CONTRACT WOLF

Security Assessment

Shiru Protocol Token

Verified on 05/23/2023

SUMMARY

Project	CHAIN	METHODOLOGY
Shiru Protocol Token	BSC	Manual & Automatic Analysis

FILES	DELIVERY	TYPE
Single	05/23/2023	Standard Audit

5	1	0	0	0	4
Total Findings	Critical	Major	Medium	Minor	Informational

🟥 1 Critical	1 Pending	An exposure that can affect the contract functions in several events that can risk and disrupt the contract
🟧 0 Major	0 Pending	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
🟧 0 Medium	0 Pending	An opening that could affect the outcome in executing the contract in a specific situation
⬜ 0 Minor	0 Pending	An opening but doesn't have an impact on the functionality of the contract
🟦 4 Informational	4 Pending	An opening that consists information but will not risk or affect the contract

STATUS ❌ **AUDIT FAILED**

TABLE OF CONTENTS | Shiru Protocol Token

DISCLAIMER | Shiru Protocol Token

ContractWolf audits and reports should not be considered as a form of project's "Advertisement" and does not cover any interaction and assessment from "Project Contract" to "External Contracts" such as PancakeSwap, UniSwap, SushiSwap or similar.

ContractWolf does not provide any warranty on its released report and should not be used as a decision to invest into audited projects.

ContractWolf provides a 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's presence is to analyze, audit and assess the Client's Smart Contract to find any underlying risk and to eliminate any logic and flow errors within its code.

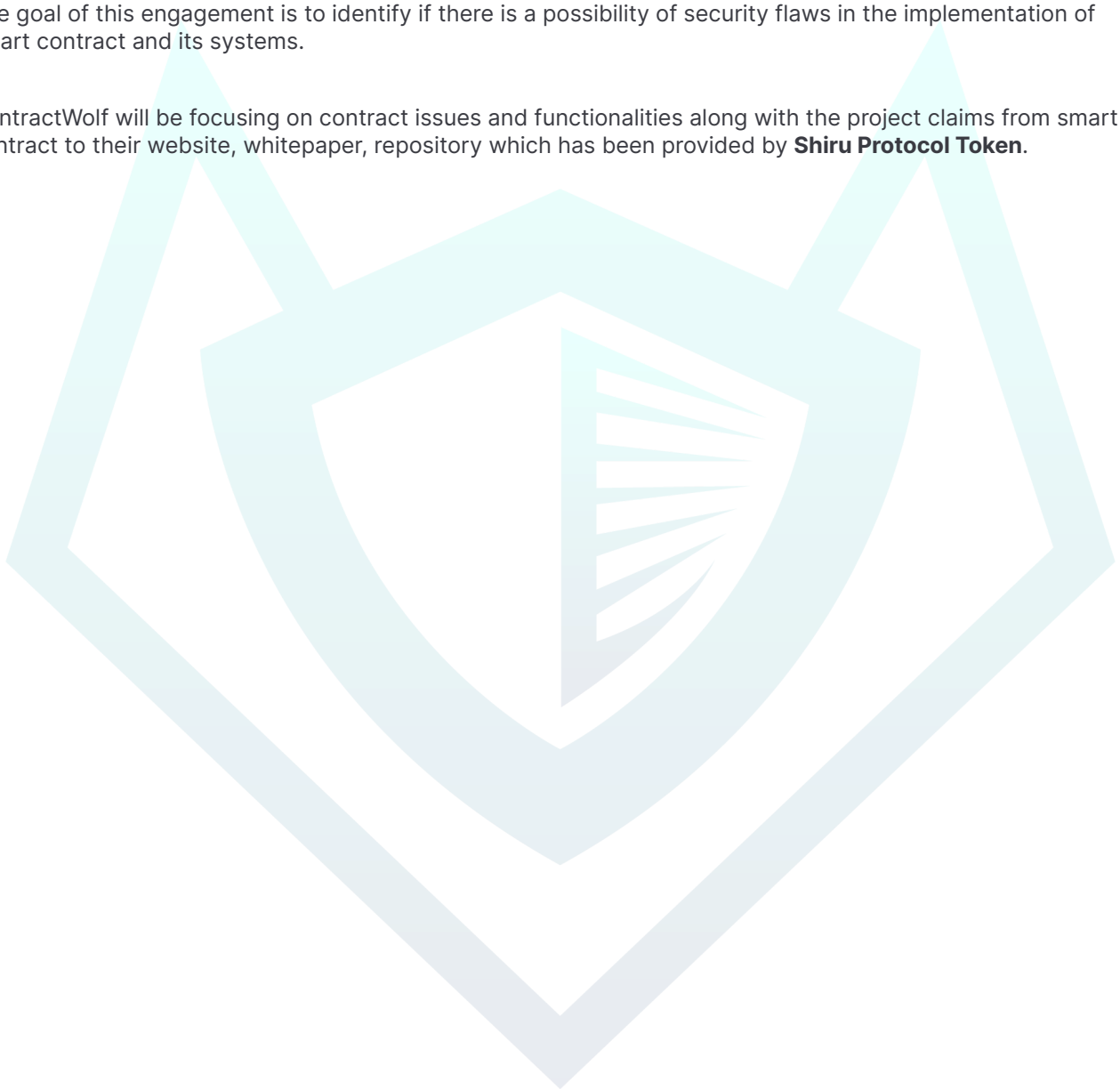
Each company or project should be liable to its security flaws and functionalities.

SCOPE OF WORK | Shiru Protocol Token

Shiru Protocol Token team has agreed and provided us with the files that need to be tested (*Github, BSCscan, Etherscan, Local files etc*). The scope of audit is the main contract.

The goal of this engagement is to identify if there is a possibility of security flaws in the implementation of smart contract and its systems.

ContractWolf will be focusing on contract issues and functionalities along with the project claims from smart contract to their website, whitepaper, repository which has been provided by **Shiru Protocol Token**.



AUDITING APPROACH | Shiru Protocol Token

Every line of code along with its functionalities will undergo manual review to check for security issues, quality of logic and contract scope of inheritance. The manual review will be done by our team that will document any issues that they 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, underlying and hidden security flaws.

2. Testing and automated analysis that includes :
   - Testing the smart contract function with common test cases and scenarios to ensure that it returns the expected results.

3. Best practices and ethical 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 eliminate or minimize threats and secure the smart contract.

TOKEN DETAILS | Shiru Protocol Token

Simplicity is the Ultimate Sophistication With so many complex processes, high gas fees, loopholes and vulnerabilities exploited daily, we recognize the need to help change the future.

Token Name	Symbol	Decimal	Total Supply	Chain
Shiru Protocol	SHP	18	1,000,000,000,000,000	BSC

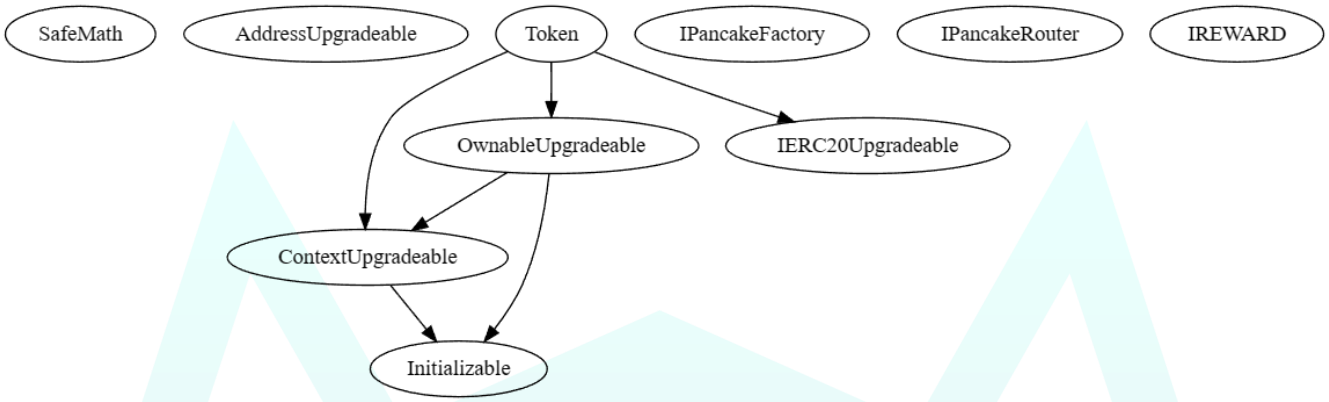
SOURCE

Source	*Sent Via local-files*

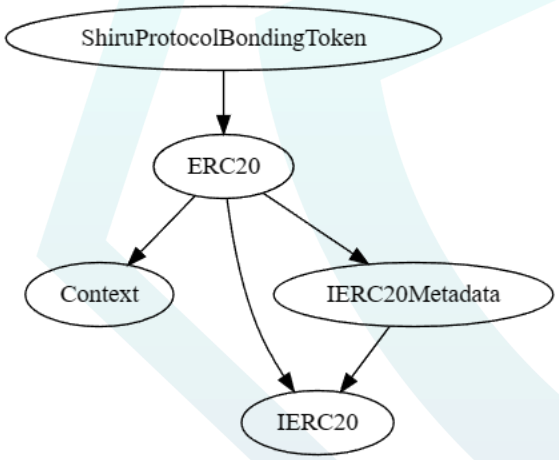
INHERITANCE GRAPH | Shiru Protocol Token

Inheritance Graph of Contract Functions

Token.sol



BondingToken.sol

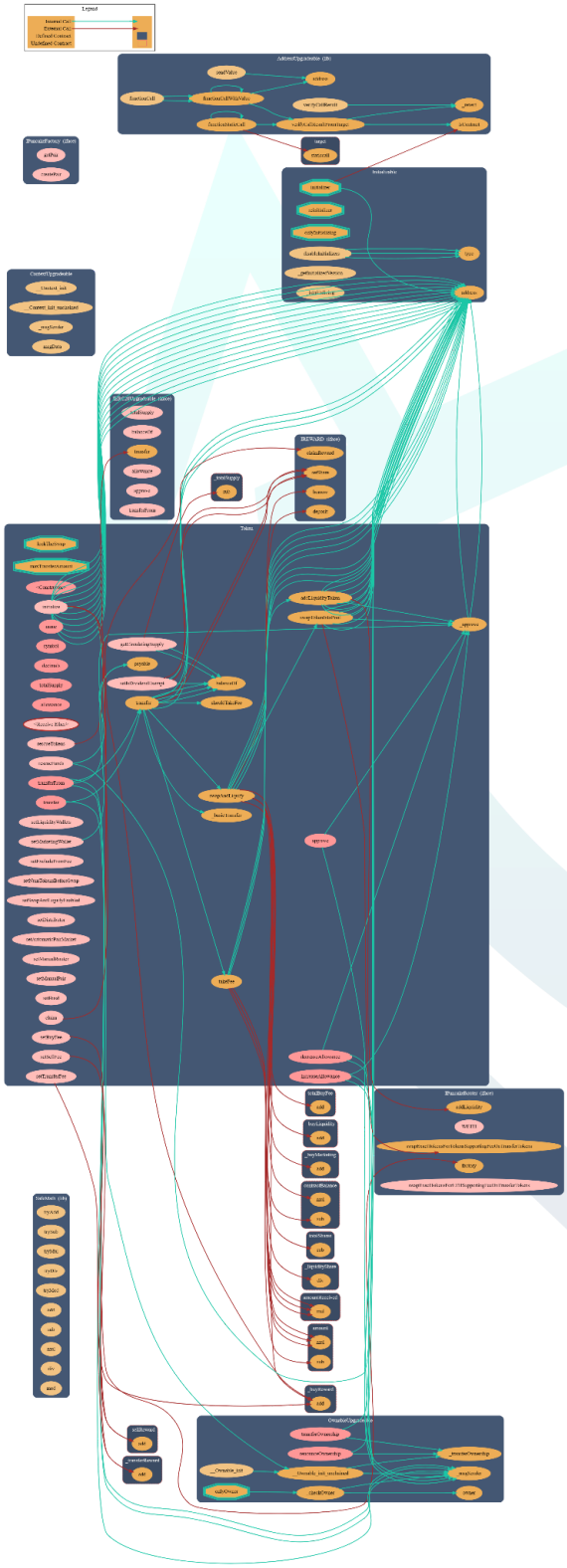


Reflections.sol

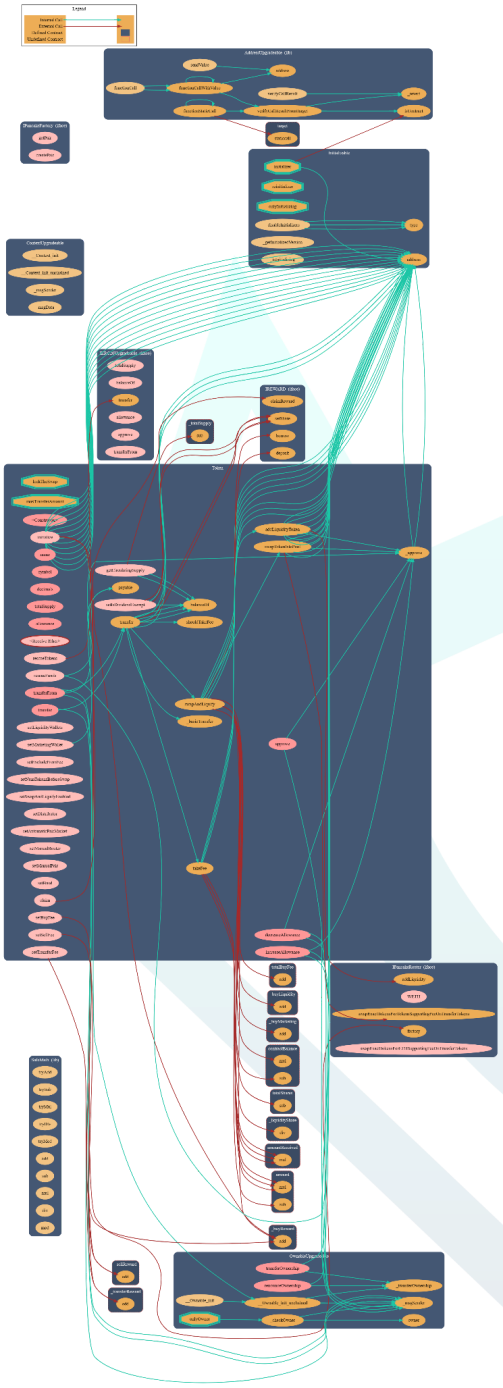
CALL GRAPH | Shiru Protocol Token

Call Graph of Contract Functions

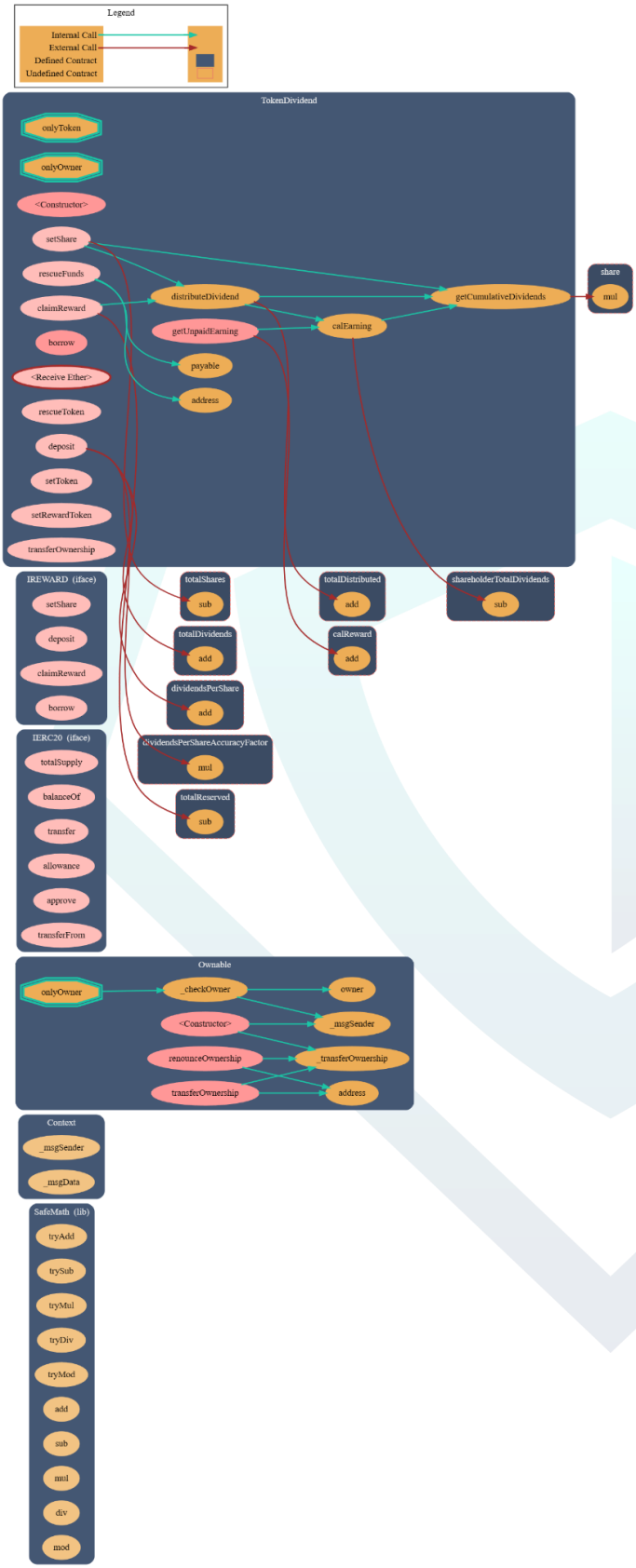
Token.sol



BondingToken.sol



Reflections.sol



FINDINGS | Shiru Protocol Token

5	1	0	0	0	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to state the issues and vulnerabilities for Shiru Protocol Token through this audit. The goal of this report findings is to identify specifically and fix any underlying issues and errors

ID	Title	File & Line #	Severity	Status
	Initializer Modifier in Constructor	Token.sol, L: 139	Critical	• Pending
SWC-103	Floating Pragma is set	Token.sol, BondingToken.sol , Reflections L: 3	Informational	• Pending
CW-011	Commented Code	Token.sol, L: 313, 329	Informational	• Pending
CW-012	SafeMath Override	SafeMath.sol	Informational	• Pending
	Boolean Equality	Token.sol, L: 132	Informational	• Pending

SWC ATTACKS | Shiru Protocol Token

Smart Contract Weakness Classification and Test Cases

ID	Description	Status
SWC-100	Function Default Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Not Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF DESTRUCT Instruction	Passed
SWC-107	Reentrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed

ID	Description	Status
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character(U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

CW ASSESSMENT | Shiru Protocol Token

ContractWolf Vulnerability and Security Tests

ID	Name	Description	Status
CW-001	Multiple Version	Presence of multiple compiler version across all contracts	✔
CW-002	Incorrect Access Control	Additional checks for critical logic and flow	✔
CW-003	Payable Contract	A function to withdraw ether should exist otherwise the ether will be trapped	✔
CW-004	Custom Modifier	major recheck for custom modifier logic	✔
CW-005	Divide Before Multiply	Performing multiplication before division is generally better to avoid loss of precision	✔
CW-006	Multiple Calls	Functions with multiple internal calls	✔
CW-007	Deprecated Keywords	Use of deprecated functions/operators such as block.blockhash() for blockhash(), msg.gas for gasleft(), throw for revert(), sha3() for keccak256(), callcode() for delegatecall(), suicide() for selfdestruct(), constant for view or var for actual type name should be avoided to prevent unintended errors with newer compiler versions	✔
CW-008	Unused Contract	Presence of an unused, unimported or uncalled contract	✔
CW-009	Assembly Usage	Use of EVM assembly is error-prone and should be avoided or double-checked for correctness	✔
CW-010	Similar Variable Names	Variables with similar names could be confused for each other and therefore should be avoided	✔
CW-011	Commented Code	Removal of commented/unused code lines	✘
CW-012	SafeMath Override	SafeMath is no longer needed starting Solidity v0.8+. The compiler now has Built in overflow checking.	✘

FIXES & RECOMMENDATION

Initializer Modifier in Constructor

The presence of initialization logic within the constructor prevents the execution of the initialization code defined in the initializer write function. The constructor takes precedence over the initializer write function, and as a result, the initialization process defined in the initializer will not be triggered.

```
constructor() initializer {}
```

Suggestion

Choose either to remove the initializer from the constructor or move its code into the constructor itself, as having initialization logic in both places won't execute the initializer write function. Consider your requirements to decide the best approach for your contract's initialization.

SWC-103 | A Floating Pragma is Set

Code

```
pragma solidity ^0.8.17;
```

The compiler version should be a fixed one to avoid undiscovered compiler bugs. Fixed version sample below

Suggestion

```
pragma solidity 0.8.17;
```

CW-011 | Commented Code

Consider removing commented code in Solidity contracts to enhance code readability and cleanliness.

```
// uint _BuyBackShare = _buyBuyBack.add(_sellBuyBack).add(_transferBuyBack);
// uint256 amountBUSDHolding =
amountReceived.sub(amountBUSDLiquidity).sub(amountBUSDMarketing).sub(amountBUSDR
eward);
```

Boolean Equality

Boolean constants can be used directly and do not need to be compared to true or false.

```
modifier maxTransferAmount(uint256 amount) {
    if (isExcludedFromFee[msg.sender] == false){
        require(amount <= maxTxAmount, "Transfer amount exceeds the
maxTxAmount.");
    }
    _;
}
```

Suggestion

Remove the `== false` as it is not needed

CW-012 | SafeMath Override

```
library SafeMath
```

SafeMath is no longer needed starting Solidity v0.8+. The compiler now has Built in overflow checking.

Taxes can be updated to 100%

Owner can update total buy fees, total sell fees, and total transfer fee with an indefinite amount

Recommendation

Adding validation checks to the setFees function to ensure that the sum of all fees is not greater to a specific percentage. This will help prevent errors and ensure that the contract works as intended.

AUDIT COMMENTS | Shiru Protocol Token

Smart Contract audit comment for a non-technical perspective

Token.sol

🟢 Owner can renounce and transfer ownership
🟢 Owner can collect BNB and tokens from contract
🟢 Owner can change liquidity wallet receiver
🟢 Owner can change marketing wallet receiver
🟢 Owner can exclude/include addresses from fees
🟢 Owner can set minimum tokens before swapping
🟢 Owner can toggle swap and liquify
🟢 Owner can change reward distributor address
🟢 Owner can exclude/include addresses from market pair and dividends
🟢 Owner can change pancake router and pair address
🟢 Owner can change BUSD address
🟢 Owner can update total buy fees, total sell fees, and total transfer fee with an indefinite amount
🟢 Owner cannot burn tokens
🟢 Owner cannot pause contract
🟢 Owner cannot mint after initial deployment
🟢 Owner cannot set max transaction limit
🟢 Owner cannot block users

BondingToken.sol

🟢 Bonding contract can mint bonding tokens after initial deployment
🟢 Bonding contract can burn minted tokens

TokenDividend.sol

🟢 Token address can set shares for dividends
🟢 Token address can deposit dividends
🟢 Token address can borrow tokens from reward token contract
🟢 Owner can collect BNB and tokens from contract
🟢 Owner change token address for dividends
🟢 Owner change reward token address
🟢 Owner can transfer ownership
🟢 Owner cannot renounce ownership

CONTRACTWOLF

Blockchain Security - Smart Contract Audits