- Hashes of all transaction will be recorded
- Behaviour of functions and gas consumption is noted, as well.

3. Unit Testing:
- Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
- In this phase intended behaviour of smart contract is verified.
- In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
- Gas limits of functions will be verified in this stage.

4. Automated Testing:
- Mythril
- Oyente
- Manticore
- Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Audit Details

Project Name: **Leandro Lopes**

Language: **Solidity**

Compiler Version: ^**0.8.10**

Blockchain:

Audit Scope

Leandrolopes-master /

contracts /

EIP712.sol

ERC20.sol

ERC20Internal.sol

ERC2612.sol

ERC3009.sol

IERC20.sol

IStake.sol

IVesting.sol

Ownable.sol

Recover.sol

Selfdesctruct.sol

Token.sol

Vesting.sol

Social Profiles

Project Website: **https://www.leandrolopes.de/**

Project Twitter: **https://twitter.com/IAMLEANDROLOPES**

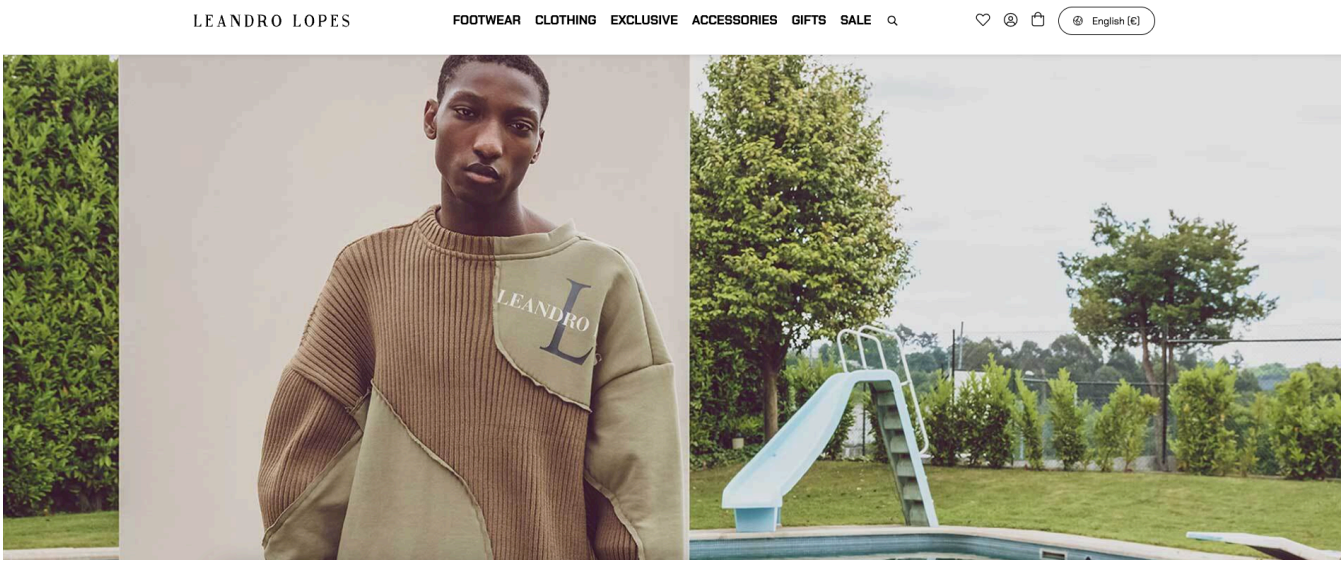
Project Facebook: **https://www.facebook.com/LeandroLopesOfficial/**

Project Instagram: **https://www.instagram.com/leandrolopesofficial/**

Project YouTube: **https://www.youtube.com/channel/UCC8lUZGUm3s8CvlWcaTS6jg**

About the project

Leandro Lopes is an internationally successful brand for high-quality fashion. The connection to Portugal is an essential part of the brand and all products are made there with care in conscientious handwork. The designer and fashion entrepreneur have remained true to his own passion for combining modern design with high-quality materials and quality workmanship in perfection.



Contract Function Details

+ EIP712.sol
- [Int] recover
- [Int] makeDomainSeparator
- [Int] recover
-
+ ERC20.sol
- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] allowance
- [Ext] transfer
- [Ext] approve
- [Ext] transferFrom
- [Ext] burn
- [Ext] burnFrom
- [Int] _ approve
- [Int] _ transfer
- [Int] _ burn

+ ERC20Internal.sol
- [Int] _approve
- [Int] _transfer

+ ERC2612.sol
- [Ext] nonces
- [Ext] permit

+ ERC3009.sol
- [Ext] authorizationState
- [Ext] transferWithAuthorization
- [Ext] receiveWithAuthorization
- [Int] _transferWithAuthorization
- [Ext] cancelAuthorization

+ IERC20.sol
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] transfer
- [Ext] allowance
- [Ext] approve
- [Ext] transferFrom

- [Ext] name
- [Ext] symbol
- [Ext] decimals

+ IStake.sol
- [Ext] getPools
- [Ext] getPoolCount
- [Ext] poolInfo
- [Ext] addStakePool
- [Ext] claim
- [Ext] claimStake
- [Ext] vestingAddress
- [Ext] tokenAddress
- [Ext] totalStakedTokens
- [Ext] rewardsAvailable
- [Ext] claim2stake
- [Ext] deposit

+ IVesting.sol
- [Ext] claimable
- [Ext] claim
- [Ext] createVest
- [Ext] massCreateVest
- [Ext] getVestings
- [Ext] getVesting

+ Ownable.sol
- [Ext] transferOwnership
- [Ext] acceptOwnership
- [Ext] renounceOwnership

+ Recover.sol
- [Ext] recover
-
+ Vesting.sol
- [Int] concat
- [Int] _addVesting
- [Ext] createVest
- [Ext] massCreateVest
- [Ext] getVestings
- [Ext] getVestingCount
- [Ext] getVesting
- [Ext] claimable
- [Int] _claimable
- [Ext] claim

- [Int] _claim
- [Ext] setStakeAddress
- [Ext] claim2stake
- [Ext] recover
- [Ext] balanceOf
- [Ext] transfer

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Completed
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Considerations

1) ERC3009.sol - Informational

Use receiveWithAuthorization instead of transferWithAuthorization when calling from other smart contracts. It is possible for an attacker watching the transaction pool to extract the transfer authorization and front-run the transferWithAuthorization call to execute the transfer without invoking the wrapper function. This could potentially result in unprocessed, locked up deposits. receiveWithAuthorization prevents this by performing an additional check that ensures that the caller is the payee. Additionally, if there are multiple contract functions accepting receive authorizations, the app developer could dedicate some leading bytes of the nonce could as the identifier to prevent cross-use.

2) ERC2612.sol - Informational

Though the signer of a Permit may have a certain party in mind to submit their transaction, another party can always front run this transaction and call permit before the intended party. The end result is the same for the Permit signer, however.
Since the ecrecover precompile fails silently and just returns the zero address as signer when given malformed messages, it is important to ensure owner != address(0) to avoid permit from creating an approval to spend “zombie funds” belong to the zero address.
Signed Permit messages are censorable. The relaying party can always choose to not submit the Permit after having received it, withholding the option to submit it. The deadline parameter is one mitigation to this. If the signing party holds ETH they can also just submit the Permit themselves, which can render previously signed Permits invalid.

Conclusion

Informational issues exist within smart contracts. Smart contracts are free from any critical, high-severity or medium-severity issues.

NOTE: Please check the disclaimer above and note, that audit makes no statements or warranties on business model, investment attractiveness or code sustainability.

Soken Contact Info

Website: www.soken.io

Mob: (+1)416-875-4174

32 Britain Street, Toronto, Ontario, Canada

Telegram: @team_soken

GitHub: sokenteam

Twitter: @soken_team