Cyberscope

Audit Report

Kitten Token

December 2024

Repository : https://github.com/kittentoken/kittentoken/tree/main

Commit : 297405f6e935520a627f6af243259f1fbb048cb9

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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
●	CCR	Contract Centralization Risk	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope’s smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

Review

Repository	https://github.com/kittentoken/kittentoken/tree/main
Commit	297405f6e935520a627f6af243259f1fbb048cb9
Badge Eligibility	Yes

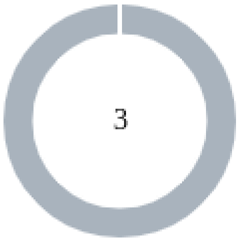
Audit Updates

Initial Audit	15 Nov 2024 https://github.com/cyberscope-io/audits/blob/main/4-kitten/v1/audit.pdf
Corrected Phase 2	06 Dec 2024 https://github.com/cyberscope-io/audits/blob/main/4-kitten/v2/audit.pdf
Corrected Phase 3	27 Dec 2024

Source Files

Filename	SHA256
CoinToken.sol	70fd337503e2de44c433cf63063a6f49ae70dfb0fb6c79bc800ce01dbf469861

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	3

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	3	0	0	0

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	CoinToken.sol#L464,521
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
   function manualSwap() external onlyOwner returns (uint256, uint256,
uint256, uint256) {
        ...
    }

    function setFeeExclusionForAccount(address account, bool boolValue)
external onlyOwner {
       ...
    }

    function setExclusionFromMaxTransaction(address account, bool boolValue)
external onlyOwner {
        ...
    }

    function setSwapTokensAtAmountSupplyPercentage(uint8
newSwapTokensAtAmountSupplyPercentage) external onlyOwner {
       ...
    }

    function setSwapPossibility(bool boolValue) external onlyOwner {
        ...
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	CoinToken.sol#L294
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
uint256 feeAmount =
    (tokenAmount * (liquidityFee + devFee + marketingFee + charityFee) *
multiplier) / (100 * 1000);
uint256 burnAmount = (tokenAmount * burnFee * multiplier) / (100 * 1000);
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	CoinToken.sol#L227
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private
returns (uint256, uint256, uint256) {
(uint256 amountTokenAddedToPool, uint256 amountETHAddedToPool, uint256
amountLiquidityToken) =
router.addLiquidityETH{value: ethAmount}(address(this), tokenAmount, 0, 0,
owner(), block.timestamp);

return (amountTokenAddedToPool, amountETHAddedToPool, amountLiquidityToken);
}
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

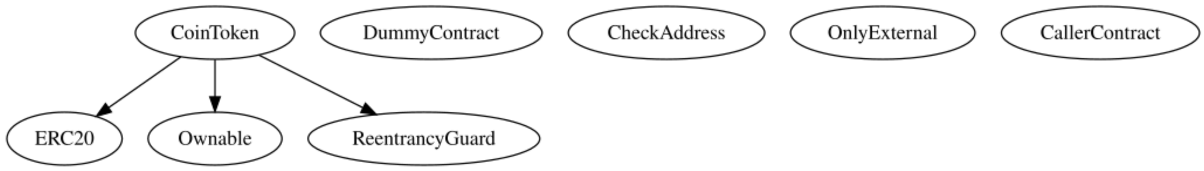
Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

Functions Analysis

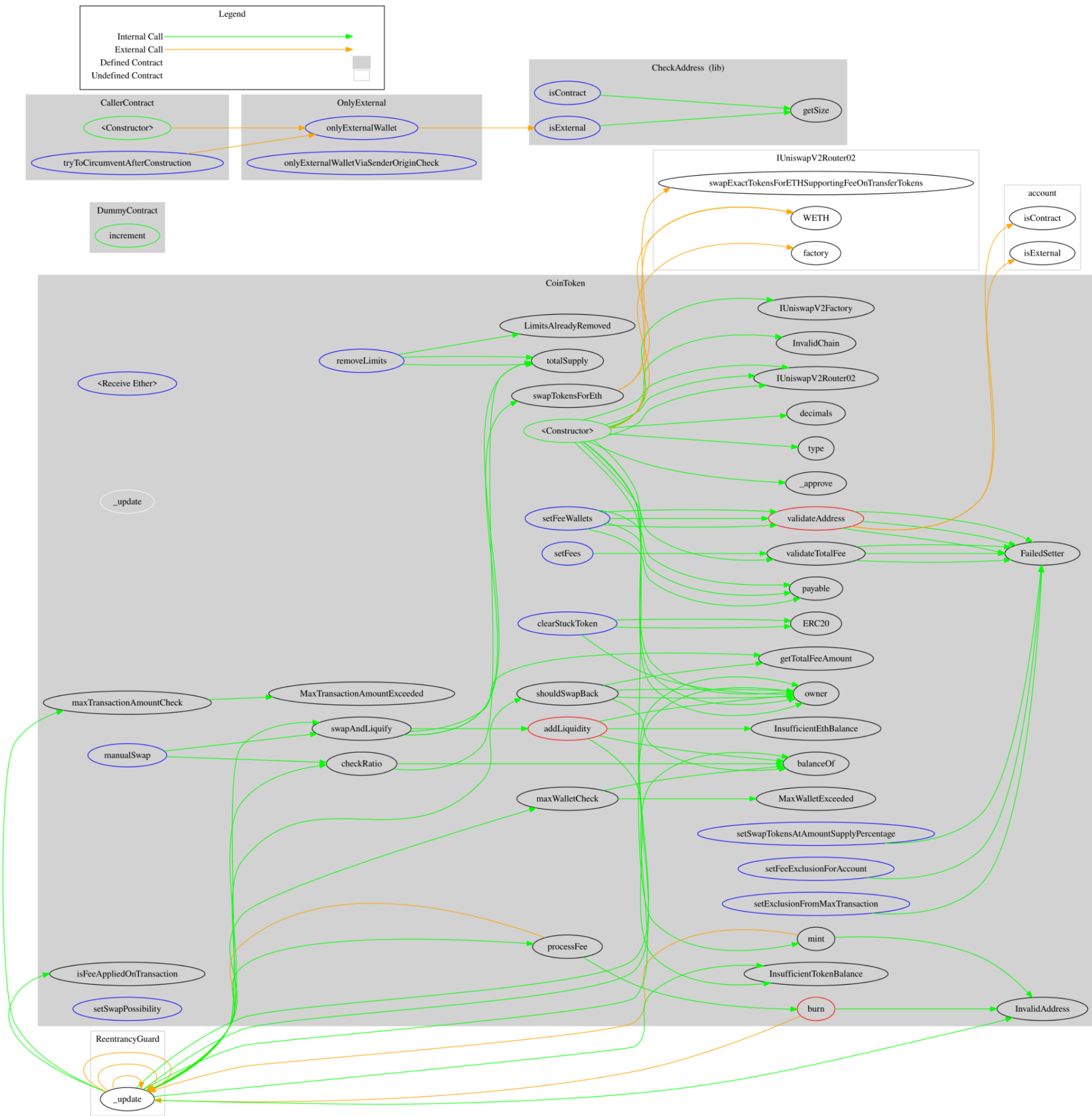
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
CoinToken	Implementation	ERC20, Ownable, ReentrancyGuard		
		Public	✓	ERC20 Ownable
		External	Payable	-
	mint	Private	✓	
	burn	Private	✓	
	_update	Internal	✓	
	addLiquidity	Private	✓	
	removeLimits	External	✓	onlyOwner
	maxWalletCheck	Private		
	maxTransactionAmountCheck	Private		
	isFeeAppliedOnTransaction	Private		
	processFee	Private	✓	
	validateTotalFee	Private		
	validateAddress	Private		
	shouldSwapBack	Private		
	swapAndLiquify	Private	✓	nonReentrant
	checkRatio	Private		
	swapTokensForEth	Private	✓	
	clearStuckToken	External	✓	onlyOwner

	manualSwap	External	✓	onlyOwner
	setFeeWallets	External	✓	onlyOwner
	setFees	External	✓	onlyOwner
	setFeeExclusionForAccount	External	✓	onlyOwner
	setExclusionFromMaxTransaction	External	✓	onlyOwner
	setSwapTokensAtAmountSupplyPercentage	External	✓	onlyOwner
	setSwapPossibility	External	✓	onlyOwner
	getTotalFeeAmount	Private		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
CallerContract
<Constructor>
tryToCircumventAfterConstruction
OnlyExternal
onlyExternalWallet
onlyExternalWalletViaSenderOriginCheck
CheckAddress (lib)
isContract
isExternal
getSize
IUniswapV2Router02
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
account
isContract
isExternal
DummyContract
increment
CoinToken
<Receive Ether>
_update
removeLimits
LimitsAlreadyRemoved
totalSupply
swapTokensForEth
<Constructor>
IUniswapV2Factory
InvalidChain
IUniswapV2Router02
decimals
type
_approve
setFeeWallets
validateAddress
setFees
validateTotalFee
FailedSetter
payable
clearStuckToken
ERC20
getTotalFeeAmount
maxTransactionAmountCheck
MaxTransactionAmountExceeded
shouldSwapBack
owner
swapAndLiquify
addLiquidity
InsufficientEthBalance
manualSwap
checkRatio
balanceOf
maxWalletCheck
MaxWalletExceeded
setSwapTokensAtAmountSupplyPercentage
setFeeExclusionForAccount
setExclusionFromMaxTransaction
mint
processFee
isFeeAppliedOnTransaction
InsufficientTokenBalance
setSwapPossibility
burn
InvalidAddress
ReentrancyGuard
_update

Summary

Kitten Token contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Kitten Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 5% 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

cyberscope.io