Cyberscope

Audit Report

FirmSeed

October 2023

Network	BSC
Address	0x9de358f91d89feb6bb156fa2572c09a66c29c2bc
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MRM	Missing Revert Messages	Unresolved
	PVC	Price Volatility Concern	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L07	Missing Events Arithmetic	Unresolved

Table of Contents

Review

Contract Name	FirmSeed
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization	200 runs
Explorer	https://bscscan.com/address/0x9de358f91d89feb6bb156fa2572c09a66c29c2bc
Address	0x9de358f91d89feb6bb156fa2572c09a66c29c2bc
Network	BSC
Symbol	FIRM
Decimals	9
Total Supply	50,000,000

Audit Updates

Initial Audit	02 Oct 2023

Source Files

Filename	SHA256
FirmSeed.sol	18af47030d17848662a82aa656a8612baaf626d617a8e6af98b1bf134e7e3bc

Findings Breakdown

Critical	2
Medium	0
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	FirmSeed.sol#L275,279
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users excluding specific addresses.. Once the transactions are enable the owner will not be able to disable them again.

Additionally, the contract owner has the authority to stop the sales for all users excluding specific addresses. The owner may take advantage of it by setting the `txMax` or `walletMax` to zero. As a result, the contract may operate as a honeypot.

```
if(!isTxLimitExempt[sender] && !isTxLimitExempt[recipient]) {
    require(launched, "Not Launched.");

    if(txLimitsEnabled) {
    require(amount <= txMax, "Transfer amount exceeds the max
limit.");
    }
}
...
if(checkWalletLimit && !isWalletLimitExempt[recipient])
    require((balanceOf(recipient) + finalAmount) <= walletMax);
```

Recommendation

The contract could embody a check for not allowing setting the `txMax` and `walletMax` less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. The team should carefully manage the private keys of the owner's account. We strongly

recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	FirmSeed.sol#L212
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `openTrading` function with a high percentage value.

```
   function openTrading(uint256 buyTax_, uint256 sellTax_)
public onlyOwner {
        launched = true;
        buyTax = buyTax_;
        sellTax = sellTax_;
    }
```

Recommendation

The contract could embody a check for the maximum acceptable value. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

MRM - Missing Revert Messages

Criticality	Minor / Informative
Location	FirmSeed.sol#L301
Status	Unresolved

Description

The contract is missing error messages. These missing error messages are making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require((balanceOf(recipient) + finalAmount) <= walletMax);
```

Recommendation

The team is advised to carefully review the source code in order to address these issues. To accelerate the debugging process and mitigate these issues, the team should use more specific and descriptive error messages.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	FirmSeed.sol#L237,242,251
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
    function changeWalletLimits(bool checkWalletLimit_, uint256
walletMax_) external onlyOwner {
        checkWalletLimit = checkWalletLimit_;
        walletMax  = walletMax_;
    }

    function whitelistAccounts(address[] memory wallets, bool
feeExempt, bool walletLimitExempt, bool txLimitExempt) public
onlyOwner {
        for(uint256 i = 0; i < wallets.length; i++){
            isExcludedFromFee[wallets[i]] = feeExempt;
            isWalletLimitExempt[wallets[i]] =
walletLimitExempt;
            isTxLimitExempt[wallets[i]] = txLimitExempt;
            emit AccountWhitelisted(wallets[i], feeExempt,
walletLimitExempt, txLimitExempt);
        }
    }

    function changeSwapSettings(bool swapEnabled_, uint256
swapThreshold_, bool swapByLimitOnly_) public onlyOwner {
        swapEnabled = swapEnabled_;
        swapThreshold = swapThreshold_;
        swapByLimitOnly = swapByLimitOnly_;
        emit SwapSettingsUpdated(swapEnabled_, swapThreshold_,
swapByLimitOnly_);
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	FirmSeed.sol#L288,251
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
uint256 contractTokenBalance = balanceOf(address(this));
bool overMinimumTokenBalance = contractTokenBalance >=
swapThreshold;
if(overMinimumTokenBalance) {
    if(swapByLimitOnly)
        contractTokenBalance = swapThreshold;
    swapAndLiquify(contractTokenBalance);
}
```

```
   function changeSwapSettings(bool swapEnabled_, uint256
swapThreshold_, bool swapByLimitOnly_) public onlyOwner {
        swapEnabled = swapEnabled_;
        swapThreshold = swapThreshold_;
        swapByLimitOnly = swapByLimitOnly_;
        emit SwapSettingsUpdated(swapEnabled_, swapThreshold_,
swapByLimitOnly_);
    }
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	FirmSeed.sol#L127,128
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
dexRouter
lpPair
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	FirmSeed.sol#L50,77,78,79,97
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
string constant private _name = "FirmSeed"
string constant private _symbol = "FIRM"
uint8 constant private _decimals = 9
uint256 constant private _totalSupply = 50 * 10**6 *
10**_decimals
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	FirmSeed.sol#L214,239
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
buyTax = buyTax_
walletMax  = walletMax_
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

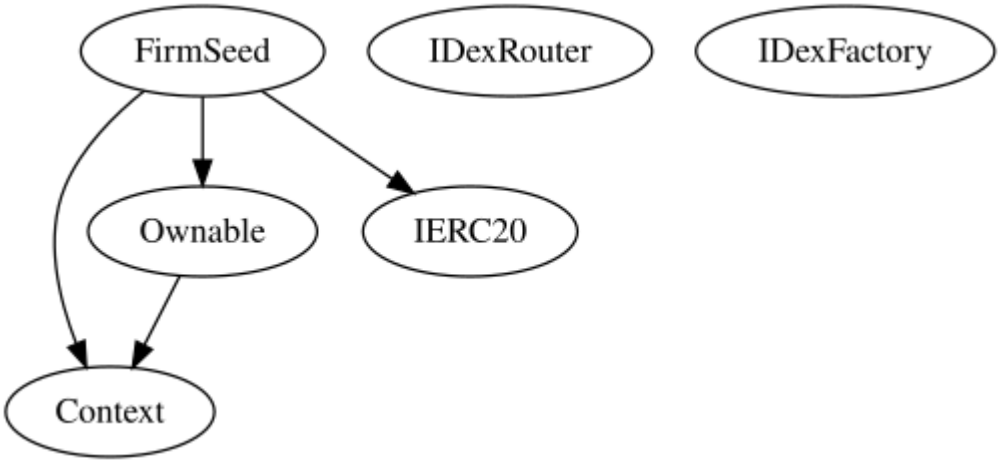
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	addLiquidityETH	External	Payable	-
	removeLiquidityETH	External	✓	-
	getAmountsOut	External		-

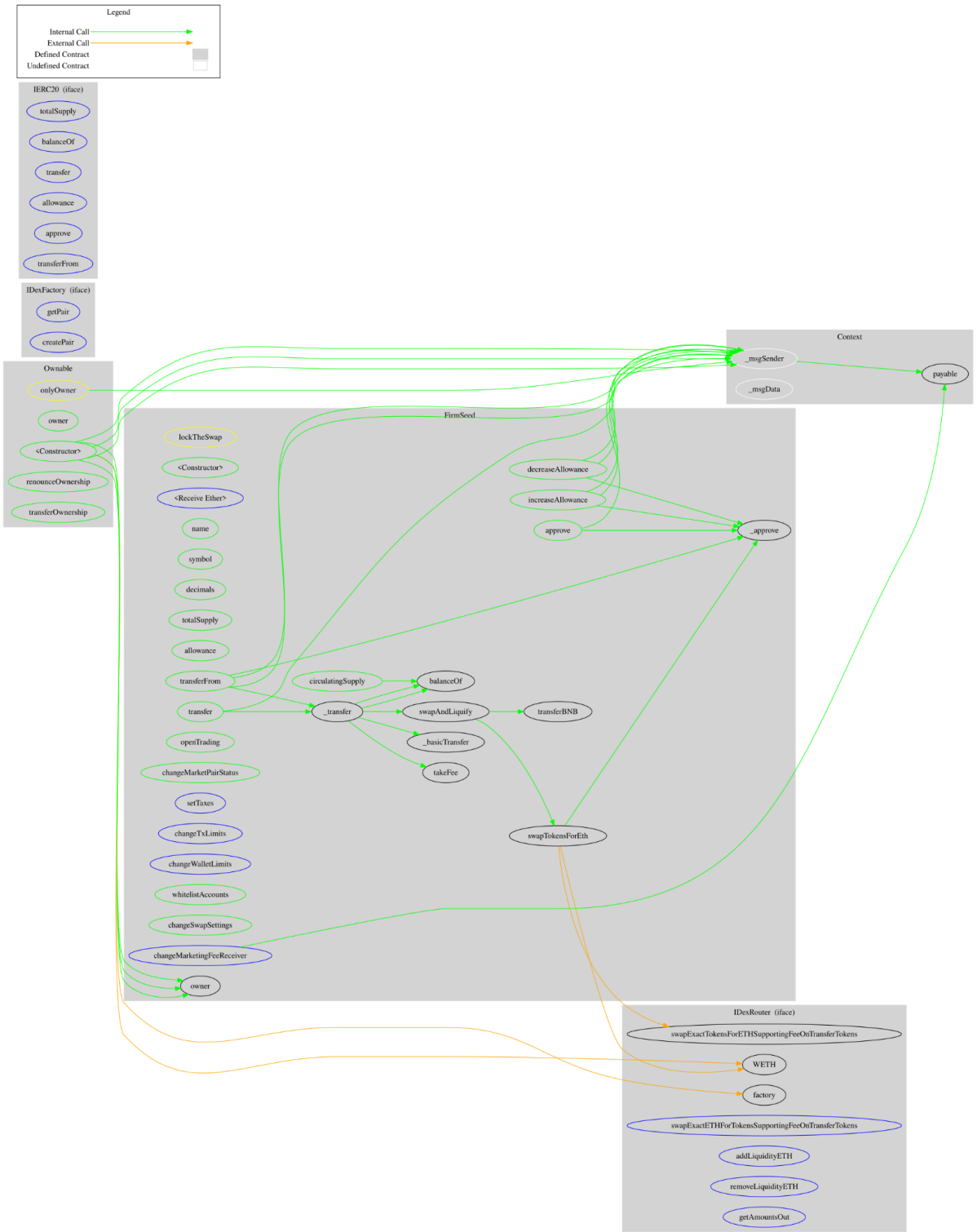
IDexFactory	Interface			
	getPair	External		-
	createPair	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
FirmSeed	Implementation	Context, IERC20, Ownable		
		Public	✓	-
		External	Payable	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	circulatingSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-

	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	approve	Public	✓	-
	_approve	Private	✓	
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	openTrading	Public	✓	onlyOwner
	changeMarketPairStatus	Public	✓	onlyOwner
	setTaxes	External	✓	onlyOwner
	changeTxLimits	External	✓	onlyOwner
	changeWalletLimits	External	✓	onlyOwner
	whitelistAccounts	Public	✓	onlyOwner
	changeSwapSettings	Public	✓	onlyOwner
	changeMarketingFeeReceiver	External	✓	onlyOwner
	transferBNB	Private	✓	
	_transfer	Private	✓	
	_basicTransfer	Internal	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	takeFee	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IDexFactory (iface)
getPair
createPair
Ownable
onlyOwner
owner
<Constructor>
renounceOwnership
transferOwnership
Context
_msgSender
_msgData
payable
FirmSeed
lockTheSwap
<Constructor>
<Receive Ether>
name
symbol
decimals
totalSupply
allowance
transferFrom
transfer
openTrading
changeMarketPairStatus
setTaxes
changeTxLimits
changeWalletLimits
whitelistAccounts
changeSwapSettings
changeMarketingFeeReceiver
owner
decreaseAllowance
increaseAllowance
approve
_approve
circulatingSupply
balanceOf
_transfer
swapAndLiquify
transferBNB
_basicTransfer
takeFee
swapTokensForEth
IDexRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
swapExactETHForTokensSupportingFeeOnTransferTokens
addLiquidityETH
removeLiquidityETH
getAmountsOut

Summary

FirmSeed contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions and manipulate the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io