BAILSEC.IO

OFFICE@BAILSEC.IO

X: @BAILSECURITY

TG: @HELLOATBAILSEC

FINAL REPORT:

TrustSwap - Vesting

April 2024

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the audit, and the audit results may not be complete or identify all possible vulnerabilities or issues. The audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the audit report and should seek additional professional advice if needed. The audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the audit report or the smart contract.

By engaging in a smart contract audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

1. Project Details

Important:
Please ensure that the deployed contract matches the source-code of the last commit hash.

Project	TrustSwap
Website	https://trustswap.com
Type	Vesting
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/trustswap/teamfinance-contract-vesting/blob/3d24c26465e5efa47b04cbbc8b22090e644727b3/src/
Resolution 1	https://github.com/trustswap/teamfinance-contract-vesting/tree/94201e9c7df868c55340a841374407334049eb0e/src

2. Detection Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High				
Medium	3	3		
Low	2			2
Informational	2	2		
Governance				
Total	7	5		2

2.1 Detection Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low-level risk to the project or users. Nevertheless the issue should be fixed immediately
Informational	Effects are small and do not post an immediate danger to the project or users
Governance	Governance privileges which can directly result in a loss of funds or other potential undesired behavior

2. Detection

VestingFactory

The VestingFactory is a factory which allows users to deploy Vesting contracts for a fee. The fee is denominated in USD but taken in ETH and it is expected that this fee is provided as msg.value upon the Vesting contract deployment and is then sent to TrustSwap's company wallet.

The vesting logic is handled off-chain and the deployer provides a valid merkleRoot upon deployment. If for any reason, this merkleRoot is broken, all funds will remain stuck within the Vesting contract.

The fee parameters can be changed by the contract owner and additionally tokens can be whitelisted such that the deployment of Vesting contracts for these tokens will not encounter a fee.

Disclaimer: The provided resolution version contains a whitelisting mechanism. This is not part of the audit scope.

Issue_01	Malicious user can manipulate pool to bypass fee
Severity	**Medium**
Description	Currently, any user can create a vesting product by calling the createVesting function. It is expected by users to provide an appropriate msg.value with the function call which covers the corresponding fee. As already publicly acknowledged, this will not work if someone manipulates the pool ratio. This can result in low fees or inflated fees to DoS a creation call.
Recommendations	Consider using the CL oracle for this purpose.
Comments / Resolution	Resolved, a separate condition has been implemented which will alternatively fetch the fee from an oracle.

Issue_02	Unused variables
Severity	**Informational**
Description	Variables/Parts of the code which are unused will unnecessarily increase the contract size for no reason and will confuse third-party reviewers. L 16: address public usdTokenAddress;
Recommendations	Consider removing any unused code parts
Comments / Resolution	Resolved.

Vesting

The Vesting contract is a simple vesting contract which leverages OpenZeppelin's merkle tree implementation for vesting purposes. The contract is deployed with an immutable merkleRoot which then allows users to withdraw vested tokens using the correct leaf.

Additionally, it exposes an onlyOwner function that allows to stop the vesting progress for a determined leaf, transfers the vesting progress to the corresponding account and the leftover amount to the contract owner.

Contrary to most vesting contracts, the vesting progress calculation incorporates a cadence parameter, which prevents strictly linear vesting and introduces specific intervals when tokens are claimable.

The contract is designed in such a manner to be able to accommodate vesting for one or multiple parties. This solely depends on the provided merkleRoot and the leaves.

Issue_03	Tokens will be permanently stuck if account is blacklisted
Severity	**Medium**
Description	There are mainly two scenarios to interact with the contract and transfer tokens: a) claim: This function allows any arbitrary caller to claim the vesting progress for a determined account, expected the correct leaf (node) is provided b) stopVesting: This function allows the contract owner to stop the vesting, while still transferring the vesting progress (thus far) to the corresponding account and the leftover amount to the owner Both functions will not work if the corresponding account is blacklisted for the token. While it can be considered as desired that

	the first function will not work, it will be negative that the second function reverts as well. This means in the worst-case scenario, tokens will remain permanently locked in the vesting contract.
Recommendations	Consider simply transferring the vesting progress and the leftover amount to the contract owner.
Comments / Resolution	Resolved, the full unclaimed vesting progress is now transferred to the contract owner.

Issue_04	Lack of emergency withdrawal
Severity	**Medium**
Description	If the Vesting contract is deployed with an incorrect/broken merkleRoot, this can result in funds being permanently stuck within the contract because there will not be any valid leaf which matches the important parameters. In such a scenario, it is impossible to withdraw any funds.
Recommendations	Consider incorporating an emergency withdraw function.
Comments / Resolution	Resolved.

Issue_05	Vesting contract is not compatible with Rebase tokens
Severity	**Low**
Description	Rebase tokens are tokens with an increasing balance over time. If such a token is used within this vesting contract, the accrued balance will be permanently lost.
Recommendations	Consider implementing a privileged withdrawal function to be able to withdraw such additional tokens.
Comments / Resolution	Acknowledged.

Issue_06	Malicious user can frontrun claim call to DoS legit claim
Severity	**Low**
Description	The claim function is publicly callable by any arbitrary address. This means Bob can claim on behalf of Alice. Consider the following scenario: 1. Alice has a claimable amount of 100e18 tokens and invokes the claim function with exactly that amount 2. Bob frontruns Alice's call with claimAmount = 1 3. Alice's call will now revert because the remaining claimable amount is only 100e18 - 1 wei While the impact of this exploit is limited, it can become annoying for legit claimers.
Recommendations	Consider if that is an acceptable design-choice. If not, consider

	simply ensuring msg.sender = account.
Comments / Resolution	Acknowledged.

Issue_07	Unused variables
Severity	**Informational**

Description	Variables/Parts of the code which are unused will unnecessarily increase the contract size for no reason and will confuse third-party reviewers.
	
	L 7:
	
	import "openzeppelin-contracts/utils/math/Math.sol";
	
	L 25:
	
	error InvalidDates();
	
	L 35:
	
	event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

Recommendations	Consider removing any unused code parts
Comments / Resolution	Resolved.