v

CONTRACT WOLF

Blockchain Security - Smart Contract Audits

# Security Assessment

February 26, 2023

# 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

**Arbi Chill Inu** 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 **Arbi Chill Inu**.

# Description

Made with love by the community for the community

# 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

- IUniswapV2Router01
- IUniswapV2Router02
- IUniswapV2Factory
- Context
- Ownable
- Address
- SafeMath
- IERC20
- ARBICHILLINU

# Description

Optimization enabled: Yes

Decimal: 9

Symbol: CHILL

Max / Total Supply: 100,000,000

# Capabilities

## Components

| Version | Contracts | Libraries | Interfaces | Abstract |
|---|---|---|---|---|
| 1.0 | 1 | 2 | 4 | 2 |

## Exposed Functions

| Version | Public | Private | External | Internal |
|---|---|---|---|---|
| 1.0 | 23 | 29 | 54 | 26 |

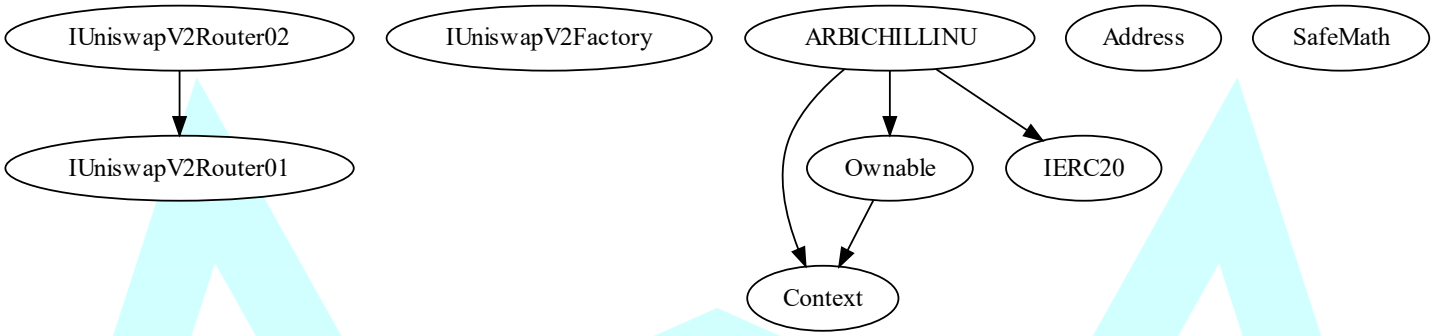
## State Variables

| Version | Total | Public |
|---|---|---|
| 1.0 | 33 | 13 |

## Capabilities

| Version | Solidity Versions Observed | Experimental Features | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|---|---|---|---|---|---|
| 1.0 | v0.8.10 | | Yes | Yes | 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 | – | – | – |
| Burn | – | – | – |
| Block | ✓ | ✓ | ✓ |
| Pause | – | – | – |

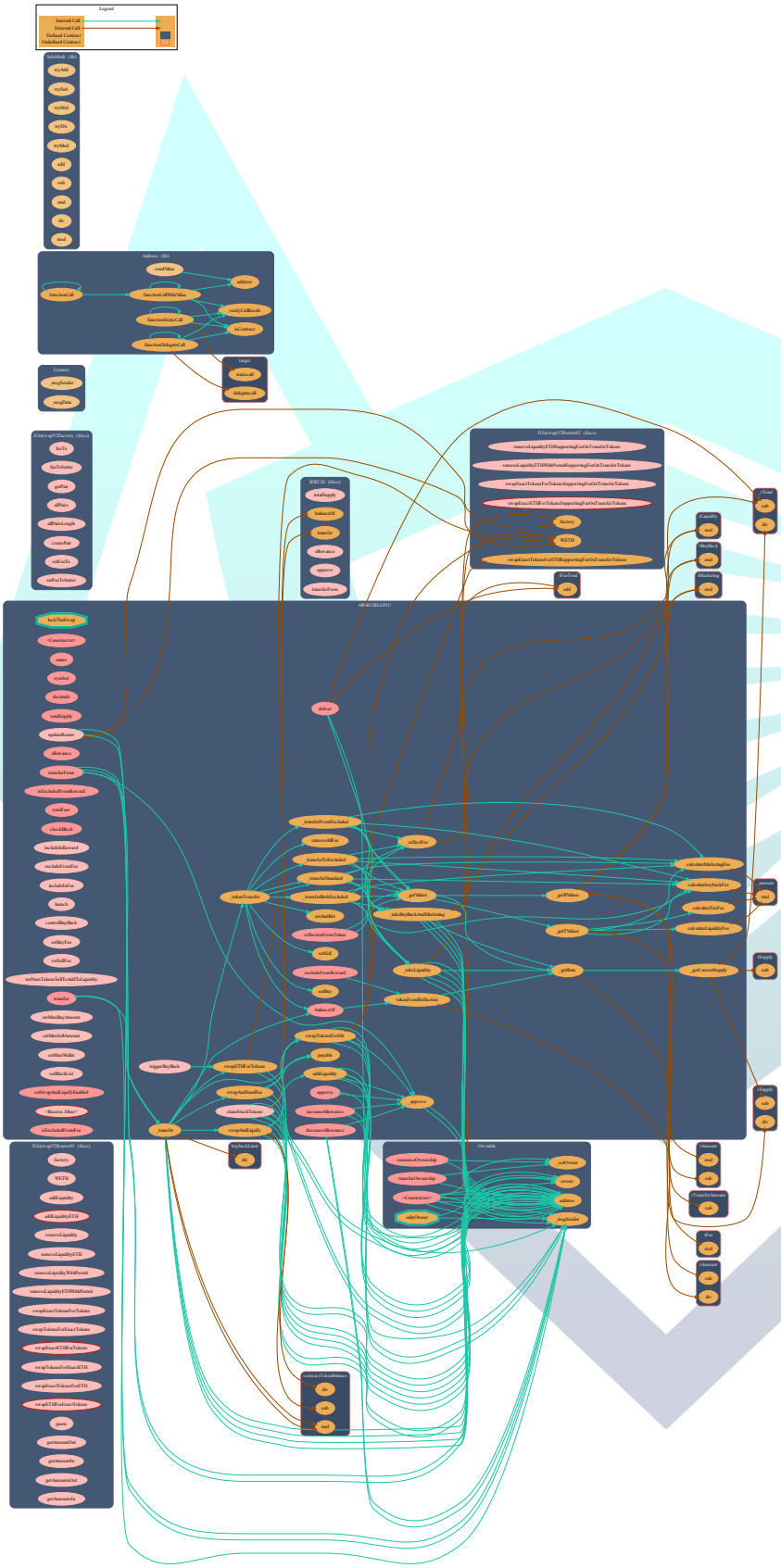
Legend

| Attribute | Symbol |
|---|---|
| Verified / Can | ✓ |
| Verified / Cannot | ✗ |
| Unverified / Not checked | ⚑ |
| Not Available | – |

# Write Functions of Contract

1. approve
2. claimStuckTokens
3. controlBuyBack
4. decreaseAllowance
5. deliver
6. excludeFromFee
7. excludeFromReward
8. includeInFee
9. includeInReward
10. increaseAllowance
11. launch
12. renounceOwnership
13. setBlackList
14. setBuyFee
15. setMaxBuyAmount
16. setMaxSellAmount
17. setMaxWallet
18. setNumTokensSellToAddToLiquidity
19. setSellFee
20. setSwapAndLiquifyEnabled
21. transfer
22. transferFrom
23. transferOwnership
24. triggerBuyBack
25. updateRouter

# Call Graph



# SWC Attacks

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

| | | |
|---|---|---|
| SWC-110 | Assert Violation | **PASSED** |
| SWC-109 | Uninitialized Storage Pointer | **PASSED** |
| SWC-108 | State Variable Default Visibility | **LOW ISSUE** |
| SWC-107 | Reentrancy | **PASSED** |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | **PASSED** |
| SWC-105 | Unprotected Ether Withdrawal | **PASSED** |
| SWC-104 | Unchecked Call Return Value | **PASSED** |
| SWC-103 | Floating Pragma | **LOW ISSUE** |
| SWC-102 | Outdated Compiler Version | **PASSED** |
| SWC-101 | Integer Overflow and Underflow | **PASSED** |
| SWC-100 | Function Default Visibility | **PASSED** |

# AUDIT PASSED

## Low Issues

| | |
|---|---|
| A floating pragma is set (SWC-103) | L: 5, L: 103, L: 149, L: 171, L: 197, L: 271, L: 490, L: 719, L: 800 |
| State variable visibility is not set (SWC-108) | L: 828, L: 857 |
| Potential use of “block.number” as source of randomness (SWC-120) | L: 1038, L: 1118, L: 1589 |

# Findings

**Description:**

A floating pragma is set (SWC-103)

**Suggestion:**

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

---

**Description:**

State variable visibility is not set (SWC-108)

**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 set buy/sell fees up to 100% **(L: 1127, L:1134)**

```
function setBuyFee(uint16 liq↑, uint16 market↑, uint16 reward↑, uint16 tax↑) external onlyOwner {
    buyFee.liquidityFee = liq↑;
    buyFee.marketingandDAO = market↑;
    buyFee.buybackFee = reward↑;
    buyFee.taxFee = tax↑;
}
```

```
function setSellFee(uint16 liq↑, uint16 market↑, uint16 reward↑, uint16 tax↑) external onlyOwner {
    sellFee.liquidityFee = liq↑;
    sellFee.marketingandDAO = market↑;
    sellFee.buybackFee = reward↑;
    sellFee.taxFee = tax↑;
}
```

## Suggestion:

We recommend adding a limiter to set total fees to a maximum value of 25%

# Audit Comments

- Contract is renounced
- Buy fee is set to 5% (Liquidity 1% and Marketing 4%)
- Sell fee is set to 5% (Liquidity 2% and Marketing 3%)
- Owner can include/exclude addresses from rewards
- Owner can include/exclude addresses from fees
- Owner can enable trading
- Owner can toggle buyback status and set buyback limit amount
- Owner can set tokens sold to add to LP
- Owner can update router address
- Owner can set max buy/sell amount up to 100%
- Owner can set max wallet limit up to 100%
- Owner can toggle swap and liquify status
- Owner can trigger buyback manually
- Owner can take tokens/BNB from contract
- Owner cannot burn tokens
- Owner cannot pause contract
- Owner cannot mint after initial deployment

# CONTRACTWOLF

Blockchain Security - Smart Contract Audits