CONTRACT WOLF

Blockchain Security - Smart Contract Audits

Security Assessment

June 29, 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 <u>decision</u> 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

Shiru 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 **Shiru.**

Description

Shiru Token is a utility token and a product of ShiruPal (Pty) Ltd.

With so many complex processes, high gas fees, loopholes and vulnerabilities exploited daily, we recognise the need to help change the future of cryptocurrencies.

Shiru Token is only 1 one of the 8 planned products of ShiruPal (Pty) Ltd.

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
- IERC20
- IERC20Metadata
- IterableMapping
- DividendPayingTokenOptionalInterface
- DividendPayingTokenInterface
- SafeMath
- SafeMathUint
- SafeMathInt
- ERC20
- IUniswapV2Router01
- IUniswapV2Router02
- IUniswapV2Factory
- IUniswapV2Pair
- Ownable
- DividendPayingToken2
- DividendTracker
- Shiru

Description

Optimization enabled: Yes

Decimal: 18

Symbol: SHIRU

Max / Total supply: 400,000,000,000,000

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	4	8	2

Exposed Functions

Version	Public	Private	External	Internal
1.0	49	6	97	30

State Variables

Version	Total	Public
1.0	42	19

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

DividendTracker · IterableMapping · SafeMathInt · SafeMathUint · SafeMath · IUniswapV2Router02 · IUniswapV2Factory · IUniswapV2Pair

Shiru

DividendPayingToken2

IUniswapV2Router01

Ownable · ERC20 · DividendPayingTokenOptionalInterface · DividendPayingTokenInterface

Context · IERC20Metadata

IERC20

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. blacklistAddress

3. claim

4. decreaseAllowance

5. excludeFromDividends

6. excludeFromFees

7. excludeFromMaxWallet

8. excludeMultipleAccountsFromFees

9. increaseAllowance

10. processDividendTracker

11. recoverBEP20

12. renounceOwnership

13. setAutomatedMarketMakerPair

14. setBuyFees

15. setMarketingWallet

16. setSellFees

17. setSwapEnabled

18. setSwapTokensAtAmount

19. setTransferFees

20. transfer

21. transferFrom

22. transferOperator

23. transferOwnership

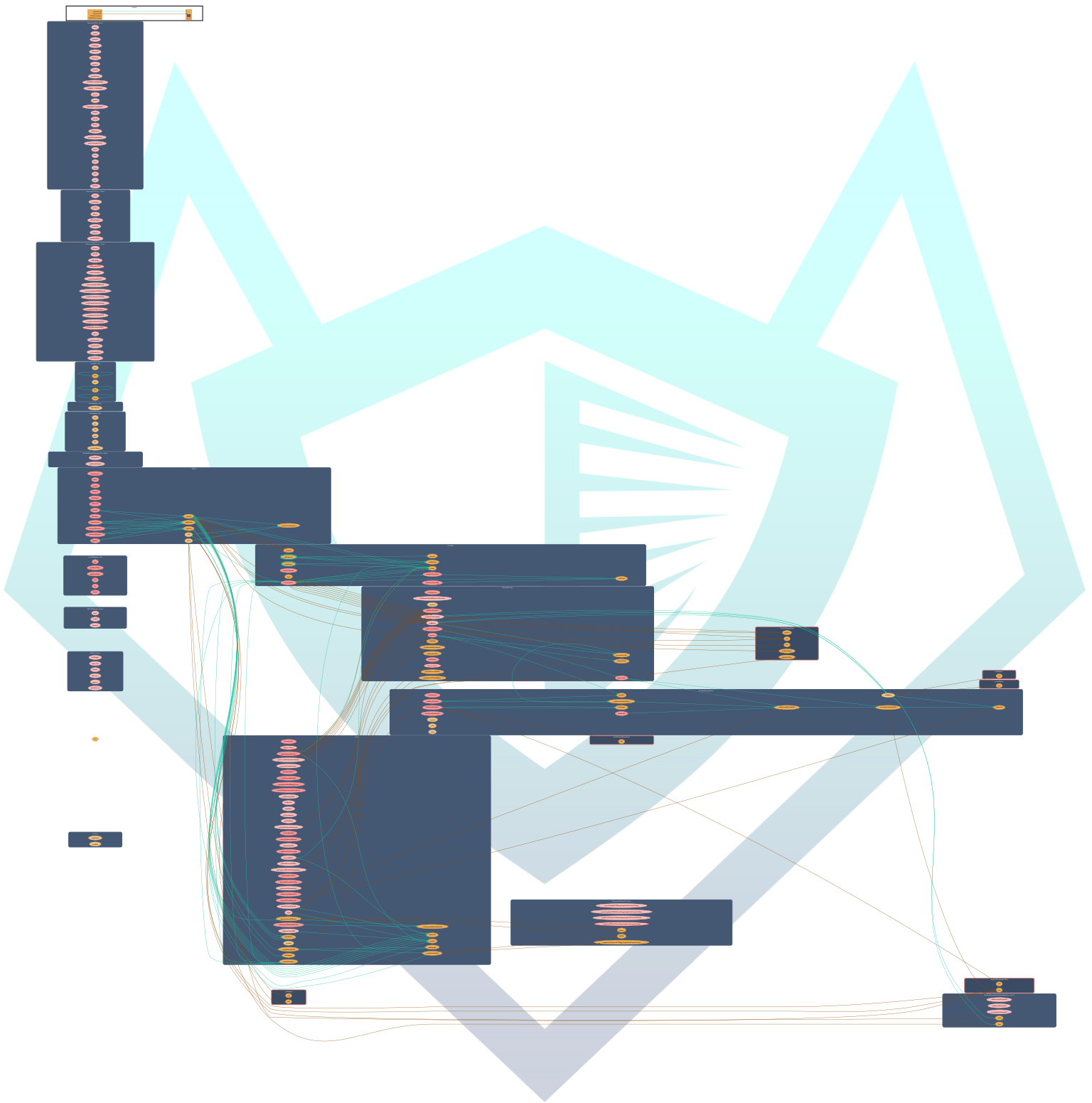24. updateClaimWait

25. updateDividendTracker

26. updateGasForProcessing

27. updateMaxWalletHoldingPercent

28. updateUniswapV2Router

29. withdrawStuckBNB

Call Graph

SWC Attacks

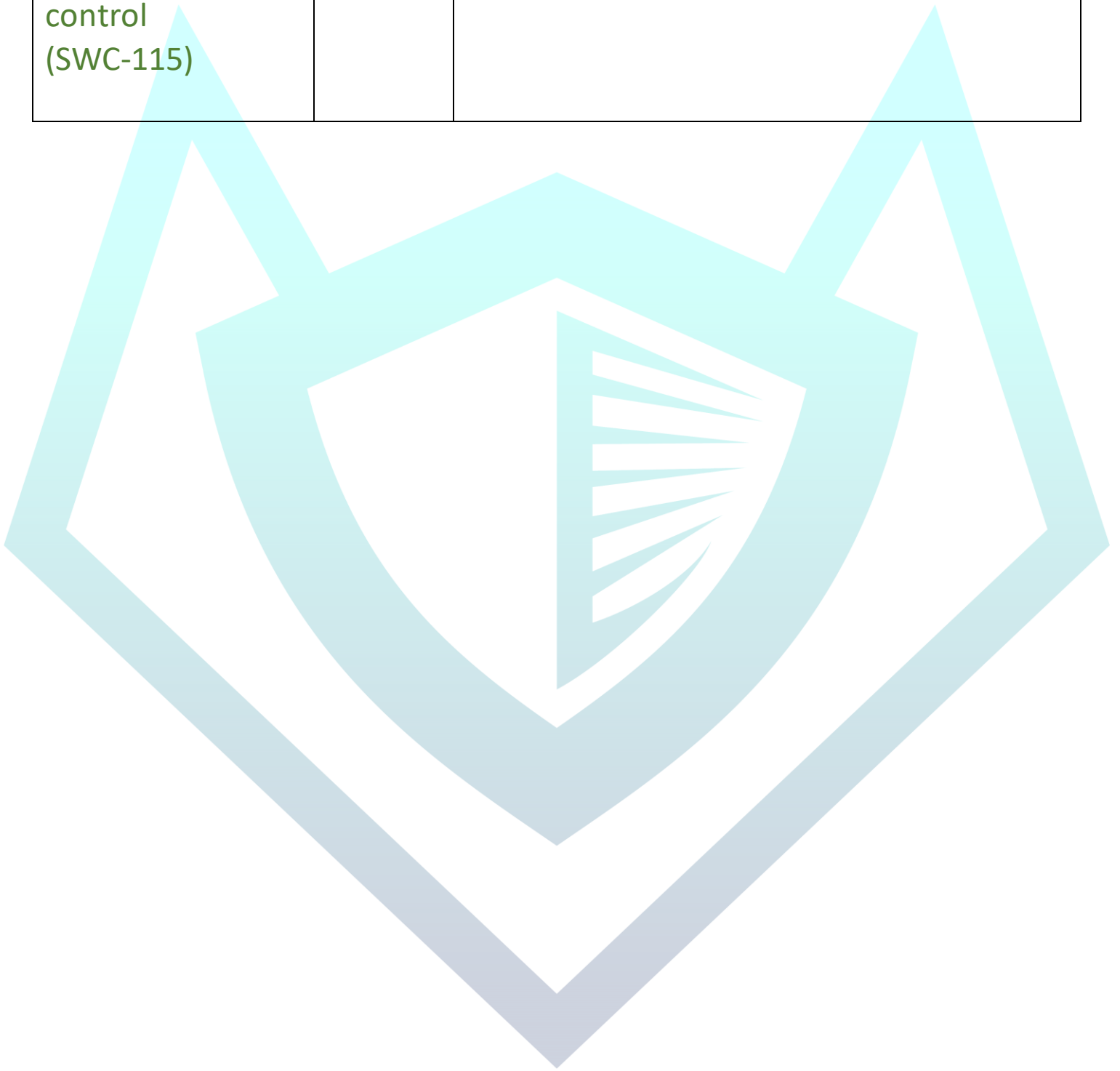| ID	Title	Status
SWC-136	Unencrypted Private Data On-Chain	PASSED
SWC-135	Code With No Effects	PASSED
SWC-134	Message call with hardcoded gas amount	PASSED
SWC-133	Hash Collisions with Multiple Variable Length Arguments	PASSED
SWC-132	Unexpected Ether balance	PASSED
SWC-131	Presence of unused variables	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	LOW ISSUE
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

	L: 2	Ownable.sol IterableMapping.sol
A floating pragma is set (SWC-103)	L: 3	ShiruV2.sol DividendPayingToken2.sol IUniswapV2Pair.sol IUniswapV2Factory.sol IUniswapV2Router.sol ERC20.sol SafeMath.sol SafeMathUint.sol DividendPayingTokenInterface.sol DividendPayingTokenOptionalInterface.sol IERC20.sol IERC20Metadata.sol Context.sol
	L: 28	SafeMathInt.sol

Use of "tx.origin" as a part of authorization control (SWC-115)	L: 342 L: 430	ShiruV2.sol

Audit Comments

- Deployer can renounce ownership
- Deployer can update dividend tracker address
- Deployer can update router address
- Deployer can update max wallet holding up to 100%
- Deployer can update claim cooldown
- Deployer can update gas for processing with an amount not greater than 500,000 and not less than 100,000
- Deployer can set swap tokens
- Deployer can set automated market maker pair address and status
- Deployer can set marketing wallet address
- Deployer can set fees up to 28%
- Deployer can include/exclude addresses from fees
- Deployer can include/exclude addresses from dividends
- Deployer can include/exclude addresses from max wallet limit
- Deployer can take tokens from contract
- Deployer can take BNB from contract
- Deployer can toggle swap
- Deployer can swap tokens from contract to liquidity
- Deployer can block/unblock user address
- Deployer cannot burn tokens
- Deployer cannot pause contract
- Deployer cannot mint after initial deployment

- Operator can transfer ownership
- Operator can transfer operator
- Operator can set minimum tokens balance for dividends
- Operator can include/exclude address from dividends
- Operator can set claim wait

CONTRACTWOLF

Blockchain Security - Smart Contract Audits