CONTRACT WOLF

Blockchain Security - Smart Contract Audits

Security Assessment

August 23, 2022

Disclaimer 3

Scope of Work & Engagement 3

Project Description 4

Risk Level Classification 5

Methodology 6

Used Code from other Frameworks / Smart Contracts (Imports) 7

Token Description 8

Inheritance Graph 9

Overall Checkup 10

Verify Claim 11

Write Functions of Contract 12

Call Graph 13

SWC Attacks 14

Audit Result 16

Audit Comments 18

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

Gempad's 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 **Gempad.**

Description

Gempad's Dividend Tokens has max wallet, max transaction, marketing, liquidity, reward fees. Fees combined can't be more than 30% and has a setting for minimum tokens to hold to get rewards.

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

- AddressUpgradeable
- Initializable
- ContextUpgradeable
- IERC20Upgradeable
- IERC20MetadataUpgradeable
- IterableMapping
- SafeMathUint
- SafeMathInt
- SafeMath
- IUniswapV2Pair
- OwnableUpgradeable
- ERC20Upgradeable
- Context
- IERC20
- IERC20Metadata
- IUniswapRouter01
- IUniswapRouter02
- IUniswapV2Factory
- Clones
- Ownable
- ERC20
- DividendPayingTokenInterface
- DividendPayingTokenOptionalInterface
- DividendPayingToken
- DividendTokenDividendTracker
- IPancakeCaller
- DividendToken

Description

Optimization enabled: Yes

Decimal: 18

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	6	11	5

Exposed Functions

Version	Public	Private	External	Internal
1.0	58	9	113	67

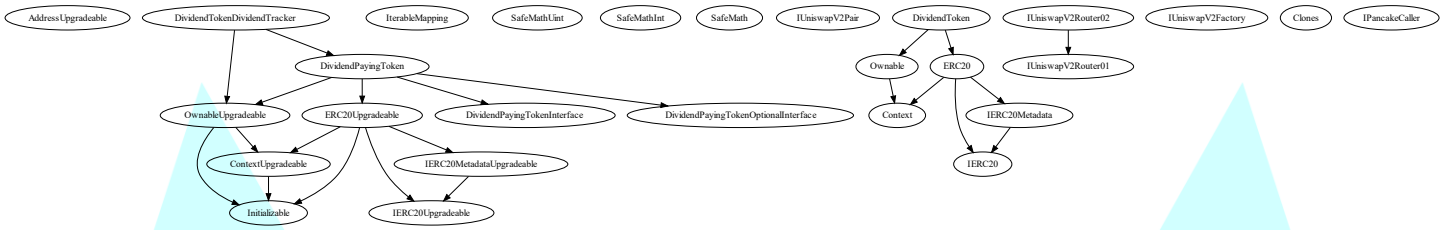
State Variables

Version	Total	Public
1.0	49	24

Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.8.13		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	–	–	–

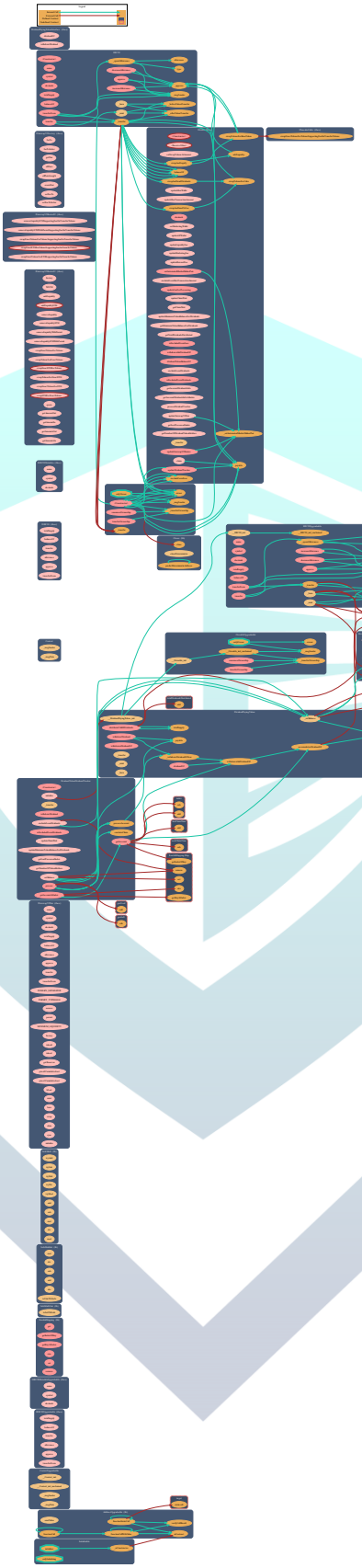
Legend

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

Write Functions of Contract

1. approve
2. claim
3. decreaseAllowance
4. excludeFromDividends
5. excludeFromFees
6. excludeFromMaxTransactionAmount
7. increaseAllowance
8. processDividendTracker
9. renounceOwnership
10. setAutomatedMarketMakerPair
11. setMarketingWallet
12. setSwapTokensAtAmount
13. transfer
14. transferFrom
15. transferOwnership
16. updateClaimWait
17. updateDividendTracker
18. updateGasForProcessing
19. updateLPWallet
20. updateLiquidityFee
21. updateMarketingFee
22. updateMaxTransactionAmount
23. updateMaxWallet
24. updateMinimumTokenBalanceForDividends
25. updateRewardFee
26. updateUniswapV2Pair
27. updateUniswapV2Router

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	PASSED
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	**PASSED**
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: 2	DividendToken.sol IUniswapV2Factory.sol IUniswapV2Router02.sol DividendTokenDividendTracker.sol IUniswapV2Pair.sol SafeMathInt.sol SafeMathUint.sol IterableMapping.sol
	L: 4	ERC20.sol Ownable.sol Clones.sol IERC20.sol IERC20Metadata.sol Context.sol ERC20Upgradeable.sol OwnableUpgradeable.sol SafeMath.sol IERC20Upgradeable.sol IERC20MetadataUpgradeable.sol ContextUpgradeable.sol Initializable.sol AddressUpgradeable.sol

Audit Comments

- Owner can change total sell fee not greater than 30%
- Owner can change total buy fee not greater than 30%
- Owner can update max wallet amount greater than 0
- Owner can update max transaction amount greater than 0
- Owner can renounce ownership
- Owner can transfer ownership
- Owner can exclude addresses from dividends
- Owner can update minimum token balance for dividends amount
- Owner can update swap tokens at amount
- Owner can change dividend tracker address
- Owner can change UniswapV2RouterPair address
- Owner can change UniswapV2Router address
- Owner can exclude/include addresses from fees
- Owner can change marketing wallet address
- Owner can change liquidity pool address
- Owner can change automated market maker pair address
- Owner can exclude/include addresses from max transaction amount
- Owner can update gas for processing between 200,000 and 500,000
- Owner can update claim wait time
- Owner cannot mint after initial deployment
- Owner cannot burn
- Owner cannot block users
- Owner cannot pause contract

CONTRACTWOLF

Blockchain Security - Smart Contract Audits