Cyberscope

Audit Report

Lucia

January 2025

SHA256 83f19e21a55198ce2f1818d4dedc83a100bfebe4f8c0e60f40be17fe3e83c8a1

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
●	MSC	Missing Sanity Check	Unresolved
●	NCE	Non Compliant ERC20	Unresolved
●	STRC	Self Transfer Restriction Concern	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Analysis 1
Diagnostics 2
Table of Contents 3
Risk Classification 4
Review 5
Audit Updates 5
Source Files 5
Findings Breakdown 6
MSC - Missing Sanity Check 7
Description 7
Recommendation 7
NCE - Non Compliant ERC20 8
Description 8
Recommendation 8
STRC - Self Transfer Restriction Concern 9
Description 9
Recommendation 9
L04 - Conformance to Solidity Naming Conventions 10
Description 10
Recommendation 11
L19 - Stable Compiler Version 12
Description 12
Recommendation 12
Functions Analysis 13
Inheritance Graph 15
Flow Graph 16
Summary 17
Disclaimer 18
About Cyberscope 19

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

Contract Name	Lucia
Testing Deploy	https://testnet.bscscan.com/address/0x84f42099ef5c54817ed1e21c4504d10dcd2e69e5
Symbol	LUCIA
Decimals	18
Total Supply	1,000,000,000
Badge Eligibility	Yes

Audit Updates

Initial Audit	02 Jan 2025

Source Files

Filename	SHA256
contracts/Lucia.sol	83f19e21a55198ce2f1818d4dedc83a100bfebe4f8c0e60f40be17fe3e83c8a1

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	5	0	0	0

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	contracts/Lucia.sol#L71
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The contract does not check if `_dexAddresses[i]` is the zero address, while in the `addDexAddress` and `removeDexAddress` functions the check is included. As a result, the contract could be initialized with the zero address as a dex address.

```
for (uint256 i = 0; i < _dexAddresses.length; i++) {
    dexAddresses[_dexAddresses[i]] = true;
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

NCE - Non Compliant ERC20

Criticality	Minor / Informative
Location	contracts/Lucia.sol#L83
Status	Unresolved

Description

The contract implements an ERC20 token, but it does not fully comply with the ERC20 standard due to the restriction on zero-amount transfers. Specifically, the `_update` function includes the following check:

```
if (_amount == 0) revert ZeroAmount();
```

This restriction contradicts the ERC20 standard, which allows transfers of zero tokens. According to the ERC20 specification, a token contract **must** allow for zero-value transfers.

Recommendation

The team is advised to remove the check to allow for zero-amount transfers, ensuring full compliance with the ERC20 standard.

STRC - Self Transfer Restriction Concern

Criticality	Minor / Informative
Location	contracts/Lucia.sol#L84
Status	Unresolved

Description

The contract explicitly disallows self-transfers by reverting the transaction if the sender (from) and recipient (to) addresses are the same. This design decision can prevent unnecessary gas expenditure on operations that would not alter the token's distribution, thereby avoiding no-op transactions. It may also serve as a safeguard against certain types of exploits or unintended interactions within the contract's logic. However, this restriction could limit the contract's flexibility and interoperability with external contracts, interfaces, or wallets that might rely on detecting Transfer events as part of their logic, even in cases where the token distribution does not change.

```
if (_from == _to) revert SelfTransfer();
```

Recommendation

The team is advised to consider the specific requirements and potential integrations of the contract within its broader ecosystem before deciding to enforce or relax the restriction on self-transfers. If the contract's operational context or security model does not necessitate disallowing self-transfers, allowing them could enhance flexibility and compatibility with external systems that monitor Transfer events for triggers, logging, or other purposes. If self-transfers are to be allowed, it is crucial to ensure that such operations do not introduce vulnerabilities or logic flaws in the contract. Specifically, the contract should handle self-transfers gracefully, ensuring that events are emitted correctly without unintended side effects on the contract's state or token distribution.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/Lucia.sol#L118,130,141,152,163,174,185
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.

```
address _treasuryWallet
uint256 _newBuyTax
uint256 _newSellTax
address _admin
address _dexAddress
```

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/stable/style-guide.html#naming-conventions.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/Lucia.sol#L2
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.13;
```

Recommendation

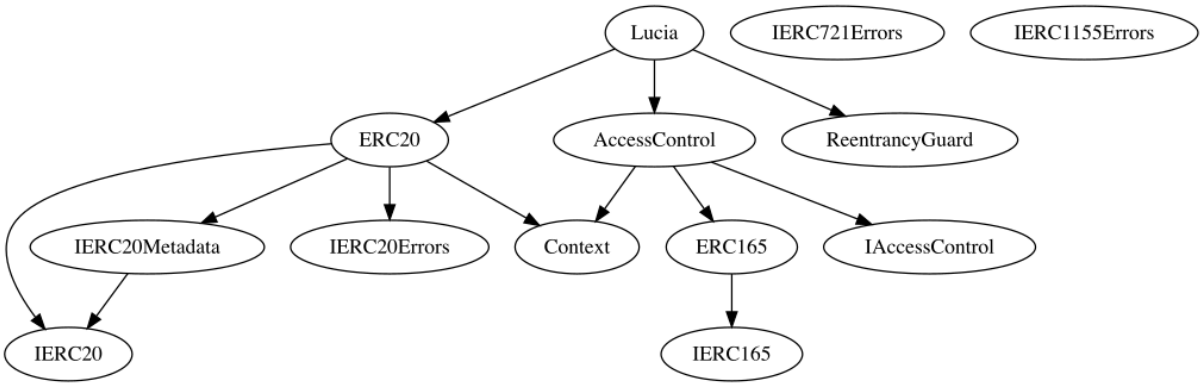
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

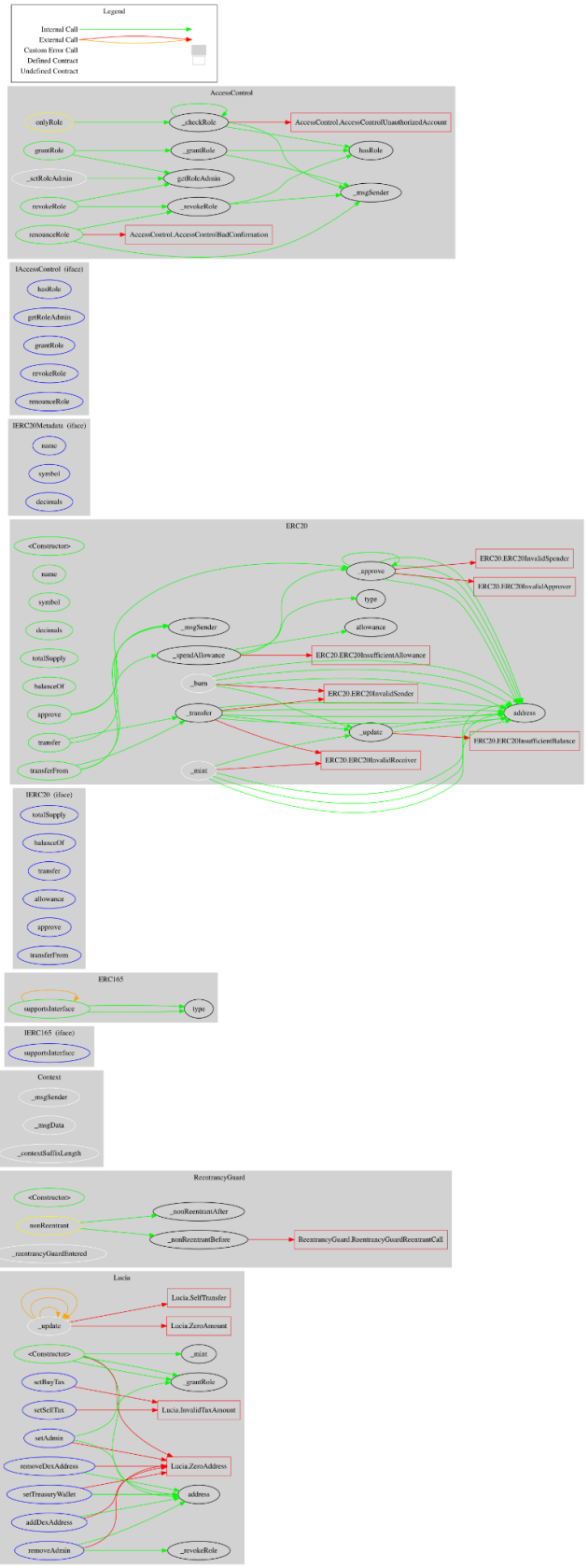
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Lucia	Implementation	ERC20, AccessControl, ReentrancyGuard		
		Public	✓	ERC20
	_update	Internal	✓	nonReentrant
	setTreasuryWallet	External	✓	onlyRole
	setBuyTax	External	✓	onlyRole
	setSellTax	External	✓	onlyRole
	setAdmin	External	✓	onlyRole
	removeAdmin	External	✓	onlyRole
	addDexAddress	External	✓	onlyRole
	removeDexAddress	External	✓	onlyRole
ERC20	Implementation	Context, IERC20, IERC20Metadata, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-

	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
AccessControl
onlyRole
_checkRole
AccessControl.AccessControlUnauthorizedAccount
grantRole
_grantRole
hasRole
_setRoleAdmin
getRoleAdmin
_msgSender
revokeRole
_revokeRole
renounceRole
AccessControl.AccessControlBadConfirmation
IAccessControl (iface)
hasRole
getRoleAdmin
grantRole
revokeRole
renounceRole
IERC20Metadata (iface)
name
symbol
decimals
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transfer
transferFrom
_approve
ERC20.ERC20InvalidSpender
ERC20.ERC20InvalidApprover
type
_msgSender
allowance
_spendAllowance
ERC20.ERC20InsufficientAllowance
_burn
ERC20.ERC20InvalidSender
_transfer
address
_update
ERC20.ERC20InsufficientBalance
ERC20.ERC20InvalidReceiver
_mint
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
ERC165
supportsInterface
type
IERC165 (iface)
supportsInterface
Context
_msgSender
_msgData
_contextSuffixLength
ReentrancyGuard
<Constructor>
nonReentrant
_reentrancyGuardEntered
_nonReentrantAfter
_nonReentrantBefore
ReentrancyGuard.ReentrancyGuardReentrantCall
Lucia
_update
Lucia.SelfTransfer
Lucia.ZeroAmount
<Constructor>
_mint
_grantRole
setBuyTax
setSellTax
Lucia.InvalidTaxAmount
setAdmin
removeDexAddress
Lucia.ZeroAddress
setTreasuryWallet
address
addDexAddress
removeAdmin
_revokeRole

Summary

Lucia contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Lucia is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors 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% buy and sell 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