



Cyberscope

## Audit Report

# Nevis Investments

July 2022

Type           BEP20

Network       BSC

Address       0xCbe59C5Cb0F6C35918c320774bc11c46252F70a3

Audited by   © cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Contract Review</b>	<b>3</b>
<b>Source Files</b>	<b>3</b>
<b>Audit Updates</b>	<b>3</b>
<b>Introduction</b>	<b>4</b>
<b>Source Files</b>	<b>5</b>
<b>Audit Updates</b>	<b>5</b>
<b>Contract Analysis</b>	<b>6</b>
<b>Contract Diagnostics</b>	<b>7</b>
<b>STC - Succeeded Transfer Check</b>	<b>8</b>
Description	8
Recommendation	8
<b>DSM - Data Structure Misuse</b>	<b>9</b>
Description	9
Recommendation	9
<b>L01 - Public Function could be Declared External</b>	<b>10</b>
Description	10
Recommendation	10
<b>L02 - State Variables could be Declared Constant</b>	<b>11</b>
Description	11
Recommendation	11
<b>L04 - Conformance to Solidity Naming Conventions</b>	<b>12</b>
Description	12
Recommendation	12
<b>L05 - Unused State Variable</b>	<b>13</b>
Description	13

<b>Recommendation</b>	<b>13</b>
<b>L09 - Dead Code Elimination</b>	<b>14</b>
<b>Description</b>	<b>14</b>
<b>Recommendation</b>	<b>14</b>
<b>L13 - Divide before Multiply Operation</b>	<b>15</b>
<b>Description</b>	<b>15</b>
<b>Recommendation</b>	<b>15</b>
<b>L14 - Uninitialized Variables in Local Scope</b>	<b>16</b>
<b>Description</b>	<b>16</b>
<b>Recommendation</b>	<b>16</b>
<b>Contract Functions</b>	<b>17</b>
<b>Contract Flow</b>	<b>22</b>
<b>Domain Info</b>	<b>23</b>
<b>Summary</b>	<b>24</b>
<b>Disclaimer</b>	<b>25</b>
<b>About Cyberscope</b>	<b>26</b>

## Contract Review

<b>Contract Name</b>	NEVISBANK
<b>Compiler Version</b>	v0.8.7+commit.e28d00a7
<b>Optimization</b>	200 runs
<b>Licence</b>	MIT
<b>Explorer</b>	<a href="https://bscscan.com/token/0xCbe59C5Cb0F6C35918c320774bc11c46252F70a3">https://bscscan.com/token/0xCbe59C5Cb0F6C35918c320774bc11c46252F70a3</a>
<b>Domain</b>	<a href="https://nevis.investments">https://nevis.investments</a>

## Source Files

<b>Filename</b>	<b>SHA256</b>
<b>contract.sol</b>	99b00f4c0e023ca16dc241979bdb7154a4fbcc19ffe1c7011b28f7af53cf72b4

## Audit Updates

<b>Initial Audit</b>	13th July 2022
<b>Corrected</b>	

# Introduction

The audited address `0xcbe59c5cb0f6c35918c320774bc11c46252f70a3` implements the contract of the Nevis Investments bank. The bank contract contains an investment platform where users can become referrers to other users to earn bonus, invest and withdraw dividends. The auditing of the bank contract is out of the scope of this audit. This audit is dedicated to the nevis bank implementation.

During the auditing period the contract:

<https://bscscan.com/address/0xcbe59c5cb0f6c35918c320774bc11c46252f70a3#contracts>

is pointed by the proxy address:

<https://bscscan.com/address/0xb37bf7f66c8187b9a23f386fed8c8bcf40f58ad#code>

## Source Files

Filename	SHA256
contract.sol	3a36341d9bb0f50fc53472f1b1cfed3ff8a98e74766f3c6 12e738ae8529e279a

## Audit Updates

Initial Audit	12th July 2022
Corrected	

## Contract Analysis

The users have the ability to invest and withdraw tokens from the Nevis Trade Bank. In addition the users have the ability to become referrers to other users.

Nevis Trade Bank has one plan which locks your tokens for 365 days minimum and returns 60.27% interest.

- On every investment on the contract the owner keeps a 10% fee on the contract. If a referrer is set, 1% bonus goes to the referrer's bonus. In addition, a minimum amount of investment is required to be greater or equal to  $1 * 10^{18}$
- The users can withdraw tokens when the plan time is passed.
- The users can reinvest all their earnings bonus and dividends if they have any.
- A referrer can withdraw his bonus if he has dividends in the Nevis Trade Bank.

# Contract Diagnostics

● Critical    ● Medium    ● Minor

Severity	Code	Description
●	STC	Succeeded Transfer Check
●	DSM	Data Structure Misuse
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L05	Unused State Variable
●	L09	Dead Code Elimination
●	L13	Divide before Multiply Operation
●	L14	Uninitialized Variables in Local Scope



## STC - Succeeded Transfer Check

**Criticality**

minor

**Location**

contract.sol#L705,L840

### 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.

```
token.transfer(msg.sender,famount);

function rewards(address userAddress, uint256 amount) public onlyOwner() {
    require(getContractBUSDBalance() > amount,"Low Balance");
    token.transfer(userAddress,amount);
}
```

### Recommendation

The contract should check if the result of the transfer methods is successful.

## DSM - Data Structure Misuse

**Criticality**

minor

**Location**

contract.sol#L511

### Description

The contract uses the valuable `_confirmedSnipers` as an array. The business logic of the contract does not require to iterate this structure sequentially. Thus, unnecessary loops are produced that increase the required gas.

Since the contract plan is always one, it could be a variable rather than an array.

```
Plan[] internal plans;
```

### Recommendation

The contract could use a data structure that provides instant access. For instance, a Set or a Map would fit better to the business logic of the contract. This way the time complexity will be reduced from  $O(n)$  to  $O(1)$ .

## L01 - Public Function could be Declared External

<b>Criticality</b>	minor
<b>Location</b>	contract.sol#L116,208,840,683,807,91,846,552,127,87,98,102,189,122,811,865,217,223,833,736,83,823,95,111,859,850,795,803,107,814

### Description

Public functions that are never called by the contract should be declared external to save gas.

```
getUserAvailable  
allowance  
getContractTokenBalance  
getUserAmountOfRoll  
getinfo  
setPairaddress  
approve  
totalSupply  
getUserDepositInfo  
...
```

### Recommendation

Use the external attribute for functions never called from the contract.

## L02 - State Variables could be Declared Constant

**Criticality**

minor

**Location**

contract.sol#L80,79,77,81,499

### Description

Constant state variables should be declared constant to save gas.

```
totalRefBonus  
_decimals  
_limitSupply  
_name  
_symbol
```

### Recommendation

Add the constant attribute to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	minor
<b>Location</b>	contract.sol#L552,713,564,545,77,80,313,329,488,312,71,79,547,549,548,49,865,859,350,620,81

### Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow \_ at the beginning of the mixed\_case match for private variables and unused parameters.

```
_decimals  
buy_token  
Busd  
WETH  
pair_address  
Nevis_token  
router_address  
_____gap  
BUSD  
...
```

### Recommendation

Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions>.

## L05 - Unused State Variable

**Criticality**

minor

**Location**

contract.sol#L530,49,77

### Description

There are segments that contain unused state variables.

```
_limitSupply  
____gap  
referralAddress
```

### Recommendation

Remove unused state variables.

## L09 - Dead Code Elimination

**Criticality**

minor

**Location**

contract.sol#L143

### Description

Functions that are not used in the contract, and make the code's size bigger.

```
_burn
```

### Recommendation

Remove unused functions.

## L13 - Divide before Multiply Operation

**Criticality**

minor

**Location**

contract.sol#L762

### Description

Performing divisions before multiplications may cause lose of prediction.

```
share = user.deposits[i].amount.mul(plans[user.deposits[i].plan].percent).div(PANPER_DIVIDER)
```

### Recommendation

The multiplications should be prior to the divisions.



## L14 - Uninitialized Variables in Local Scope

**Criticality**

minor

**Location**

contract.sol#L765

### Description

There are variables that are defined in the local scope and are not initialized.

```
totalAmount
```

### Recommendation

All the local scoped variables should be initialized.

# Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>Initializable</b>	Implementation			
	isConstructor	Private		
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>ERC20</b>	Implementation	IERC20		
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
<b>Context</b>	Implementation			

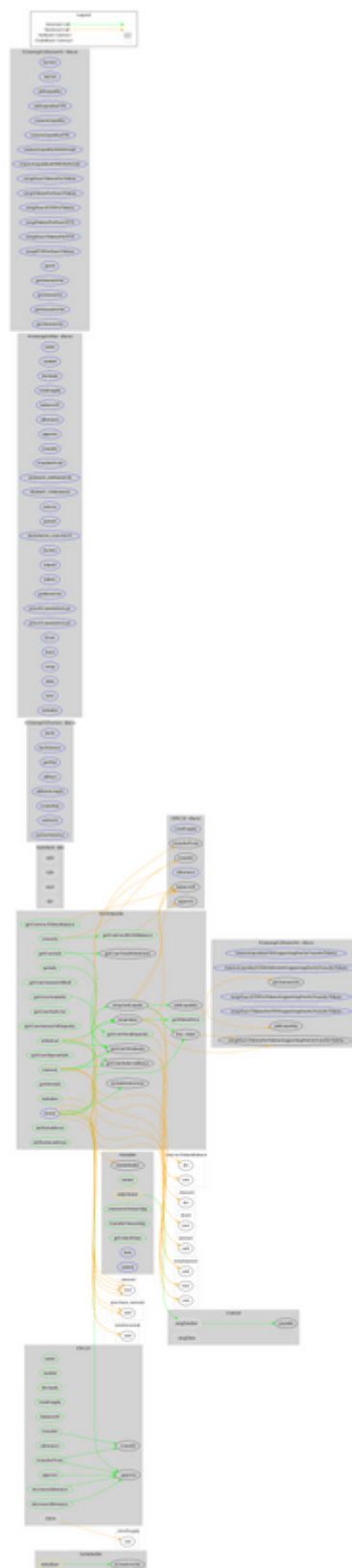
	_msgSender	Internal		
	_msgData	Internal		
<b>Ownable</b>	Implementation	Context, Initializable		
	owninitialize	Public	✓	initializer
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	geUnlockTime	Public		-
	lock	External	✓	onlyOwner
	unlock	External	✓	-
<b>SafeMath</b>	Library			
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
<b>IUniswapV2Factory</b>	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
<b>IUniswapV2Pair</b>	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-

	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
<b>IUniswapV2Router01</b>	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-

	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
<b>IUniswapV2Router02</b>	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
<b>NEVISBANK</b>	Implementation	ERC20, Ownable		
	initialize	Public	✓	initializer
	invest	External	Payable	-
	buy_token	Private	✓	
	swapAndLiquify	Private	✓	
	addLiquidity	Private	✓	
	withdraw	Public	✓	-
	getTokenPrice	Public		-
	swaptoken	Private	✓	
	reinvest	Public	✓	-
	getUserDividends	Public		-
	includeInvestor	Private		
	getUserReferralBonus	Public		-

	getUserAmountOfRoll	Public		-
	getContractBUSDBalance	Public		-
	getContractTokenBalance	Public		-
	getUserReferrer	Public		-
	getUserAmountOfDeposits	Public		-
	getUserAvailable	Public		-
	getUserTotalDeposits	Public		-
	getUserDepositInfo	Public		-
	rewards	Public	✓	onlyOwner
	getSiteInfo	Public		-
	getUserTotalWithdrawn	Public		-
	getUserInfo	Public		-
	getinfo	Public		-
	setPairaddress	Public	✓	onlyOwner
	setRouteraddress	Public	✓	onlyOwner

# Contract Flow



## Domain Info

<b>Domain Name</b>	nevis.investments
<b>Registry Domain ID</b>	96638f912b3b43dea5c1a1d6d10aee9e-DONUTS
<b>Creation Date</b>	2022-02-02T10:11:43Z
<b>Updated Date</b>	2022-05-05T11:44:18Z
<b>Registry Expiry Date</b>	2027-02-02T10:11:43Z
<b>Registrar WHOIS Server</b>	www.bigrock.com.in/whois-lookup.php
<b>Registrar URL</b>	http://bigrock.com
<b>Registrar</b>	BigRock Solutions Ltd.
<b>Registrar IANA ID</b>	1495

The domain has been created in over 4 years before the creation of the audit.

There is no public billing information, the creator is protected by the privacy settings.



## Summary

Nevis Investment Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

# Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

## About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.



The Cyberscope team

<https://www.cyberscope.io>