Cyberscope

Audit Report

Skeleton Key

September 2023

Network BSC

Address 0x329cdae71290117327abf6b4f6abd26025178387

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	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	PVC	Price Volatility Concern	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	MEE	Missing Events Emission	Unresolved
	ULTW	Transfers Liquidity to Team Wallet	Unresolved
	PAV	Pair Address Validation	Unresolved
	FSA	Fixed Swap Address	Unresolved
	RES	Redundant Event Statement	Unresolved
	L02	State Variables could be Declared Constant	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L07	Missing Events Arithmetic	Unresolved
	L13	Divide before Multiply Operation	Unresolved
	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Explorer	https://bscscan.com/address/0x329cdae71290117327abf6b4f6abd26025178387
Symbol	KEY$
Decimals	9
Total Supply	1,000,000,000

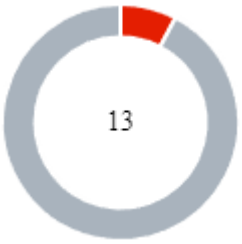
Audit Updates

Initial Audit	16 Sep 2023

Source Files

Filename	SHA256
SkeletonKey.sol	e460f0f849f62c3a47c5270c54c714ed1e72492e96def16a0982b0d7a200891e

Findings Breakdown



Critical	1
Medium	0
Minor / Informative	12

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	12	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	SkeletonKey.sol#L816
Status	Unresolved

Description

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

```
if (!Trade_Open && from != address(this)){
  require(_isWhiteListed[from] || _isWhiteListed[to], "TO2");  // Trade
closed, only whitelisted wallets can move tokens
}
```

Recommendation

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. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	SkeletonKey.sol#L
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTrigger` triggers 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.

```
// Check Transaction Count
if(swapCounter >= swapTrigger){

    // Check Contract Tokens
    uint256 contractTokens = balanceOf(address(this));

    if (contractTokens > 0) {

        // Check if fee is removed during processing
        if (noFeeWhenProcessing && takeFee){takeFee = false;}

        // Limit Swap to Max Transaction
        if (contractTokens <= max_Tran) {

            processFees (contractTokens);

            } else {

            processFees (max_Tran);

        }
    }
}
```

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 total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	SkeletonKey.sol#L372,400,445,471,481,491,501,521,525,603,613,624
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

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.

```
function Process_Add_Liquidity_Pair(

    address Wallet_Address,
    bool true_or_false)

    external onlyOwner {

    _isPair[Wallet_Address] = true_or_false;
    _isLimitExempt[Wallet_Address] = true_or_false;

}
...
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	SkeletonKey.sol#L372,400,413,445,451,471,481,491,501,513,525,566,577,603,624
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function Wallet_Pre_Launch_Access(

    address Wallet_Address,
    bool true_or_false

    ) external onlyOwner {
    _isWhiteListed[Wallet_Address] = true_or_false;
}

...
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	SkeletonKey.sol#L419
Status	Unresolved

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `Process_Trigger_Now` method.

```
function Process_Trigger_Now (uint256 Percent_of_Tokens_to_Process)
external onlyOwner {

    require(!processingFees, "E15"); // Already in swap, try later

    if (Percent_of_Tokens_to_Process >
100){Percent_of_Tokens_to_Process = 100;}
    uint256 tokensOnContract = balanceOf(address(this));
    uint256 sendTokens = tokensOnContract *
Percent_of_Tokens_to_Process / 100;
    processFees(sendTokens);

}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped, since a huge amount may volatile the token's price. 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. Some suggestions are:

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

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

PAV - Pair Address Validation

Criticality	Minor / Informative
Location	SkeletonKey.sol#L372
Status	Unresolved

Description

The contract is missing address validation in the pair address argument. The absence of validation reveals a potential vulnerability, as it lacks proper checks to ensure the integrity and validity of the pair address provided as an argument. The pair address is a parameter used in certain methods of decentralized exchanges for functions like token swaps and liquidity provisions.

The absence of address validation in the pair address argument can introduce security risks and potential attacks. Without proper validation, if the owner's address is compromised, the contract may lead to unexpected behavior like loss of funds.

```
function Process_Add_Liquidity_Pair(
    address Wallet_Address,
    bool true_or_false)
    external onlyOwner {
    _isPair[Wallet_Address] = true_or_false;
    _isLimitExempt[Wallet_Address] = true_or_false;
}
```

Recommendation

To mitigate the risks associated with the absence of address validation in the pair address argument, it is recommended to implement comprehensive address validation mechanisms. A recommended approach could be to verify pair existence in the decentralized application. Prior to interacting with the pair address contract, perform checks to verify the existence and validity of the contract at the provided address. This can be achieved by querying the provider's contract or utilizing external libraries that provide contract verification services.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	SkeletonKey.sol#L308
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
function Launch_03_Launch_Directly_On_PancakeSwap(uint256
percent_of_supply) external payable onlyOwner {
    require(percent_of_supply <= 100, "P100"); // Max permitted is 100
    uint256 tokensIntoLP = _tTotal * percent_of_supply / 100;
    IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E); // BSC
MAIN
    // Create initial liquidity pair with BNB on PancakeSwap factory
    uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
    uniswapV2Router = _uniswapV2Router;
    ...
}
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

RES - Redundant Event Statement

Criticality	Minor / Informative
Location	SkeletonKey.sol#L172,173
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The `TokenCreated` and `LiquidityAdded` event statements are not used in the contract's implementation.

```
event TokenCreated(address indexed Token_CA);
event LiquidityAdded(uint256 Tokens_Amount, uint256 BNB_Amount);
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. It is recommend removing the unused event statement from the contract.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	SkeletonKey.sol#L101,102,103
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
string private  _name = "Skeleton Key"
string private  _symbol = "KEY$"
uint256 private _decimals = 9
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	SkeletonKey.sol#L52,94,95,96,97,115,116,119,120,121,122,125,166,167,168,186,187,188,189,190,202,203,211,251,253,254,255,256,291,293,294,308,342,372,374,375,400,408,413,419,445,451,453,454,471,473,481,483,491,493,501,503,513,515,525,527,566,577,603,605,606,613,615,616,624,626,627,649,661,744,783,897,975
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);
address public _owner = 0x1a6DC28FB7f9CaE01e6ca5733e720d189b7A4103
address public Wallet_Liquidity =
0x74C2025F1DFCfE8A517d3Fef040157C76e33E2B7
address payable public Wallet_Discretion =
payable(0x339DB8A55F0aee632c2049Dd465E09383e6caAf3)
address payable public Wallet_Marketing =
payable(0x3c8B1aa140DA77E50312ab37b51e7567Eb3cc122)
uint256 private max_Hold
uint256 private max_Tran
uint8 public _fee__Marketing  = 6
uint8 public _fee__Liquidity  = 1
uint8 public _fee__Discretion = 1
uint8 public _fee__Reflection = 2
uint8 private _SwapFeeTotal = 8
event updated_Wallet_Limits(uint256 max_Tran, uint256 max_Hold);
event updated_fees(uint8 Marketing, uint8 Liquidity, uint8 Fund, uint8
Reflection);

...
```

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	SkeletonKey.sol#L415
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.

```
swapTrigger = Transaction_Count + 1
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	SkeletonKey.sol#L1007,1008,1024,1025
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
tReflect        = tAmount * _fee__Reflection  / 100
uint256 rReflect        = tReflect      * RFI
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	SkeletonKey.sol#L336,459
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(address(this)).transfer(owner(), balanceOf(address(this)))
IERC20(random_Token_Address).transfer(msg.sender, number_of_Tokens)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

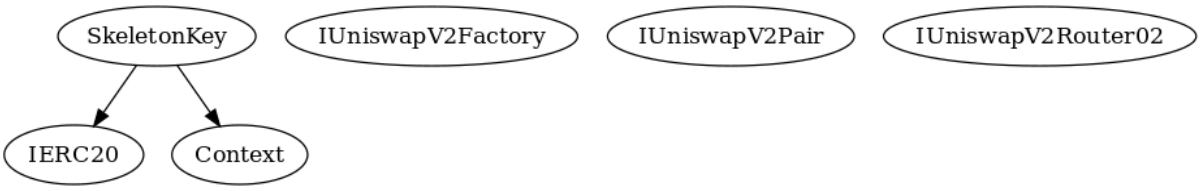
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IUniswapV2Factory	Interface			
	createPair	External	✓	-
	getPair	External		-
IUniswapV2Pair	Interface			
	factory	External		-

IUniswapV2Router02	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
Context	Implementation			
	_msgSender	Internal		
SkeletonKey	Implementation	Context, IERC20		
		Public	✓	-
	Project_Information	External		-
	Launch_01_Set_Fees	External	✓	onlyOwner
	Launch_02_Set_Wallet_Limits	External	✓	onlyOwner
	Launch_03_Launch_Directly_On_PancakeSwap	External	Payable	onlyOwner
	Launch_04_Open_Trade	External	✓	onlyOwner
	Process_Add_Liquidity_Pair	External	✓	onlyOwner
	Process_No_Fee_Wallet_Transfers	External	✓	onlyOwner
	Process_Trigger_Automatic	External	✓	onlyOwner
	Process_Trigger_Count	External	✓	onlyOwner
	Process_Trigger_Now	External	✓	onlyOwner

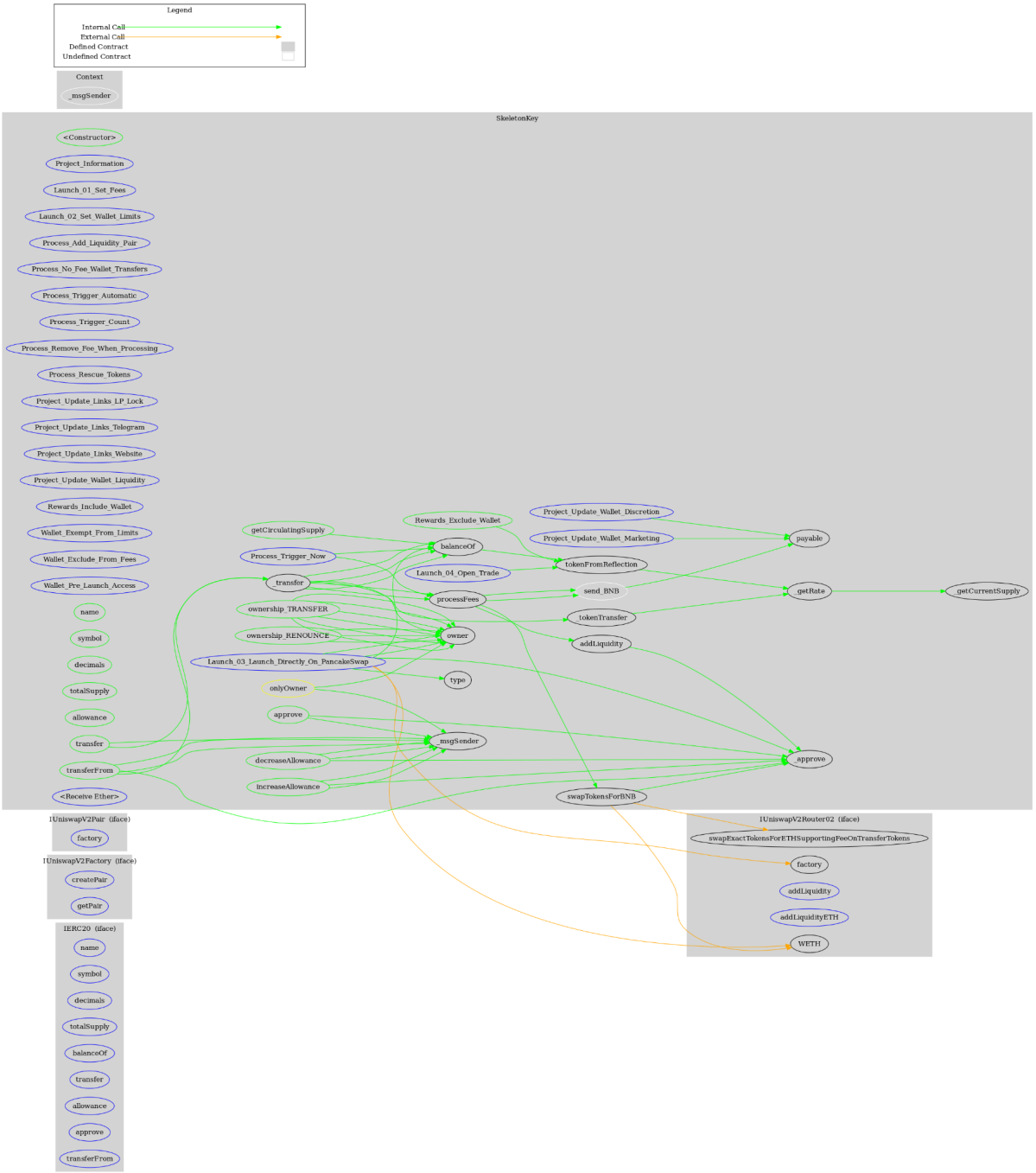
	Process_Remove_Fee_When_Processing	External	✓	onlyOwner
	Process_Rescue_Tokens	External	✓	onlyOwner
	Project_Update_Links_LP_Lock	External	✓	onlyOwner
	Project_Update_Links_Telegram	External	✓	onlyOwner
	Project_Update_Links_Website	External	✓	onlyOwner
	Project_Update_Wallet_Discretion	External	✓	onlyOwner
	Project_Update_Wallet_Liquidity	External	✓	onlyOwner
	Project_Update_Wallet_Marketing	External	✓	onlyOwner
	Rewards_Exclude_Wallet	Public	✓	onlyOwner
	Rewards_Include_Wallet	External	✓	onlyOwner
	Wallet_Exempt_From_Limits	External	✓	onlyOwner
	Wallet_Exclude_From_Fees	External	✓	onlyOwner
	Wallet_Pre_Launch_Access	External	✓	onlyOwner
	ownership_RENOUNCE	Public	✓	onlyOwner
	ownership_TRANSFER	Public	✓	onlyOwner
	owner	Public		-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	increaseAllowance	Public	✓	-

	decreaseAllowance	Public	✓	-
	approve	Public	✓	-
	_approve	Private	✓	
	tokenFromReflection	Internal		
	_getRate	Private		
	_getCurrentSupply	Private		
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	send_BNB	Internal	✓	
	getCirculatingSupply	Public		-
	_transfer	Private	✓	
	processFees	Private	✓	
	swapTokensForBNB	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Context
_msgSender
SkeletonKey
<Constructor>
Project_Information
Launch_01_Set_Fees
Launch_02_Set_Wallet_Limits
Process_Add_Liquidity_Pair
Process_No_Fee_Wallet_Transfers
Process_Trigger_Automatic
Process_Trigger_Count
Process_Remove_Fee_When_Processing
Process_Rescue_Tokens
Project_Update_Links_LP_Lock
Project_Update_Links_Telegram
Project_Update_Links_Website
Project_Update_Wallet_Liquidity
Rewards_Include_Wallet
Wallet_Exempt_From_Limits
Wallet_Exclude_From_Fees
Wallet_Pre_Launch_Access
name
symbol
decimals
totalSupply
allowance
transfer
transferFrom
<Receive Ether>
getCirculatingSupply
Process_Trigger_Now
transfer
ownership_TRANSFER
ownership_RENOUNCE
Launch_03_Launch_Directly_On_PancakeSwap
onlyOwner
approve
decreaseAllowance
increaseAllowance
Rewards_Exclude_Wallet
balanceOf
Launch_04_Open_Trade
processFees
owner
type
_msgSender
Project_Update_Wallet_Discretion
Project_Update_Wallet_Marketing
tokenFromReflection
send_BNB
tokenTransfer
addLiquidity
swapTokensForBNB
payable
getRate
getCurrentSupply
approve
IUniswapV2Pair (iface)
factory
IUniswapV2Factory (iface)
createPair
getPair
IERC20 (iface)
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
factory
addLiquidity
addLiquidityETH
WETH

Summary

Skeleton Key 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. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 10% fees.

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