AUDIT REPORT

SecureWise

SHIBART AI VESTING

Findings

Risk Classification	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.
Low	Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.
Informational	A vulnerability that have informational character but is not effecting any of the code

Severity	Found	Pending	Resolved
High	0	0	0
Medium	2	0	0
Low	0	0	0
Informational	0	0	0
Total	2	0	0

Scope

The following smart contracts were in scope of the audit:

- **PrescheduledTokenVesting.sol**

Contents

Overview

Token Name: PrescheduledTokenVesting

Language: Solidity

Contract Address: 0x02cf47728f19d4911FfAeb89ce6c3862078dF293

Network: Ethereum

KYC: No KYC

Website: https://shibart.ai

Twitter: https://twitter.com/shibartAI

Telegram: https://t.me/shibart

Report Date: July 2, 2023

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.

Low: Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.

Informational: A vulnerability that have informational character but is not effecting any of the code

Findings Summary

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

Findings

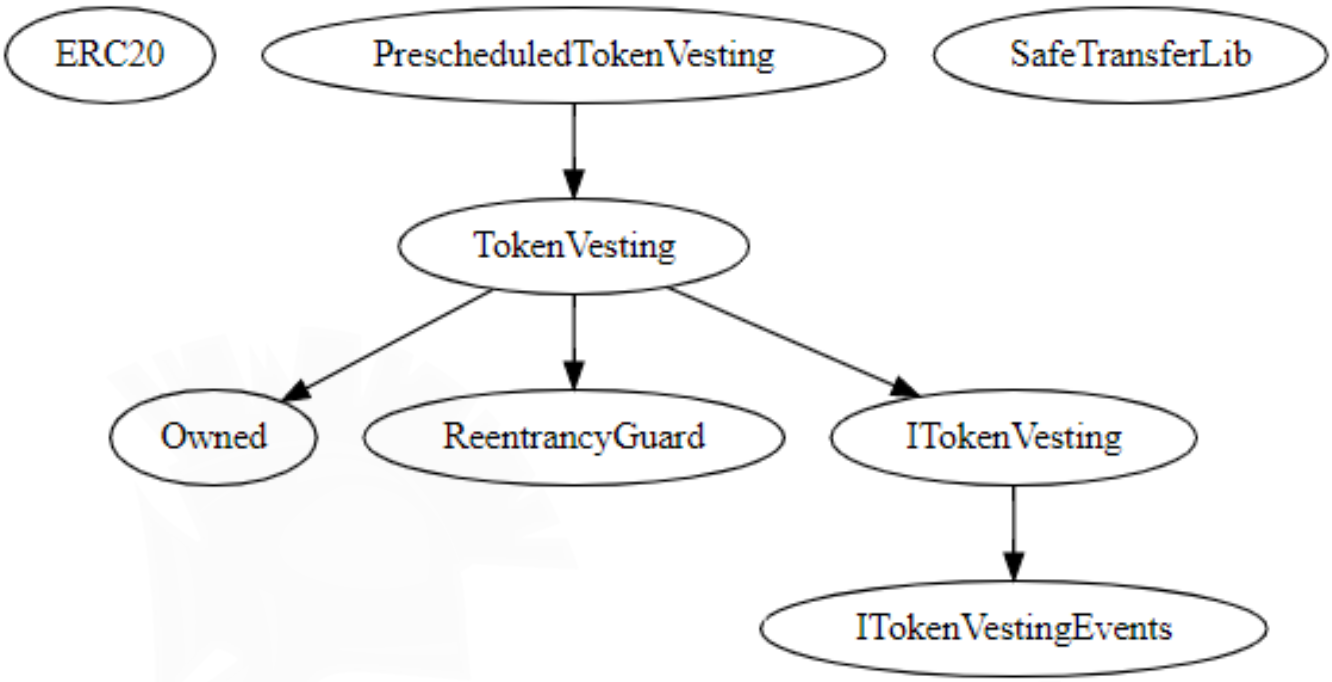
	The revoke mechanics are not compatible with tokens that implement a block list feature
	Insufficient input validation in function createVestingSchedule and extend

Page 10 for more details

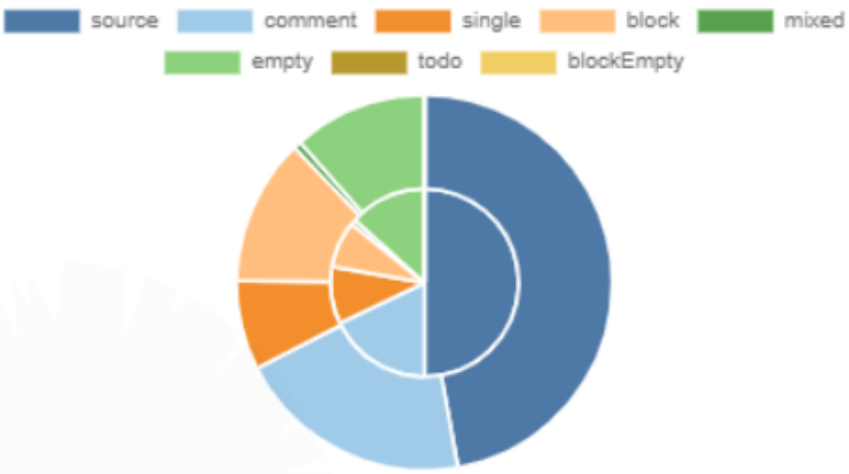
Function Privileges

ITokenVestingEvents	Interface				
ITokenVesting	Interface	ITokenVestingEvents			
└	createVestingSchedule	External !	🔴	NO !	
└	revoke	External !	🔴	NO !	
└	extend	External !	🔴	NO !	
└	withdraw	External !	🔴	NO !	
└	release	External !	🔴	NO !	
└	getVestingSchedulesCountByBeneficiary	External !		NO !	
└	getVestingIdAtIndex	External !		NO !	
└	getVestingScheduleByAddressAndIndex	External !		NO !	
└	getVestingSchedulesTotalAmount	External !		NO !	
└	getToken	External !		NO !	
└	getVestingSchedulesCount	External !		NO !	
└	computeReleasableAmount	External !		NO !	
└	getVestingSchedule	External !		NO !	
└	getWithdrawableAmount	External !		NO !	
└	computeNextVestingScheduleIdForHolder	External !		NO !	
└	getLastVestingScheduleForHolder	External !		NO !	
└	computeVestingScheduleIdForAddressAndIndex	External !		NO !	
TokenVesting	Implementation	ITokenVesting, Owned, ReentrancyGuard			
└	<Constructor>	Public !	🔴	Owned	
└	createVestingSchedule	External !	🔴	NO !	
└	revoke	External !	🔴	NO !	
└	extend	External !	🔴	NO !	
└	withdraw	External !	🔴	nonReentrant	
└	release	Public !	🔴	nonReentrant	
└	getVestingSchedulesCountByBeneficiary	External !		NO !	
└	getVestingIdAtIndex	External !		NO !	
└	getVestingScheduleByAddressAndIndex	External !		NO !	
└	getVestingSchedulesTotalAmount	External !		NO !	
└	getToken	External !		NO !	
└	getVestingSchedulesCount	Public !		NO !	
└	computeReleasableAmount	External !		NO !	
└	getVestingSchedule	Public !		NO !	
└	getWithdrawableAmount	Public !		NO !	
└	computeNextVestingScheduleIdForHolder	Public !		NO !	
└	getLastVestingScheduleForHolder	External !		NO !	
└	computeVestingScheduleIdForAddressAndIndex	Public !		NO !	
└	_createVestingSchedule	Internal 🔒	🔴		
└	_computeReleasableAmount	Internal 🔒			
└	_onlyOwner	Internal 🔒			
└	_onlyOwnerOrBeneficiary	Internal 🔒			
└	_vestingScheduleNotRevoked	Internal 🔒			
└	_vestingScheduleNotExpired	Internal 🔒			
└	_vestingScheduleRevocable	Internal 🔒			
PrescheduledTokenVesting	Implementation	TokenVesting			
└	<Constructor>	Public !	🔴	TokenVesting	

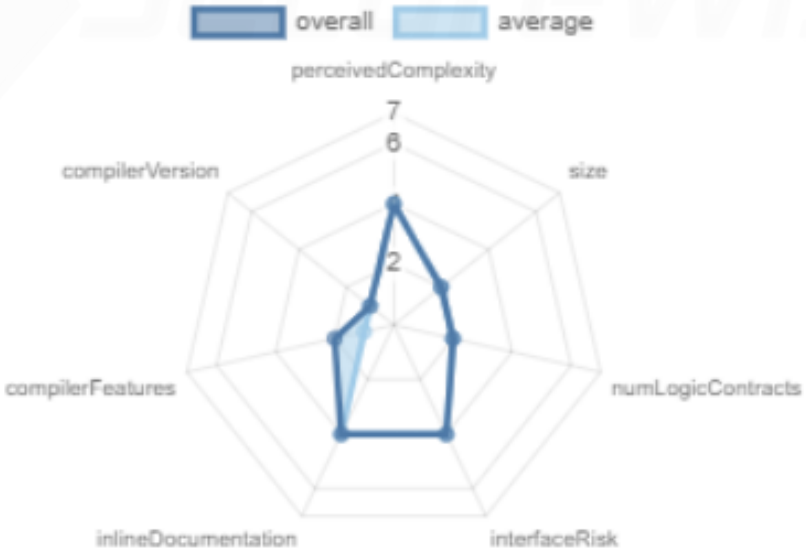
Inheritance Graph



Source Lines



Risk



Manual Review

Medium Risk

The revoke mechanics are not compatible with tokens that implement a block list feature

Description

Certain tokens like USDC and USDT have an address block list feature, which allows administrators to control specific addresses that are blocked from receiving transfers. If a transfer is attempted to a blocked address, the transaction will be reversed. When the revoke functionality is used, it forcefully transfers the vested tokens that can be claimed to an address with a predefined vesting schedule. However, if the chosen address has a claimable balance and is included in the token's block list, any attempts to revoke will result in the transaction being reversed.

Recommendation

Use the Pull over Push pattern to send tokens out of the contract in a revoke scenario.

Manual Review

Medium Risk

Insufficient input validation in function createVestingSchedule and extend

Description

*The **createVestingSchedule** function lacks sufficient validation for its input arguments, leading to several problematic scenarios:*

1. *The **config.start** argument can accept a timestamp that has already passed or is too far in the future. This lack of validation may result in incorrect or unexpected behavior.*
2. *The **config.cliff** argument can have a value that is too large, rendering users unable to claim their vested amount. It is important to validate this argument to ensure that the cliff period is reasonable and allows users to claim their vested funds.*
3. *The check for **config.duration != 0** is not sufficient to validate the duration argument. While the current implementation excludes a duration of zero, it does not account for other valid values such as **1.** Further validation should be added to ensure that the duration meets the necessary requirements.*
4. *If the **slicePeriodSeconds** argument is set to a large value, the mathematical calculations in the **_computeReleasableAmount** function may encounter rounding errors. It is crucial to consider and handle such scenarios appropriately to ensure accurate and predictable release calculations.*

*The **extend** function lacks sufficient validation for its input arguments, leading some problematic scenarios:*

1. ***extensionDuration** argument specifies the additional duration in seconds to be added to the vesting schedule. However, there is a potential issue with the function implementation as it does not enforce any limits on the value of **extensionDuration**, which could lead to unintended consequences.*

Recommendation

*Add sensible lower and upper bounds for all arguments of the **createVestingSchedule and extend** methods.*

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

AUDIT REPORT

SecureWise

securewise.org

t.me/securewisehub

twitter.com/securewiseAudit