



# CONTRACT WOLF

**Blockchain Security - Smart Contract Audits**

## Security Assessment

September 3, 2022





# 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 it's **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

The **Alvey Nomics'** 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 the **Alvey Nomics**.

## Description

Alveynomics is a token on binance smart chain. Inspired by Alvey Chain, Alveynomics is developing supportive protocols and services for Alvey Chain.



# 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

- SafeMath
- IBEP2
- Auth
- IDEXFactory
- IDEXRouter
- IDividendDistributor
- DividendDistributor
- Alveynomics

## Description

Optimization enabled: No

Decimal: 9

Symbol: ALVEY

Max / Total supply: 100,000,000,000

## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	1	4	1

### Exposed Functions

Version	Public	Private	External	Internal
1.0	15	0	55	18

### State Variables

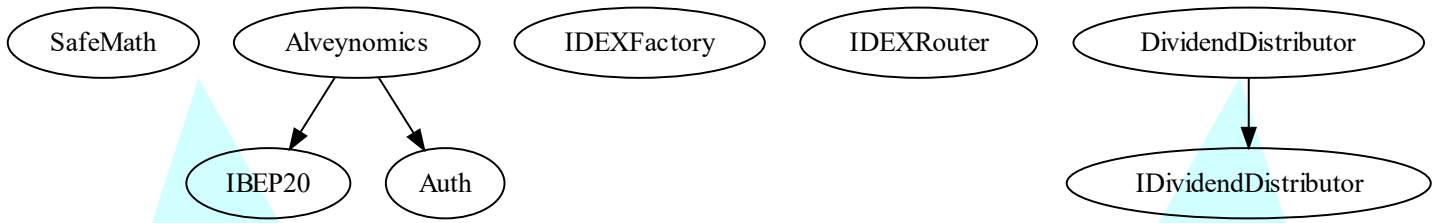
Version	Total	Public
1.0	61	31

### Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.7.6		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	✓	✓	✓

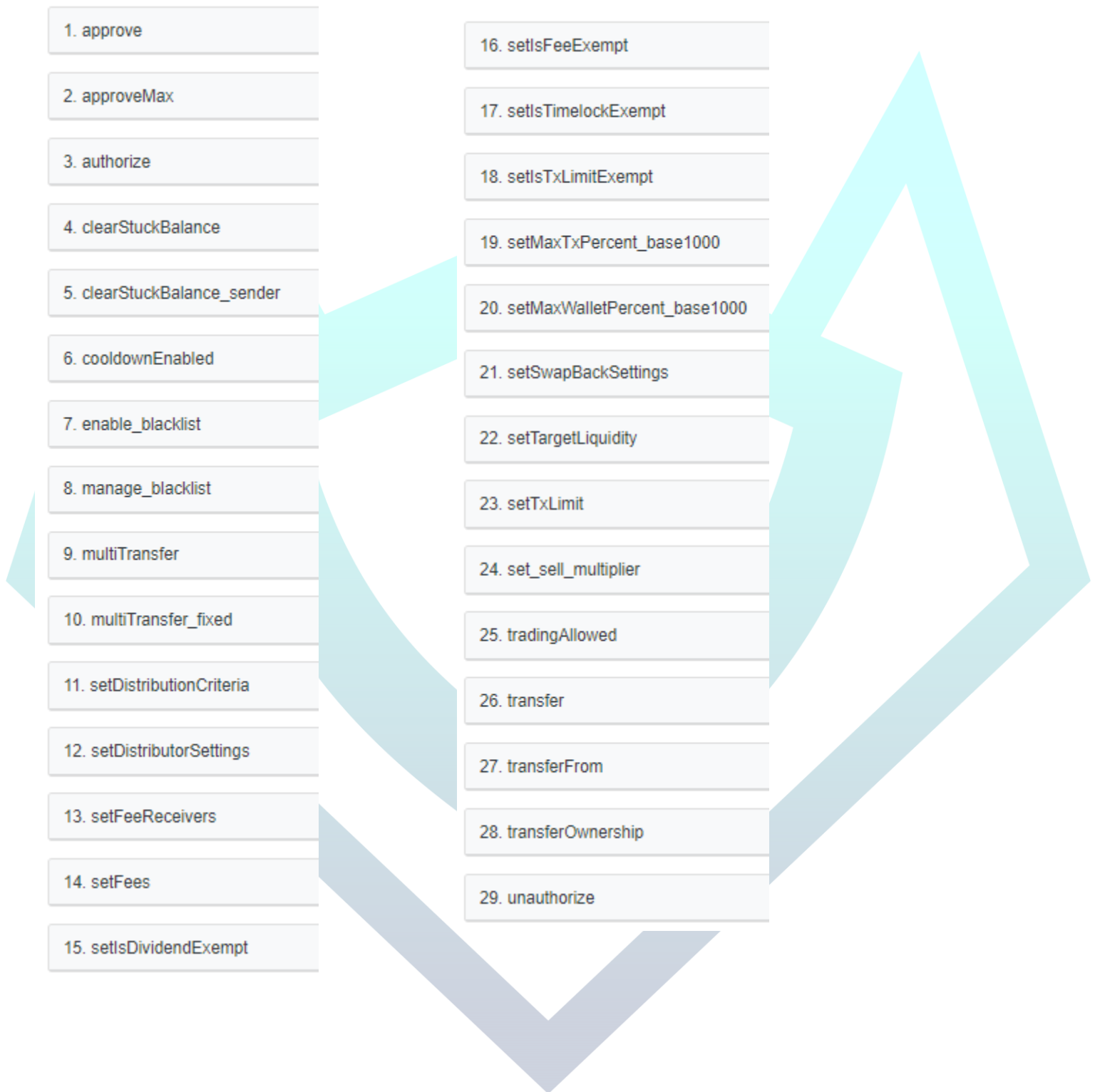
# Verify Claims

Statement	Exist	Tested	Owner
Renounce Ownership	✓	✓	✓
Mint	—	—	—
Burn	✓	✓	✗
Block	✓	✓	✓
Pause	—	—	—

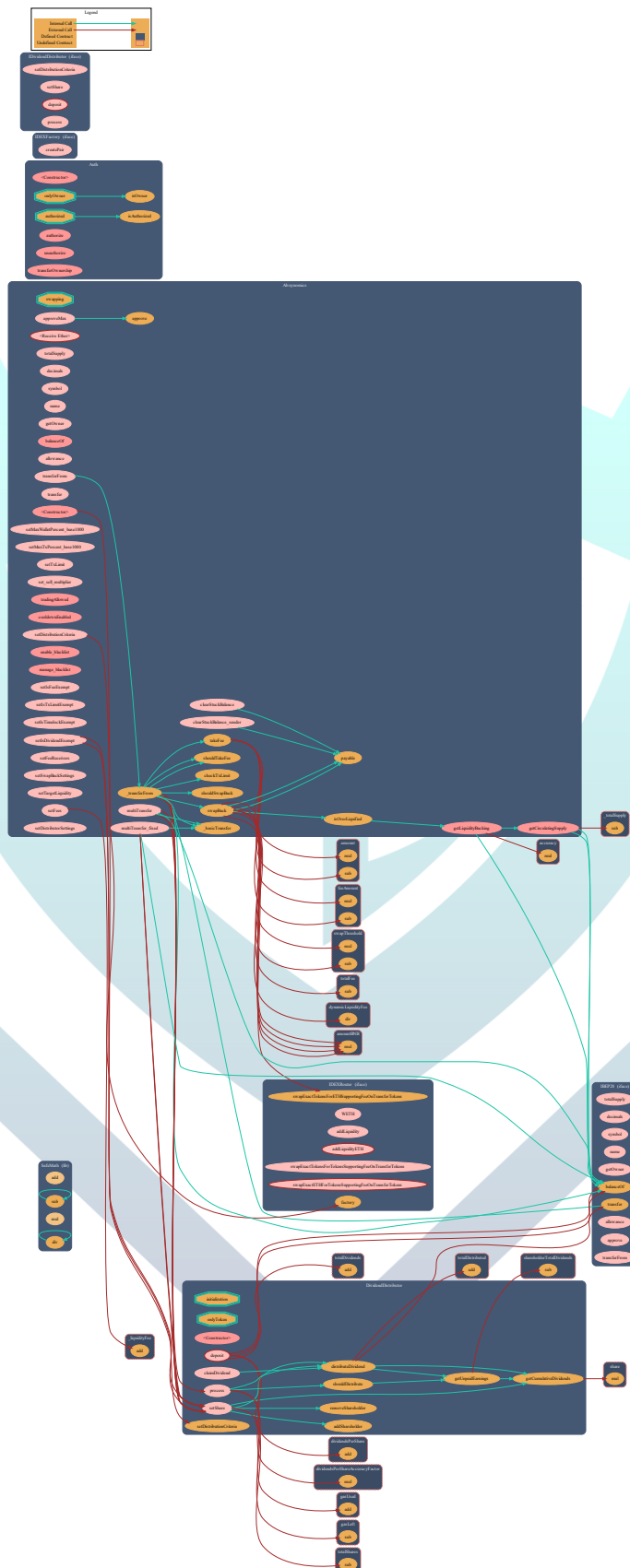
## Legend

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

# Write Functions of Contract



## 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	PASSED
<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	PASSED
<a href="#">SWC-114</a>	Transaction Order Dependence	PASSED
<a href="#">SWC-113</a>	DoS with Failed Call	PASSED
<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>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

## Low Issues

A floating pragma is set (SWC-103)	L: 11
State variable visibility is not set (SWC-108)	L: 172, 180, 181, 182, 184, 185, 186, 199, 201, 338, 339, 340, 346, 351, 352, 358, 359, 360, 361, 378, 379, 385, 387, 388, 391, 399
Potential use of “block.number” as source of randomness	L: 483, 582



# Audit Comments

## Description:

A floating pragma is set (SWC-103)

```
pragma solidity ^0.7.6;
```

## Suggestion:

Specific version to ensure that the bytecode does not vary between builds

---

## Description:

State variable visibility is not set (SWC-108)

```
bool inSwap;
```

## Suggestion:

Specify variables as **public**, **internal**, or **private**.

---

## Description:

Potential use of “block.number” as source of randomness (SWC-120)

## Suggestion:

- Using commitment scheme, e.g. RANDAO.
- Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles.
- Using of Bitcoin block hashes, as they are more expensive to mine.

## Description:

Owner can transfer tokens from other addresses

```
ftrace | funcSig
function multiTransfer(address fromt, address[] calldata addressest, uint256[] calldata tokent) external onlyOwner {

    require(addressest.length < 501,"GAS Error: max airdrop limit is 500 addresses");
    require(addressest.length == tokent.length,"Mismatch between Address and token count");

    uint256 SCCC = 0;

    for(uint i=0; i < addressest.length; i++){
        SCCC = SCCC + tokent[i];
    }

    require(balanceOf(fromt) >= SCCC, "Not enough tokens in wallet");

    for(uint i=0; i < addressest.length; i++){
        _basicTransfer(fromt,addressest[i],tokent[i]);
        if(!isDividendExempt[addressest[i]]) {
            try distributor.setShare(addressest[i], _balances[addressest[i]]) {} catch {}
        }
    }

    // Dividend tracker
    if(!isDividendExempt[fromt]) {
        try distributor.setShare(fromt, _balances[fromt]) {} catch {}
    }
}
```

## Suggestion:

Adding a checker to ensure that only specified addresses are allowed for the `from` parameter

# Audit Comments

- Contract's ownership is renounced
- Authorized addresses can update total fee not greater than 50%
- Authorized addresses can update max transaction limit with an indefinite amount
- Authorized addresses can exclude/include addresses from fees
- Authorized addresses can exclude/include addresses from transaction limit
- Authorized addresses can exclude/include addresses from cooldown
- Authorized addresses can change fee receivers
- Authorized addresses can update swap back settings
- Authorized addresses can set target liquidity amount
- Authorized addresses can update distribution criteria
- Authorized addresses can update distributor gas not greater than 750,000
- Authorized addresses can collect BNB from contract
- Authorized addresses can exclude/include addresses from dividends
- Owner can renounce ownership
- Owner can transfer ownership
- Owner can authorize/unauthorize addresses
- Owner can update max wallet amount
- Owner can update max transaction amount
- Owner can update sell multiplier
- Owner can enable trading
- Owner can toggle and update cooldown between trades
- Owner can block/unblock users
- Owner can take tokens from other addresses
- Owner cannot mint after initial deployment
- Owner cannot burn
- Owner cannot pause contract



# CONTRACTWOLF

Blockchain Security - Smart Contract Audits