Cyberscope

Audit Report

Nabana Vesting

January 2023

Type	BEP20
Network	BSC
Address	0x33242f967441a666aD65D6cBb910f9a93b1d911e
Audited by	© cyberscope

Table of Contents

Review

Contract Name	NabanaVesting
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x33242f967441a666ad65d6cbb910f9a93b1d911e
Address	0x33242f967441a666ad65d6cbb910f9a93b1d911e
Network	BSC

Audit Updates

Initial Audit	03 Jan 2023

Source Files

Filename	SHA256
NabanaVesting.sol	8d334ee7131f793bd7e57f74a628951d368f6ceba6724d7ec4f38cee925c5ea4

Introduction

This audit investigates security issues, business logic concerns, and potential improvements for the **Nabana Vesting** contract.

The **Nabana Token** and **Nabana DividendTracker** contracts audit can be found at https://github.com/cyberscope-io/audits/tree/main/nabana/audit.pdf.

Nabana Vesting

The **Nabana Vesting** contract implements a vesting locker mechanism. The tokens of some predefined addresses are vested. Each address has a different lock configuration.

Roles

The Nabana Vesting contract has two roles, the VESTER and the OWNER role.

VESTER

The VESTER role has the authority to

- Claim the rewards of his investment.

OWNER

The OWNER role has the authority to

- Lock/Unlock vesting at any given time.
- Add/Remove accounts as VESTER role.
- Withdraw Nabana Token’s balance to the owner’s account.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	OCTD	Transfers Contract's Tokens	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	NabanaVesting.sol#L429
Status	Unresolved

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `emergencyWithdraw` function.

```
function emergencyWithdraw() external onlyOwner {
  nabana.transfer(owner(), nabana.balanceOf(address(this)));
}
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	NabanaVesting.sol#L68
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead, and increases unnecessarily the gas consumption.

```
library SafeMath {}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	NabanaVesting.sol#L311,312,313,314,315,316,357,362,367,372
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address private constant managementAllocationWallet =
0x49E695A35bd65C735BB2b0033e77205b4C8cf02c
address private constant advisorsWallet = 0x20027f3D87D107d7a5A304A940f3A63F98E67298
address private constant influencersWallet =
0x3d574DE90E80b9Ce7CC48aC9520bDC9d19e549E4
address private constant strategicPartnersWallet =
0xA0f77506698e9e2F5919fbbC76c362a0e41555f3
address private constant rewardsIncentivesWallet =
0xa9E8627F2c78992de6aF3c76120625930E1A826F
address private constant reserveWallet = 0xFA38bFA8D5f3B4FB2E3B410F86a27776Ab3AAb4d
address _nabana
bool _vestState
address _vester
bool _state
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	NabanaVesting.sol#L408
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 unlockValidator
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	NabanaVesting.sol#L393,430
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
nabana.transfer(vesting.vester, unlockAmount)
nabana.transfer(owner(), nabana.balanceOf(address(this)))
```

Recommendation

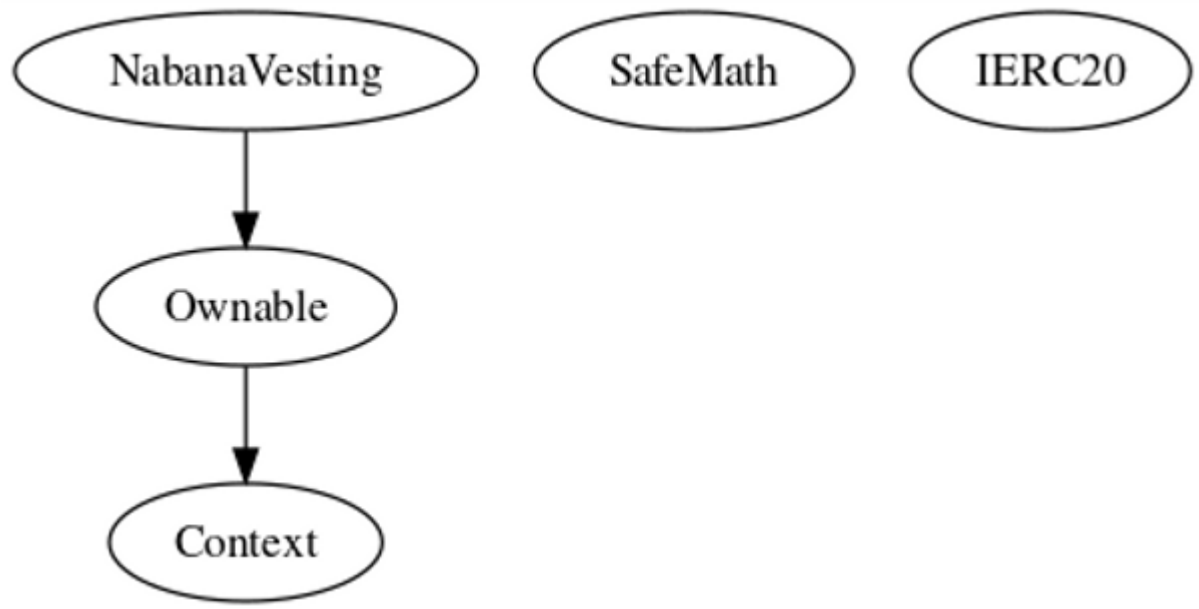
The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

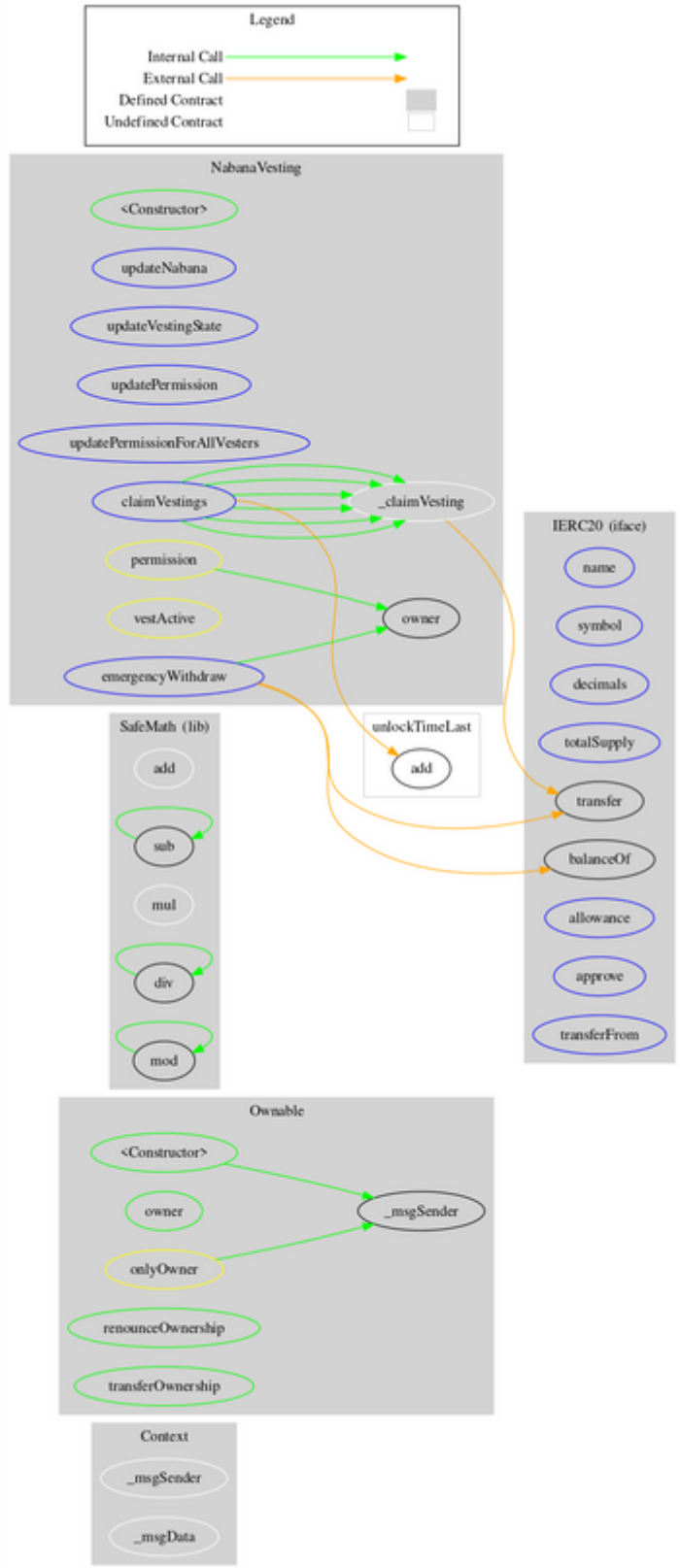
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
IERC20	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-

	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
NabanaVesting	Implementation	Ownable		
		Public	✓	-
	updateNabana	External	✓	onlyOwner
	updateVestingState	External	✓	onlyOwner
	updatePermission	External	✓	onlyOwner
	updatePermissionForAllVesters	External	✓	onlyOwner
	_claimVesting	Internal	✓	
	claimVestings	External	✓	vestActive permission
	emergencyWithdraw	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
NabanaVesting
<Constructor>
updateNabana
updateVestingState
updatePermission
updatePermissionForAllVesters
claimVestings
_claimVesting
permission
vestActive
owner
emergencyWithdraw
SafeMath (lib)
add
sub
mul
div
mod
unlockTimeLast
add
IERC20 (iface)
name
symbol
decimals
totalSupply
transfer
balanceOf
allowance
approve
transferFrom
Ownable
<Constructor>
owner
_msgSender
onlyOwner
renounceOwnership
transferOwnership
Context
_msgSender
_msgData

Summary

NabanaVesting contract implements a locker vesting mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io