



**CONTRACT  
WOLF**

**Blockchain Security - Smart Contract Audits**

# **Security Assessment**

November 22, 2022



<b>Disclaimer</b>	<b>3</b>
<b>Scope of Work &amp; Engagement</b>	<b>3</b>
<b>Project Description</b>	<b>4</b>
<b>Risk Level Classification</b>	<b>5</b>
<b>Methodology</b>	<b>6</b>
<b>Used Code from other Frameworks / Smart Contracts (Imports)</b>	<b>7</b>
<b>Token Description</b>	<b>8</b>
<b>Inheritance Graph</b>	<b>9</b>
<b>Overall Checkup</b>	<b>10</b>
<b>Verify Claim</b>	<b>11</b>
<b>Write Functions of Contract</b>	<b>12</b>
<b>Call Graph</b>	<b>13</b>
<b>SWC Attacks</b>	<b>14</b>
<b>Audit Result</b>	<b>16</b>
<b>Findings</b>	<b>17</b>
<b>Audit Comments</b>	<b>18</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 decision 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 "clients 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.

## Scope of Work

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

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 has been provided by **Project Sample**.

# Description

**ContractWolf** – Blockchain Security, Smart Contract Audits, and KYC.



# 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

- Context
- SafeMath
- Ownable
- ThisContract
- IBEP20

## Description

Optimization enabled: Yes

Decimal: 18

Symbol: SAMPLE

Max / Total Supply: 500,000,000

## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	1	1	0

### Exposed Functions

Version	Public	Private	External	Internal
1.0	5	8	20	15

### State Variables

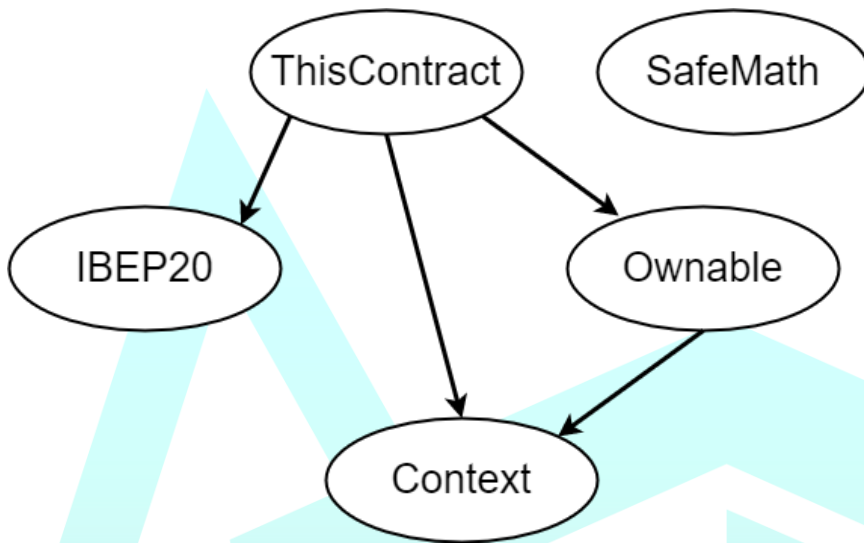
Version	Total	Public
1.0	15	7

### Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.8.0		Yes	No	No



# Inheritance Graph



## Correct implementation of Token Standard

Tested	Verified
✓	✓

## Overall Checkup (Smart Contract Security)

Tested	Verified
✓	✓

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

# Verify Claims

Statement	Exist	Tested	Deployer
Renounce Ownership	—	—	—
Mint	✓	✓	X
Burn	✓	✓	X
Block	—	—	—
Pause	—	—	—

## Legend

Attribute	Symbol
Verified / Can	✓
Verified / Cannot	X
Unverified / Not checked	🚩
Not Available	—

# Write Functions of Contract

1. approve

2. decreaseAllowance

3. increaseAllowance

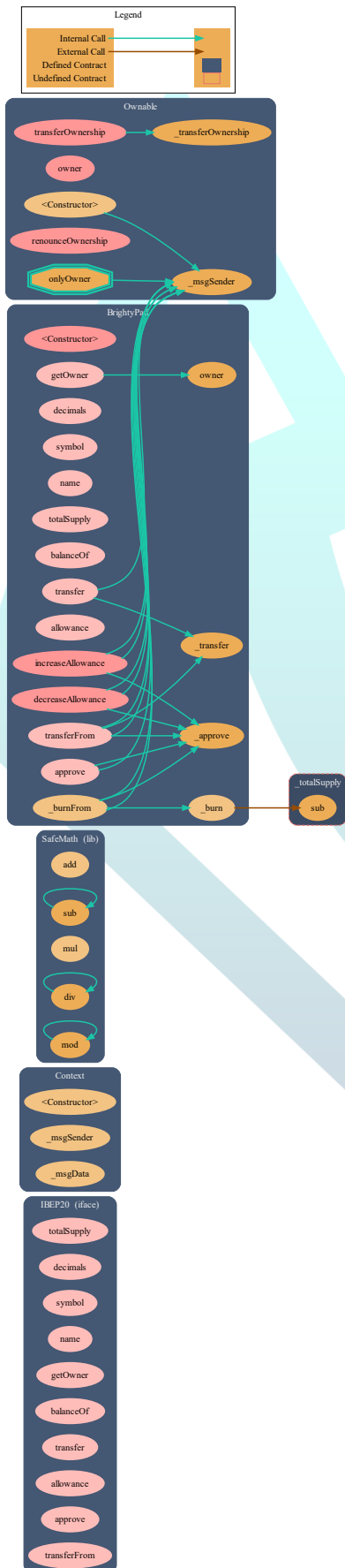
4. renounceOwnership

5. transfer

6. transferFrom

7. transferOwnership

# Call Graph



# SWC Attacks

ID	Title	Status
<a href="#">SWC-136</a>	Unencrypted Private Data On-Chain	PASSED
<a href="#">SWC-135</a>	Code With No Effects	PASSED
<a href="#">SWC-134</a>	Message call with hardcoded gas amount	PASSED
<a href="#">SWC-133</a>	Hash Collisions with Multiple Variable Length Arguments	PASSED
<a href="#">SWC-132</a>	Unexpected Ether balance	PASSED
<a href="#">SWC-131</a>	Presence of unused variables	PASSED
<a href="#">SWC-130</a>	Right-To Left Override control character (U+202E)	PASSED
<a href="#">SWC-129</a>	Typographical Error	PASSED
<a href="#">SWC-128</a>	DoS With Block Gas Limit	PASSED
<a href="#">SWC-127</a>	Arbitrary Jump with Function Type Variable	PASSED
<a href="#">SWC-126</a>	Insufficient Gas Griefing	PASSED
<a href="#">SWC-125</a>	Incorrect Inheritance Order	PASSED
<a href="#">SWC-124</a>	Write to Arbitrary Storage Location	PASSED
<a href="#">SWC-123</a>	Requirement Violation	LOW ISSUE
<a href="#">SWC-122</a>	Lack of Proper Signature Verification	PASSED
<a href="#">SWC-121</a>	Missing Protection against Signature Replay Attacks	PASSED
<a href="#">SWC-120</a>	Weak Sources of Randomness from Chain Attributes	LOW ISSUE
<a href="#">SWC-119</a>	Shadowing State Variables	PASSED
<a href="#">SWC-118</a>	Incorrect Constructor Name	PASSED
<a href="#">SWC-117</a>	Signature Malleability	PASSED
<a href="#">SWC-116</a>	Block values as a proxy for time	PASSED
<a href="#">SWC-115</a>	Authorization through tx.origin	LOW ISSUE
<a href="#">SWC-114</a>	Transaction Order Dependence	PASSED
<a href="#">SWC-113</a>	DoS with Failed Call	LOW ISSUE
<a href="#">SWC-112</a>	Delegate call to Untrusted Callee	PASSED
<a href="#">SWC-111</a>	Use of Deprecated Solidity Functions	PASSED

<a href="#"><u>SWC-110</u></a>	Assert Violation	<b>PASSED</b>
<a href="#"><u>SWC-109</u></a>	Uninitialized Storage Pointer	<b>PASSED</b>
<a href="#"><u>SWC-108</u></a>	State Variable Default Visibility	<b>LOW ISSUE</b>
<a href="#"><u>SWC-107</u></a>	Reentrancy	<b>NOT PASSED</b>
<a href="#"><u>SWC-106</u></a>	Unprotected SELFDESTRUCT Instruction	<b>PASSED</b>
<a href="#"><u>SWC-105</u></a>	Unprotected Ether Withdrawal	<b>PASSED</b>
<a href="#"><u>SWC-104</u></a>	Unchecked Call Return Value	<b>PASSED</b>
<a href="#"><u>SWC-103</u></a>	Floating Pragma	<b>LOW ISSUE</b>
<a href="#"><u>SWC-102</u></a>	Outdated Compiler Version	<b>PASSED</b>
<a href="#"><u>SWC-101</u></a>	Integer Overflow and Underflow	<b>PASSED</b>
<a href="#"><u>SWC-100</u></a>	Function Default Visibility	<b>PASSED</b>

# AUDIT PASSED

## High Issues

Cancel Function	L: 1247
-----------------	---------

## Medium Issues

Multiple calls executed in the same transaction	L: 861
---	--------

## Low Issues

A floating pragma is set (SWC-103)	L: 11
Potential use of “block.number” as source of randomness (SWC-120)	L: 327, 521, 564



# Findings

## High Issues

- Cancel any non-executed proposal via cancel function

## Medium Issues

### Description:

Multiple calls are executed in the same transaction (SWC – 113)

```
408     function withdrawTokensBNB(uint256 id)public {
409         require (id <= investor[msg.sender].deposits.length,"Invalid Id");
410         require (!investor[msg.sender].deposits[id].isToken,"Not A BNB Deposit");
411         uint256 totalDeposit=investor[msg.sender].deposits[id].amount;
412         require(totalDeposit>0,"No Deposit Found");
413         require(totalDeposit<=getContractBNBBalance(),"Not Enough Token for withdrawl from contract please try after some time");
414         uint256 tax=totalDeposit.mul(withdrawTax).div(divider);
415         payable(treasury).transfer(tax);
416         payable(msg.sender).transfer(totalDeposit.sub(tax));
417         remove(id);
418         investor[msg.sender].checkBNB=block.timestamp;
419         investor[msg.sender].withdrawCheckBNB=block.timestamp;
420         emit Withdrawn(msg.sender, totalDeposit);
421     }
```

### Suggestion:

It is recommended to follow call best practices:

- Avoid combining multiple calls in a single transaction, especially when calls are executed as part of a loop
- Always assume that external calls can fail
- Implement the contract logic to handle failed calls

## Low Issues

- Token migration may result in orphaned balances
- Approve function may lead to token theft
- Multiple calls are executed on a single transaction, this call is executed following another call within the same transaction
- Unnecessary nonReentrant modifier waste gas
- Variable shadowing between Unitroller and Comptroller, try to avoid state variables in the proxy, particularly public ones

# Audit Comments

- Owner can initialize market
- Owner can transfer admin role
- Owner can set comptroller
- Owner can accrue interests and set reserve factor fresh
- Owner can accrue interests and reduce reserves to admin address
- Owner can accrue interest and update interest rate model



# CONTRACTWOLF

Blockchain Security - Smart Contract Audits