

**Blockchain Security - Smart Contract Audits** 

# **Security Assessment**

March 30, 2022



Disclaimer	3	
Description	5	
Engagement	5	
Project Engagement	5	
Logo	6	
Contract Link	6	
Risk Level Classification	7	
Methodology	8	
Used Code from other Frameworks / Smart Contrac	ts (Imports)	
Description	10	0
Scope of Work	13	3
Inheritance Graph	14	4
Verify Claim	1!	5
Overall Checkup	19	9
Write Functions of Contract	20	0
SWC Attack	2:	1
Audit Result	2!	5
Audit Comments	20	<b>-</b>

#### **Disclaimer**

**ContractWolf.io** audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

ContractWolf does not provide any warranty on its released reports.

**ContractWolf** should not be used as a <u>decision</u> to invest into an audited project and is not affiliated nor partners to its audited contract projects.

ContractWolf provides transparent report to all its "clients" and to its "client's participants" and will not claim any guarantee of bug-free code within its SMART CONTRACT.

**ContractWolf** presence is to analyze, audit and assess the client's smart contract's code.

Each company or projects should be liable to its security flaws and functionalities.

#### **Network**

BSC / Binance Smart Chain (BEP20 protocol)

#### Website

https://entire-dino.monster/

### **Telegram**

https://t.me/lastentiredino

#### **Twitter**

https://twitter.com/DinoEntire

#### **Description**

BEP20 lottery token with high reflection and mini-game to win huge reward token.

#### **ContractWolf Engagement**

30<sup>th</sup> of March 2022, **Entire Dino** engaged and agrees to audit their smart contract's code by ContractWolf. The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

**ContractWolf** will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, whitepaper and repository which have been provided by **Entire Dino**.

### Logo



#### **Contract link**

https://bscscan.com/address/0xD3B093d59fF6e7Ffc25d494B56bDAF2E5 4dC89F3

#### **Risk Level Classification**

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.

Risk Level is computed based on CVSS Version 3.0

Level	Value	Vulnerability
Critical	9 - 10	An Exposure that can affect the contract functions in several events that can risk and disrupt the contract
High	7 - 8.9	An Exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner
Medium	4 - 6.9	An opening that could affect the outcome in executing the contract in a specific situation
Low	0.1 - 3.9	An opening but doesn't have an impact on the functionality of the contract
Informational	0	An opening that consists of information's but will not risk or affect the contract

#### **Auditing Approach**

Every line of code along with its functionalities will undergo manual review to check its security issues, quality, and contract scope of inheritance. The manual review will be done by our team that will document any issues that there were discovered.

#### Methodology

The auditing process follows a routine series of steps:

- 1. Code review that includes the following:
  - Review of the specifications, sources, and instructions provided to ContractWolf to make sure we understand the size, scope, and functionality of the smart contract.
  - Manual review of code, our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities and security flaws.
- 2. Testing and automated analysis that includes:
  - Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.
- 3. Best practices review, the team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security, and control within the smart contract.
- 4. Recommendations to help the project take steps to secure the smart contract.

# **Used Code from other Frameworks/Smart Contracts (Direct Imports)**

#### **Imported Packages**

- IERC20
- IERC20Metadata
- Context
- ERC20
- Ownable
- SafeMath
- Clones
- IUniswapV2Factory
- IUniswapV2Router01
- IUniswapV2Router02
- IERC20Upgradeable
- IERC20MetadataUpgradeable
- Initializable
- ContextUpgradeable
- ERC20Upgradeable
- OwnableUpgradeable
- IUniswapV2Pair
- SafeMathInt
- SafeMathUInt

- IterableMapping
- DividendPayingTokenInterface
- DividendPayingTokenOptionalInterface
- DividendPayingToken
- BABYTOKENDividendTracker
- BaseToken
- BABYTOKEN

## **Description**

Optimization enabled: Yes

Version: v0.8.0

Decimal: 18

Symbol: DINO

### **Capabilities**

#### **Components**

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	5	10	10

#### **Exposed Functions**

Version	Public	Private
1.0	55	10

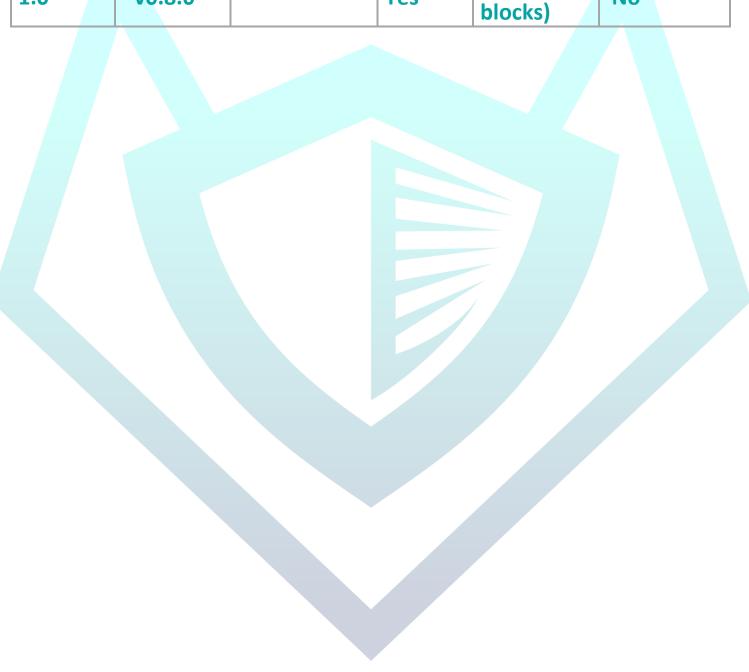
Version	External	Internal
1.0	88	74

#### **State Variables**

Version	Total	Public
1.0	46	20

#### Capabilities

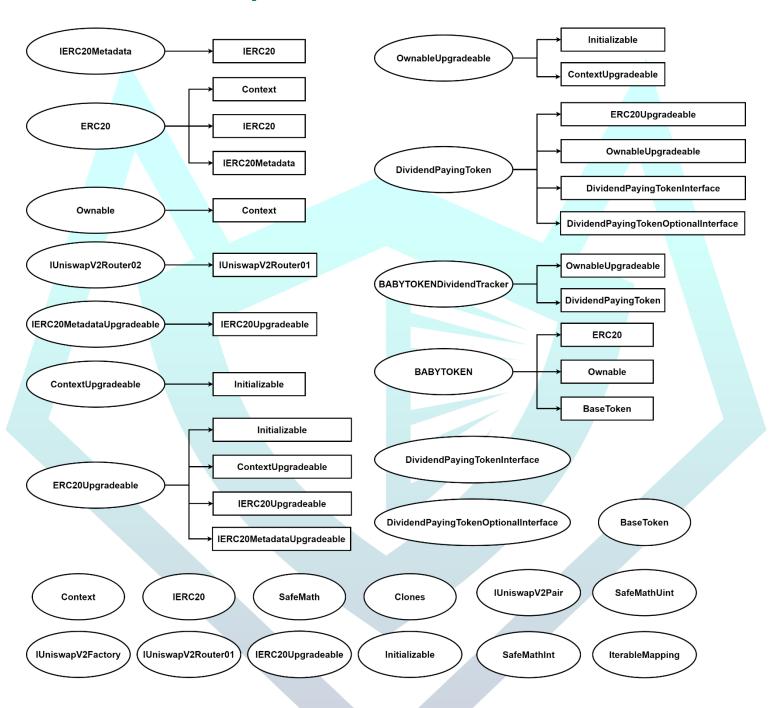
Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyabl e Contracts
1.0	^v0.8.0		Yes	Yes (3 asm blocks)	No



#### **Scope of Work**

**Client** team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract.

#### **Inheritance Graph**



# **Verify Claims**

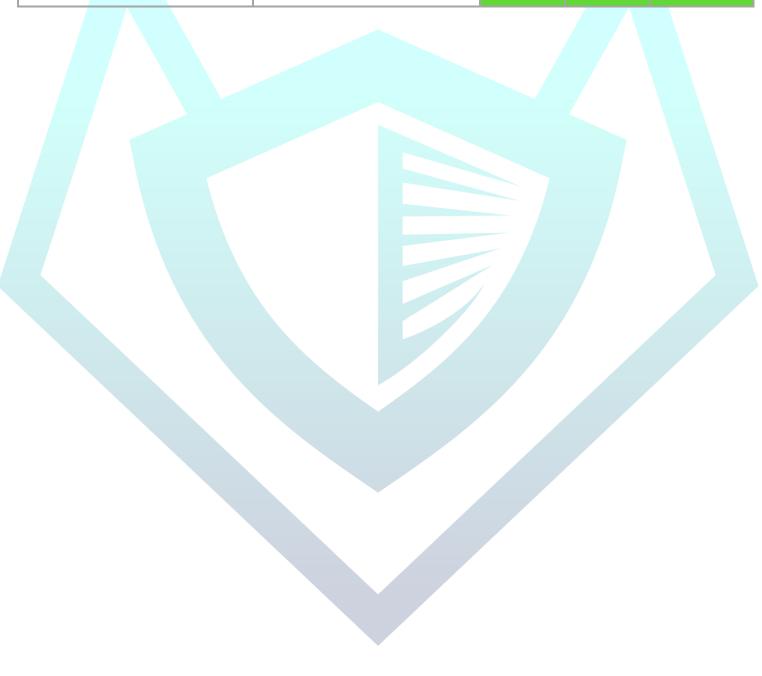
#### **Correct implementation of Token Standard**

Tested	Verified
<b>√</b>	X

Function	Description	Exist	Tested	Verified
TotalSupply	Information about the total coin or token supply	<b>√</b>	<b>√</b>	<b>√</b>
BalanceOf	Details on the account balance from a specified address	<b>√</b>	<b>√</b>	✓
Transfer	An action that transfers a specified amount of coin or token to a specified address	<b>√</b>	<b>√</b>	✓
TransferFrom	An action that transfers a specified amount of coin or token from a specified address	<b>√</b>	<b>√</b>	<b>√</b>
Approve	Provides permission to withdraw specified number of coin or token from a specified address	<b>√</b>	<b>√</b>	<b>√</b>

#### **Optional implementation**

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	<b>√</b>	<b>√</b>	<b>√</b>



#### **Deployer cannot mint after initial deployment**

Statement	Exist	Tested	Verified	File
Deployer cannot mint	<b>√</b>	<b>√</b>	<b>✓</b>	Main

Max / Total supply: 10,000,000 DINO

#### **Deployer cannot block user**

Statement	Exist	Tested	Verified
Deployer cannot block user	_	_	_

#### **Deployer cannot burn**

Statement	Exist	Tested	Verified
Deployer cannot burn	<b>√</b>	<b>√</b>	<b>√</b>

#### **Deployer cannot pause contract**

Statement	Exist	Tested	Verified
Deployer cannot pause	_	_	_

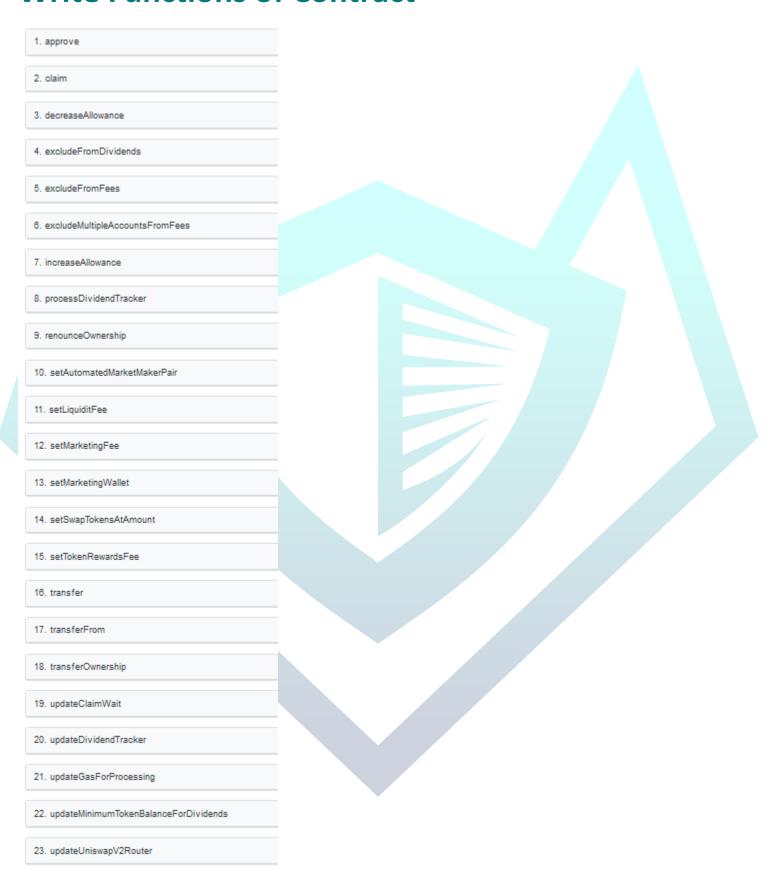
## **Overall Checkup (Smart Contract Security)**



#### Legend

Attribute	Symbol
Verified / Checked	<b>√</b>
Partly Verified	X
Unverified / Not checked	P
Not Available	_

#### **Write Functions of Contract**



#### **SWC Attacks**

ID	Title	Relationships	Status
<u>SWC-136</u>	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
<u>SWC-135</u>	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
<u>SWC-133</u>	Hash Collisions with Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
<u>SWC-132</u>	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
<u>SWC-131</u>	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
<u>SWC-130</u>	Right-To Left Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
<u>SWC-129</u>	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-126	Insufficient Gas Griefing	CWE-691: Insufficient Control Flow Management	PASSED
<u>SWC-125</u>	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
<u>SWC-124</u>	Write to Arbitrary Storage Location	CWE-123: Write-what- where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
<u>SWC-119</u>	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED

SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
<u>SWC-115</u>	Authorization through tx.origin	CWE-477: Use of Obsolete Function	NOT PASSED
<u>SWC-114</u>	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegate call to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
<u>SWC-110</u>	Assert Violation	CWE-670: Always- Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED

SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED
<u>SWC-105</u>	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
<u>SWC-104</u>	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

# **AUDIT PASSED**

#### **Low Issues**

Use of "tx.origin" as part of authorization control. (SWC-115)

L: 2952 C:12, L: 3048 C:20

#### **Audit Comments**

- Deployer can renounce ownership
- Deployer can transfer ownership
- Deployer cannot mint after initial deployment
- Deployer cannot burn
- Deployer cannot block user
- Deployer cannot pause contract
- Deployer can set fees up to 25% only



# CONTRACTWOLF

**Blockchain Security - Smart Contract Audits**