Cyberscope

Audit Report

MuQuant

June 2023

MUQT	e4f6cf3f2d59023628c53947f2666a9a06db408c30606bd0ebf1b2af133610c3
Launchpad	552f06b794855593666a7b0c2abc7e6170ed82418dfb417c7af938100ddb432c
Router	0a9531b79839f03686552ac7f5e9d47cbbca384dd81304ff1eec237da79d5350
Audited by	© cyberscope

Table of Contents

Review

Audit Updates

Initial Audit	08 Jun 2023

Source Files

Filename	SHA256
interfaces/ICRC20.sol	9e84c467c9645e8e02580b98bd13d2f1bf22f70135e52a323ffb0adef2f06e7d
interfaces/ILaunchpad.sol	0472b8fc71cdcd0c3e62611e6240b05750aa5c5a286a5b0ebf327acbda96d9cf
interfaces/IVault.sol	f2ad0a441b67ec2f02633053ab17a14b6add3d5968bc099c78b98fd129418515
Launchpad.sol	552f06b794855593666a7b0c2abc7e6170ed82418dfb417c7af938100ddb432c
libraries/MuLibrary.sol	5c892a04fe9c458262bc052f303c0dc67c130b79a6557854c3797df7edd5e8c5
MUQT.sol	e4f6cf3f2d59023628c53947f2666a9a06db408c30606bd0ebf1b2af133610c3
Router.sol	0a9531b79839f03686552ac7f5e9d47cbbca384dd81304ff1eec237da79d5350

Overview

The MuQuant ecosystem consists of various contracts. This audit report focuses on the `MUQT` , `Launchpad` and `Router` contracts. The Launchpad represents a launchpad contract that allows the creation and management of token pairs for decentralized exchanges. The launchpad contract has the ability to add new token pairs, set various parameters and limits for each pair, and enable voting for token approval. Additionally, The Router contract acts as a crucial component in facilitating token swaps within a decentralized exchange ecosystem, enforcing access control, managing roles, and handling the various aspects of the swapping process. Lastly, the MUQT contract is a simple ERC20 token with built-in functionality to burn a portion of the transferred amount on each transaction.

MUQT Roles

Owner

The owner has authority over the following functions:

- `function setTransferFee(uint256 _transferFee)`
- `function mint(address to, uint256 amount)`

User

The user can interact with the following functions:

- `transferFrom(address from, address to, uint256 amount)`
- `transfer(address to, uint256 amount)`

Launchpad Roles

Stable

The stable role has authority over the following functions:

- `function addStable(address _stable)`
- `function mint(address to, uint256 amount)`

Approve

The approve role has authority over the following functions:

- `function approve(address _token)`
- `function reject(address _token)`

Router

The router role has authority over the following functions:

- `function setRouter(address _router)`
- `function increaseOwned(address _sender, address _token, uint256 _amount)`
- `function setFrequency(address _sender, address _token)`
- `function sync(address _token)`
- `function reserves(address[2] memory _path, uint256[2] memory _amounts)`

Lock

The lock role has authority over the following functions:

- `function setLock(address _lock)`

TokenCreator

The token creator role has authority over the following functions:

- `function modifySponsored(address _token, address _sponsored)`
- `function modifyEquation(address _token, address _stable, uint256 _base, uint256 _quote)`
- `function addLimit(address _token, uint256 _startTime, uint256 _minVotes, uint256 _limitBySupply, uint256 _limitByBalance, uint256 _taxOverLimit, uint256 _frequency, uint256 _maxByOwner)`
- `function modifyTime(address _token, uint256 _startTime, uint256 _minVotes)`

Token

The token role has authority over the following functions:

- `function modifyFees(address _token, uint256 _fees)`
- `function modifyLimit(address _token, uint256 _limitBySupply, uint256 _limitByBalance, uint256 _taxOverLimit, uint256 _frequency)`
- `function modifyMaxOwn(address _token, uint256 _maxByOwner)`

User

The user can interact with the following functions:

- `function TOKEN_ROLE(address _token)`
- `function addPair(address _token, address _stable, uint256 _base, uint256 _quote, uint256 _fees, uint256[3] memory _rates, address _sponsored)`
- `function vote(address _token)`
- `function enableSwap(address _token)`
- `function getPair(address _token)`
- `function getLimit(address _token)`
- `function getAgent(address _token)`
- `function getReserves(address _token)`
- `function checkOwned(address _sender, address _token, uint256 _amount)`
- `function checkLimitBySupply(address _token, uint256 _amount)`
- `function getTax(address _sender, address _token, uint256 _amountIn, uint256 _amountOut)`
- `function checkFrequency(address _sender, address _token)`
- `function checkStable(address _token)`
- `function getStable(address _token)`
- `function getCreator(address _token)`
- `function getStableList()`
- `function combine(address[3] memory addrs, uint256[3] memory u8, uint256[7] memory u4, uint256[3] memory u5)`

Router Roles

Launchpad

The launchpad role has authority over the following functions:

- `function setLaunchpad(address _launchpad)`

Vault

The vault role has authority over the following functions:

- `function setVault(address _vault)`

Lottery

The lottery role has authority over the following functions:

- `function setLottery(address _lottery)`

Refs

The refs role has authority over the following functions:

- `function setDefaultRef(address _defaultRef)`

User

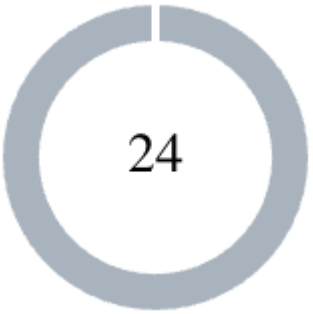
The user can interact with the following functions:

- `function swapExactStableForToken(address ref, uint256 amountIn, uint256 amountOutMin, address token, address to, uint256 deadline)`
- `function swapExactTokenForStable(address ref, uint256 amountIn, uint256 amountOutMin, address token, address to, uint256 deadline)`
- `function swapStableForExactToken(address ref, uint256 amountOut, uint256 amountInMax, address token, address to, uint256 deadline)`
- `function swapTokenForExactStable(address ref, uint256 amountOut, uint256 amountInMax, address token, address to, uint256 deadline)`

Test Deployment

Contract	Explorer
MUQT	https://testnet.bscscan.com/address/0xf42A380492e8555a17390F1991e2E131AD5A9cF4
Launchpad	https://testnet.bscscan.com/address/0xEbaFfA9C04Af3FBBcb19E9B666A920B3B44691C4
Router	https://testnet.bscscan.com/address/0xE8D9EBa7a2F55F75C6F9bb823d025aE3E3eda641

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	24	0	0	0

Severity	Code	Description	Status
	EIS	Excessively Integer Size	Unresolved
	MEE	Missing Events Emission	Unresolved
	FUI	Fee Update Inconsistency	Unresolved
	MMN	Misleading Method Naming	Unresolved
	MSC	Missing Sanity Check	Unresolved
	RSW	Redundant Storage Writes	Unresolved
	DST	Duplicate Stable Tokens	Unresolved
	DKO	Delete Keyword Optimization	Unresolved
	OO	Operator Optimization	Unresolved
	PEO	Pair Existence Optimization	Unresolved
	RVC	Redundant Variable Casting	Unresolved
	MVN	Misleading Variables Naming	Unresolved
	MU	Modifiers Usage	Unresolved
	FCI	Fees Condition Inconsistency	Unresolved
	RVD	Redundant Variable Declaration	Unresolved
	AAO	Accumulated Amount Overflow	Unresolved

●	RC	Redundant Calculations	Unresolved
●	NWA	Non-Guaranteed Withdrawal Amount	Unresolved
●	MT	Mints Tokens	Unresolved
●	FBC	Fees Boundaries Check	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

EIS - Excessively Integer Size

Criticality	Minor / Informative
Location	Launchpad.sol#L218,417
Status	Unresolved

Description

The contract is using a bigger unsigned integer data type that the maximum size that is required. By using an unsigned integer data type larger than necessary, the smart contract consumes more storage space and requires additional computational resources for calculations and operations involving these variables. This can result in higher transaction costs, longer execution times, and potential scalability bottlenecks.

```
uint256(block.timestamp)
```

Recommendation

To address the inefficiency associated with using an oversized unsigned integer data type, it is recommended to accurately determine the required size based on the range of values the variable needs to represent.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	Launchpad.sol#L133,138,148,163,169,223,234,245,259,267,295,314,323, 413Router.sol#L81,87,92,96
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 addStable(address _stable) external onlyRole(STABLE_ROLE)
{
    stable[_stable] = true;
    stableList.push(_stable);
}
function approve(address _token) external onlyRole(APPROVE_ROLE) {
    agent[_token].approval = true;
    ICRC20(_token).setTransferFee(pairs[_token].fees);
    ICRC20(_token).transferOwnership(Router);
    Tokens++;
    if (Tokens == 1) {
        Original = _token;
    }
}
...
```

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.

FUI - Fee Update Inconsistency

Criticality	Minor / Informative
Location	Launchpad.sol#L259
Status	Unresolved

Description

When the approved method is executed, the token fees variable is updated according to the `pairs[_token].fees` value. Hence, when the token is approved the `pairs[token]` fees are synchronized with the token's fees. The `modifyFees()` method updates the `pairs[token]` fees. If the token is approved, it may create inconsistency in the states since the `pairs[token].fees` will be different from the token's fees.

```
function modifyFees(
    address _token,
    uint256 _fees
) public virtual onlyRole(TOKEN_ROLE(_token)) validToken(_token) {
    require(_fees <= DENOMINATOR, "Pair: fees");
    pairs[_token].fees = _fees;
}
...
function approve(address _token) external onlyRole(APPROVE_ROLE) {
    agent[_token].approval = true;
    ICRC20(_token).setTransferFee(pairs[_token].fees);
    ICRC20(_token).transferOwnership(Router);
    Tokens++;
    if (Tokens == 1) {
        Original = _token;
    }
}
```

Recommendation

The team is advised to ensure that the token's fees will always equal to the `pairs[token].fees` variable when the token is approved.

MMN - Misleading Method Naming

Criticality	Minor / Informative
Location	Launchpad.sol#L337
Status	Unresolved

Description

Methods can have misleading names if their names do not accurately reflect the functionality they contain or the purpose they serve. The contract uses some method names that are too generic or do not clearly convey the underneath functionality. Misleading method names can lead to confusion, making the code more difficult to read and understand. Methods can have misleading names if their names do not accurately reflect the functionality they contain or the purpose they serve. The contract uses some method names that are too generic or do not clearly convey the underneath functionality. Misleading method names can lead to confusion, making the code more difficult to read and understand.

The `enableSwap` function name indicates that the function modifies the state of the contract. However, its implementation shows that it only returns if the launchpad is enabled or not.

```
function enableSwap(address _token) public virtual view returns
(bool) {
    return
        agent[_token].approval &&
        block.timestamp >= limits[_token].startTime &&
        agent[_token].votes >= limits[_token].minVotes;
}
```

Recommendation

It's always a good practice for the contract to contain method names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

MSC - Missing Sanity Check

Criticality	Minor / Informative
Location	Launchpad.sol#L138,373
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 the current owned amount plus the `amount` is less than the `limits[_token].maxByOwner` .

```
function increaseOwned(
    address _sender,
    address _token,
    uint256 _amount
) public virtual onlyRole(ROUTER) validToken(_token) {
    owned[_sender][_token] += _amount;
}
```

The contract does not check if the `agent[_token].creator` is the zero address.

```
function approve(address _token) external onlyRole(APPROVE_ROLE) {
    agent[_token].approval = true;
    ICRC20(_token).setTransferFee(pairs[_token].fees);
    ICRC20(_token).transferOwnership(Router);
    Tokens++;
    if (Tokens == 1) {
        Original = _token;
    }
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications. A recommended approach would be the following:

```
require(checkOwned(address _sender, address _token, uint256
_amount), "Amount more than max");
```

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Launchpad.sol#L134,139,164,170Router.sol#L84,88,93,97
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 moodifies the state of certain variable without checking if their current state is equal to the one passed as an argument. As a result, the contract performs redundant storage writes.

```
stable[_stable] = true;
agent[_token].approval = true;
Router = _router;
Lock = _lock;
...
```

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.

DST - Duplicate Stable Tokens

Criticality	Minor / Informative
Location	Launchpad.sol#L135
Status	Unresolved

Description

The `addStable` function sets a token's state as stable and adds it to the `stableList` array. The contract does not check if the token has already been added to the list previously. As a result, the `stableList` may contain duplicate token addresses.

```
function addStable(address _stable) external onlyRole(STABLE_ROLE)
{
    stable[_stable] = true;
    stableList.push(_stable);
}
```

Recommendation

The team is advised to add validation checks to prevent adding a token to the `stableList` more than once. A recommended approach would be to utilize the `stable` variable which indicates if a token is stable or not.

```
function addStable(address _stable) external onlyRole(STABLE_ROLE)
{
    require(!stable[_stable], "Token has already been added");
    stable[_stable] = true;
    stableList.push(_stable);
}
```

DKO - Delete Keyword Optimization

Criticality	Minor / Informative
Location	Launchpad.sol#L149,158,159
Status	Unresolved

Description

The contract resets variables to the default state by setting the initial values. Setting values to state variables increases the gas cost.

```
pairs[_token] = Pair(
    address(0),
    0,
    0,
    0,
    0,
    [uint256(0), 0, 0],
    address(0)
);
agent[_token] = Agent(address(0), 0, false, 0);
limits[_token] = Limit(0, 0, 0, 0, 0, 0, 0);
```

Recommendation

The team is advised to use the `delete` keyword instead of setting variables. This can be more efficient than setting the variable to a new value, using delete can reduce the gas cost associated with storing data on the blockchain.

OO - Operator Optimization

Criticality	Minor / Informative
Location	Launchpad.sol#L187,188,252,253
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 implements certain validation checks before proceeding with its implementation. The `_base` and `_quote` variables are both unsigned integers, meaing their value will always be greater than or equal to zero. Since these variables need to be greater than zero, it would be more optimal to modify the `>` operator to `!=` operator.

```
require(_base > 0, "Pair: base");
require(_quote > 0, "Pair: quote");
```

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.

PEO - Pair Existence Optimization

Criticality	Minor / Informative
Location	Launchpad.sol#L199
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 performs certain validations to check if a pair exists or not. This check could be optimized to a single validation. As a result, the contract consumes more gas to execute these operations.

```
require(
    pairs[_token].stable == address(0) &&
        pairs[_token].liquidity == 0 &&
        pairs[_token].base == 0 &&
        pairs[_token].quote == 0 &&
        pairs[_token].fees == 0,
    "Pair: exists"
);
```

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.

Validating only the `pairs[_token].stable == address(0)` statement would be enough.

```
require(pairs[_token].stable == address(0), "Pair: exists");
```

RVC - Redundant Variable Casting

Criticality	Minor / Informative
Location	Launchpad.sol#L218,417
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 casts the `block.timestamp` variable to a 256-bit unsigned integer. The `block.timestamp` variable is a special variable which always exist in the global namespace and returns the current block timestamp as seconds since unix epoch. The type of the variable is 256-bit unsigned integer. As a result, the contract performs redundant variable casting.

```
uint256(block.timestamp)
```

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.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	Launchpad.sol#L471,472,473
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

```
uint256[3] memory u8
uint256[7] memory u4
uint256[3] memory u5
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

MU - Modifiers Usage

Criticality	Minor / Informative
Location	Launchpad.sol#L186,187,188,251,252,253,283
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

Additionally, the `require(_token != address(0) && _token != address(this), "Pair: token");` operation already exists in the `validToken` modifier.

```
require(stable[_stable], "Pair: stable");
require(_base > 0, "Pair: base");
require(_quote > 0, "Pair: quote");
...
require(_token != address(0) && _token != address(this), "Pair:
token");
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

FCI - Fees Condition Inconsistency

Criticality	Minor / Informative
Location	Launchpad.sol#L189,263
Status	Unresolved

Description

The contract demonstrates an inconsistency between the validation checks regarding the `_fees` variable. The `addPair` and `modifyFees` function which include these checks do not execute the same validation check. As a result, the contract may assign a value to the `_fees` that it shoundn't.

```
require(_fees < DENOMINATOR, "Pair: fees");
require(_fees <= DENOMINATOR, "Pair: fees");
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the `_fees` validation is consistent across the contract.

RVD - Redundant Variable Declaration

Criticality	Minor / Informative
Location	Router.sol#L47
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 stores the address of the `Vault` contract in a separate variable. Getting the address of a contract can be done by using the `address` solidity function. As a result, the variable's declaration is redundant.

```
address public Vault;
Vault = _vault;
```

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.

A recommended approach would be the following:

```
address(vault)
```

AAO - Accumulated Amount Overflow

Criticality	Minor / Informative
Location	Router.sol#L114,116,126,128
Status	Unresolved

Description

The contract is using variables to accumulate values. The contract could lead to an overflow when the total value of a variable exceeds the maximum value that can be stored in that variable's data type. This can happen when an accumulated value is updated repeatedly over time, and the value grows beyond the maximum value that can be represented by the data type.

```
volumes += amounts[0];
muVol += amounts[0];
volumes += amounts[1];
muVol += amounts[1];
```

Recommendation

The team is advised to carefully investigate the usage of the variables that accumulate value. A suggestion is to add checks to the code to ensure that the value of a variable does not exceed the maximum value that can be stored in its data type.

RC - Redundant Calculations

Criticality	Minor / Informative
Location	Router.sol#L230,251,292,296,314,349,353,370,409,413,431
Status	Unresolved

Description

Redundant calculations can occur in various forms in a smart contract, such as repetitive calculations for the same result, the recalculation of unchanging values, and the repeated execution of complex computations. These redundant calculations can significantly increase the gas cost of executing the smart contract. The contract executes redundant calculations in the segments listed below.

The operation `(amountIn * (10000 - pair.fees)) / 10000` is calculated more than once.

```
(amountIn * (10000 - pair.fees)) / 10000
```

Recommendation

The team is advised to take into consideration these segments 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.

NWA - Non-Guaranteed Withdrawal Amount

Criticality	Minor / Informative
Location	Router.sol#L125
Status	Unresolved

Description

As part of the swap functionality, the contract deposits the `amountIn` to the vault if the token is stable or withdraws the `amountOut` from the vault and transfers it to the recipient. Additionally, a token can be set as stable by the `STABLE_ROLE`. Hence, initially all tokens are not stable and when the `_swap` function is executed, it will try to deposit the `amountOut` to the recipient. But since the token was never stable, no amount was deposited to the vault previously. As a result, the transaction will revert.

```
vault.withdraw(path[0], to, amounts[1]);
```

Recommendation

The owner is responsible for the optimal configuration of the tokens. 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.

MT - Mints Tokens

Criticality	Minor / Informative
Location	MUQT.sol#L19
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
function mint(address to, uint256 amount) public onlyOwner {
    _mint(to, amount);
}
```

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 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.

FBC - Fees Boundaries Check

Criticality	Minor / Informative
Location	MUQT.sol#L15
Status	Unresolved

Description

The contract allows setting any value as `_transferFee`. This variable is used in the expression `(amount * transferFee) / 10 ** 4` in order to calculate the burn amount. If the burned amount is more than the transferred amount, then the transaction may fail because the recipient's balance will not be sufficient.

```
function setTransferFee(uint256 _transferFee) public onlyOwner {
    transferFee = _transferFee;
}
```

Recommendation

The team is advised to add a fee limit check in the `setTransferFee()` method. The limit should be at least `10 ** 4` but it could also be decreased to a reasonable amount, so it will not be able to burn the entire recipient's balance.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Router.sol#L47,82,87,92,96MUQT.sol#L15Launchpad.sol#L84,85,86,87,129,133,138,148,163,169,174,175,176,177,178,179,180,224,225,235,236,246,247,248,249,260,261,268,271,272,273,274,296,297,298,299,300,315,316,317,324,325,330,337,344,348,352,357,363,364,365,374,375,376,382,383,392,393,394,395,406,407,414,415,420,424,428,441,442,457interfaces/ILaunchpad.sol#L92,93
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 public Vault
address _launchpad
address _vault
address _lottery
address _defaultRef
uint256 _transferFee
address public Lock
address public Router
uint256 public Tokens
address public Original

function TOKEN_ROLE(address _token) public pure returns (bytes32) {
        return keccak256(abi.encodePacked(_token));
    }
address _token

...
```

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	MUQT.sol#L16
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.

```
transferFee = _transferFee
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	Router.sol#L88,93,97Launchpad.sol#L144,164,170
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
Vault = _vault
lottery = _lottery
defaultRef = _defaultRef
Original = _token
Router = _router
Lock = _lock
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Router.sol#L2MUQT.sol#L2Launchpad.sol#L2interfaces/ILaunchpad.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 ^0.8.18;
pragma solidity ^0.8.9;
```

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**
ILaunchpad	Interface			
	enableSwap	External		-
	getPair	External		-
	getLimit	External		-
	getAgent	External		-
	getReserves	External		-
	checkOwned	External		-
	increaseOwned	External	✓	-
	checkLimitBySupply	External		-
	getTax	External		-
	checkFrequency	External		-
	setFrequency	External	✓	-
	checkStable	External		-
	sync	External	✓	-
	reserves	External	✓	-
	getCreator	External		-
	getStable	External		-
	getStableList	External		-
	Tokens	External		-

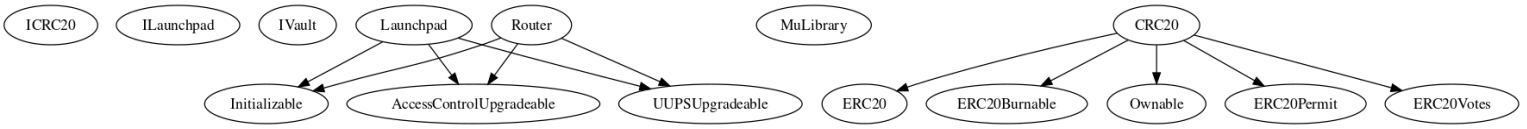
	Original	External		-
Launchpad	Implementation	Initializable, AccessControlUpgradeable, UUPSUpgradeable		
		Public	✓	-
	initialize	Public	✓	initializer
	_authorizeUpgrade	Internal	✓	onlyRole
	TOKEN_ROLE	Public		-
	addStable	External	✓	onlyRole
	approve	External	✓	onlyRole
	reject	External	✓	onlyRole
	setRouter	External	✓	onlyRole
	setLock	External	✓	onlyRole
	addPair	Public	✓	beforeApprove validToken
	modifySponsored	Public	✓	onlyCreator validToken
	modifyRates	Public	✓	onlyCreator validToken
	modifyEquation	Public	✓	onlyCreator beforeApprove validToken
	modifyFees	Public	✓	onlyRole validToken
	addLimit	Public	✓	onlyCreator beforeApprove validLimit
	modifyLimit	Public	✓	onlyRole validLimit validToken

	modifyTime	Public	✓	onlyCreator beforeApprove validToken
	modifyMaxOwn	Public	✓	onlyRole validToken
	vote	Public	✓	validToken
	enableSwap	Public		-
	getPair	Public		-
	getLimit	Public		-
	getAgent	Public		-
	getReserves	Public		-
	checkOwned	Public		-
	increaseOwned	Public	✓	onlyRole validToken
	checkLimitBySupply	Public		-
	getTax	Public		-
	checkFrequency	Public		-
	setFrequency	Public	✓	onlyRole validToken
	checkStable	Public		-
	getStable	Public		-
	sync	Public	✓	onlyRole validToken
	reserves	Public	✓	onlyRole
	getCreator	Public		-
	getStableList	Public		-
	combine	Public	✓	-

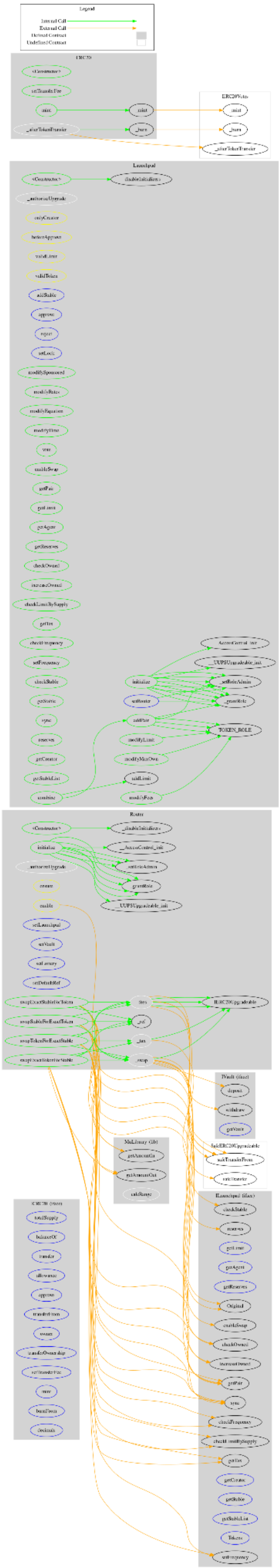
CRC20	Implementation	ERC20, ERC20Burnable, Ownable, ERC20Permit, ERC20Votes		
		Public	✓	ERC20 ERC20Permit
	setTransferFee	Public	✓	onlyOwner
	mint	Public	✓	onlyOwner
	_afterTokenTransfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
Router	Implementation	Initializable, AccessControlUpgradeable, UUPSUpgradeable		
		Public	✓	-
	initialize	Public	✓	initializer
	_authorizeUpgrade	Internal	✓	onlyRole
	setLaunchpad	External	✓	onlyRole
	setVault	External	✓	onlyRole
	setLottery	External	✓	onlyRole
	setDefaultRef	External	✓	onlyRole
	_swap	Internal	✓	
	_fees	Internal	✓	
	_tax	Internal	✓	

	_ref	Internal	✓	
	swapExactStableForToken	Public	✓	ensure enable
	swapExactTokenForStable	Public	✓	ensure enable
	swapStableForExactToken	Public	✓	ensure enable
	swapTokenForExactStable	Public	✓	ensure enable

Inheritance Graph



Flow Graph



Summary

MuQuant contract implements a utility and financial mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

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