



# CONTRACT WOLF

**Blockchain Security - Smart Contract Audits**

## Security Assessment

October 24, 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>Function Comments</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 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 **Phoenix Revive** 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 **Phoenix Revive**.

## Description

Phoenix Revive is a self-sustained token, via a hard-coded automatic process at every optimized interval: It will relaunch itself by stopping trading, create a new token, pull the liquidity and add it to the new token, existing holders are reflected in the new contract.



# 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
- Ownable
- AddressUpgradeable
- Initializable
- OwnableUpgrade
- ISwapRouter01
- ISwapRouter02
- PhoenixCommon
- IERC20
- IERC20Metadata
- IPhoenixTracker
- IPhoenix
- Numbers
- ISwapFactory
- PhoenixRevive

## Description

Optimization enabled: No

Decimal: 18

## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	3	7	4

### Exposed Functions

Version	Public	Private	External	Internal
1.0	1	4	70	25

### State Variables

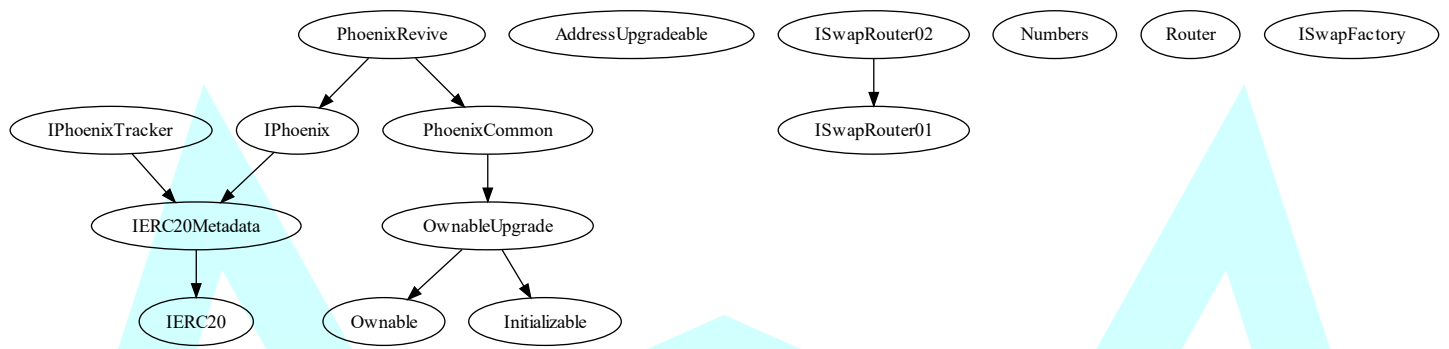
Version	Total	Public
1.0	6	2

### Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.8.16		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	Owner
Renounce Ownership	—	—	—
Mint	—	—	—
Burn	—	—	—
Block	—	—	—
Pause	—	—	—

## Legend

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

# Write Functions of Contract

1. addLiquidity

2. approve

3. endRound

4. initialize

5. setOtherAddr

6. setOwner

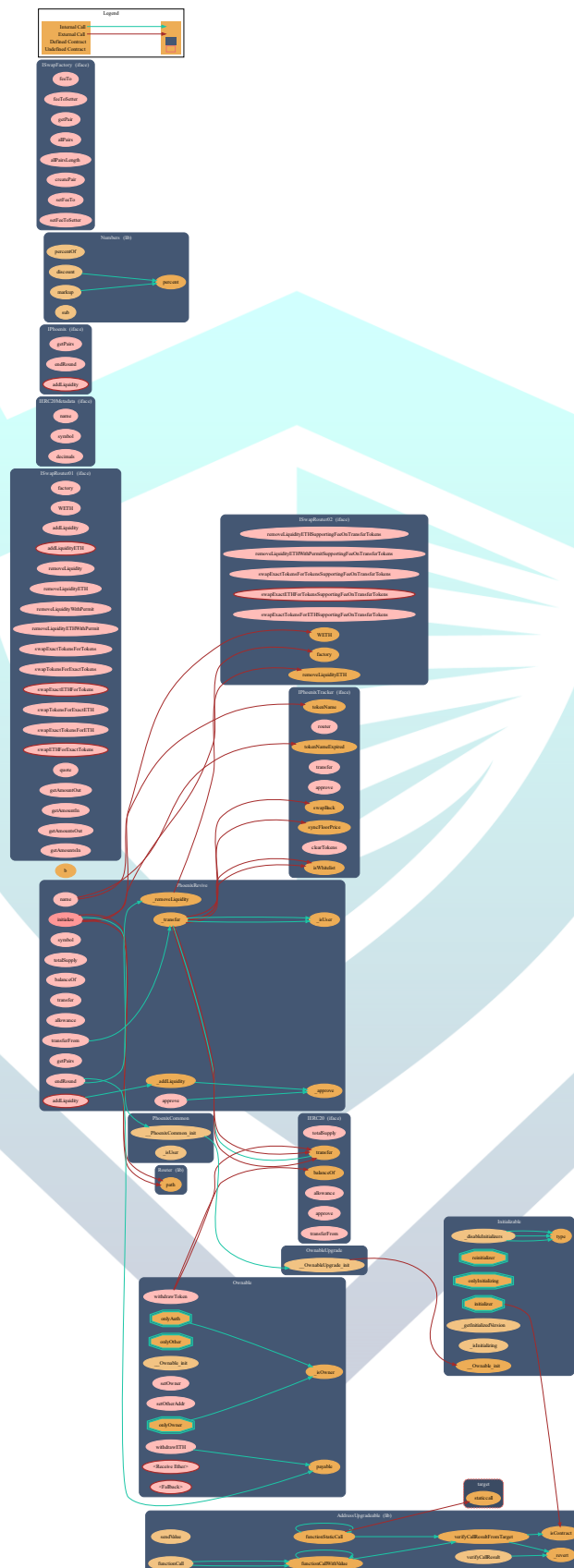
7. transfer

8. transferFrom

9. withdrawETH

10. withdrawToken

# 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	PASSED
<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="#">SWC-110</a>	Assert Violation	PASSED

<a href="#">SWC-109</a>	Uninitialized Storage Pointer	<b>PASSED</b>
<a href="#">SWC-108</a>	State Variable Default Visibility	<b>PASSED</b>
<a href="#">SWC-107</a>	Reentrancy	<b>PASSED</b>
<a href="#">SWC-106</a>	Unprotected SELFDESTRUCT Instruction	<b>PASSED</b>
<a href="#">SWC-105</a>	Unprotected Ether Withdrawal	<b>PASSED</b>
<a href="#">SWC-104</a>	Unchecked Call Return Value	<b>PASSED</b>
<a href="#">SWC-103</a>	Floating Pragma	<b>PASSED</b>
<a href="#">SWC-102</a>	Outdated Compiler Version	<b>PASSED</b>
<a href="#">SWC-101</a>	Integer Overflow and Underflow	<b>PASSED</b>
<a href="#">SWC-100</a>	Function Default Visibility	<b>PASSED</b>

# AUDIT PASSED

## Low Issues

A floating pragma is set (SWC-103)	RESOLVED
Use of “tx.origin” as a part of authorization control (SWC-115)	RESOLVED



# Function Comments

**Status: RESOLVED**

## Description

A floating pragma is set (SWC-103)

## Suggestion:

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

---

**Status: RESOLVED**

## Description

Use of “tx.origin” as a part of authorization control (SWC-115)

## Suggestion:

tx.origin should not be used for authorization. Use msg.sender instead.

# Audit Comments

- Contract is a proxy contract
- Owner can change “**other**” address
- Owner can withdraw BNB from contract
- Owner can withdraw tokens from contract
- Owner may change fees via tracker address
- Owner may burn tokens via tracker address
- Authorized address can pause contract via endRound function
- Authorized address can change owner address
- Authorized address can add liquidity amount
- “**Other**” address can end round and take tokens from contract
- Owner cannot renounce ownership
- Owner cannot mint after initial deployment
- Owner cannot block users



# CONTRACTWOLF

Blockchain Security - Smart Contract Audits