Cyberscope

Audit Report

TypeAI

March 2024

Commit 05c878687ca655363cd8cd0e4f255227a24706da

Audited by © cyberscope

Table of Contents

Review

Audit Updates

Initial Audit	20 Mar 2024
Corrected Phase 2	27 Mar 2024

Source Files

Filename	SHA256
TypeAI.sol	c05ff484a6c4f3558d4c2c951b2a8698688b0fa568f770d1b434dbf6c21ae73d
Pool.sol	4e2df8afc1bdaf132442b6a3cb74eec96eb5ce0c483515c5189c26cfd2bafdc7

Test Deployment

Contract	Explorer
TypeAI	https://testnet.bscscan.com/address/0xf89aa14900b15d726950ad45fc55057188c250a8
Pool	https://testnet.bscscan.com/address/0x599098a63fbcf1b4f6b18d5c1226ca8571a5da0d

Overview

TypeAI is a smart contract implementing a locking staking mechanism with rewards distributed in tokens. Additionally, it offers the option for stakers to receive rewards in ETH if they choose to relock their stake. The contract is designed to provide staking rewards based on a fixed Annual Percentage Rate (APR) and a specified lock-in period.

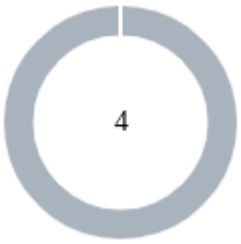
Roles

Owner

The contract owner who is set at contract's constructor can interact with the following functions:

- function updateAPR(uint8 newAPR)
- function updateLockInPeriod(uint256 newLockInPeriod)
- function withdrawResidualBalance()

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	0	3	0	1

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CCR	Contract Centralization Risk	Acknowledged
●	MSC	Missing Sanity Checks	SemiResolved
●	TSI	Tokens Sufficiency Insurance	Acknowledged
●	L09	Dead Code Elimination	Acknowledged

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	TypeAI.sol#L166,178,256
Status	Acknowledged

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 updateAPR(uint8 _newAPR) public onlyOwner {
    /// Change the APR
    uint256 _oldAPR = fixedAPR;
    fixedAPR = _newAPR;
    /// Emitting `APRUpdated` event.
    emit APRUpdated(_oldAPR, _newAPR);
}
...
function updateLockInPeriod(uint256 _newLockInPeriod) public onlyOwner {
    /// Change the Lock in period.
    uint256 _oldLockInPeriod = lockInPeriod;
    lockInPeriod = _newLockInPeriod;
    /// Emitting `LockInPeriodUpdated` event.
    emit LockInPeriodUpdated(_oldLockInPeriod, _newLockInPeriod);
}
...
function withdrawResidualBalance() public onlyOwner nonReentrant {
    uint256 residualBalance = token.balanceOf(address(this)) -
        _totalValueLocked;
    if (residualBalance == 0) revert
TypeAI__InsufficientResidualBalance();
    /// Transfer the tokens.
    token.safeTransfer(_msgSender(), residualBalance);
}
```

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.

MSC - Missing Sanity Checks

Criticality	Minor / Informative
Location	TypeAI.sol#L57,166,178
Status	SemiResolved

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.

It was observed that the protocol values are initialized without any validation or constraint checks. Within the contract, function parameters are being used directly without ensuring that they fall within reasonable limits.

The lack of constraints on these parameters poses a potential risk of extreme values being set, which could lead to unintended consequences such as excessive rewards for participants or excessively long lock-in periods.

```
constructor(
    address _token,
    uint256 _fixedAPR,
    uint256 _lockInPeriod,
    address router_
) Ownable(_msgSender()) {
    /// Updating the state.
    token = IERC20(_token);
    fixedAPR = _fixedAPR;
    lockInPeriod = _lockInPeriod;
    _router = IUniswapV2Router(router_);

    /// Emitting Events.
    emit APRUpdated(0, _fixedAPR);
    emit LockInPeriodUpdated(0, _lockInPeriod);
}
...
function updateAPR(uint8 _newAPR) public onlyOwner {
    /// Change the APR
    uint256 _oldAPR = fixedAPR;
    fixedAPR = _newAPR;
    /// Emitting `APRUpdated` event.
    emit APRUpdated(_oldAPR, _newAPR);
}
...
function updateLockInPeriod(uint256 _newLockInPeriod) public onlyOwner {
    /// Change the Lock in period.
    uint256 _oldLockInPeriod = lockInPeriod;
    lockInPeriod = _newLockInPeriod;
    /// Emitting `LockInPeriodUpdated` event.
    emit LockInPeriodUpdated(_oldLockInPeriod, _newLockInPeriod);
}
```

Recommendation

The team is advised to implement robust parameter validation mechanisms within the constructor to ensure that the values provided are within reasonable bounds. This can be achieved by setting appropriate upper and lower limits for these parameters and validating them before updating the state variables.

Suggested values to check would be:

- `fixedAPR` is greater than zero
- `lockInPeriod` less than a year (or similar)

- `_router` and `token` are not the zero address

Team Update

The team adjusted the checking of the values `fixAPR` , `_router` , `token` and acknowledged the `lockInPeriod` .

TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Location	TypeAI.sol#L482
Status	Acknowledged

Description

The tokens are not held within the contract itself. Instead, the contract is designed to provide the tokens from an external administrator. While external administration can provide flexibility, it introduces a dependency on the administrator's actions, which can lead to various issues and centralization risks.

```
function _claimGainedInterest(address _stakeHolder) private {
    /// Getting the stake holder details as storage.
    StakeHolder storage _holderDetails = _stakeHolderDetailsOf[
        _stakeHolder
    ];
    /// @dev Updating the claimable interest.
    _updateClaimableInterestOf(_stakeHolder);

    /// @dev Checking if any claimable amount available.
    uint256 _claimableInterest = _holderDetails.claimableInterest;
    if (_claimableInterest == 0)
        revert TypeAI__NoInterestGained(_stakeHolder);
    if (
        (token.balanceOf(address(this)) - _claimableInterest) <
        _totalValueLocked
    )
        revert TypeAI__InsufficientRewardPresent(
            _stakeHolder,
            _claimableInterest
        );

    /// Removing the claimable amount.
    delete _holderDetails.claimableInterest;

    /// @dev Transfer the interest and emitting the event.
    token.safeTransfer(_stakeHolder, _claimableInterest);
    /// Emitting `InterestClaimed` event.
    emit InterestClaimed(_stakeHolder, _claimableInterest);
}
```

Recommendation

It is recommended to consider implementing a more decentralized and automated approach for handling the contract tokens. One possible solution is to hold the presale tokens within the contract itself. If the contract guarantees the process it can enhance its reliability, security, and participant trust, ultimately leading to a more successful and efficient process.

Team Update

The team commented `Reward token is coming from external source.`

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Pool.sol#L8,9
Status	Acknowledged

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _contextSuffixLength() internal view virtual returns (uint256)
{
    return 0;
}

function _burn(address account, uint256 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidSender(address(0));
    }
    _update(account, address(0), value);
}
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Pool	Implementation	ERC20, Ownable		
		Public	✓	ERC20 Ownable
TypeAI	Implementation	ITypeAI, Ownable, ReentrancyGuard		
		Public	✓	Ownable
	noOfStakeHolders	Public		-
	stakeHolderDetailsOf	Public		-
	totalAmountStakedBy	Public		-
	totalValueLocked	Public		-
	claimableInterestGainedBy	Public		-
	getRealizedETH	Public		-
	getStakeHolders	Public		-
	updateAPR	Public	✓	onlyOwner
	updateLockInPeriod	Public	✓	onlyOwner
	stake	Public	✓	nonReentrant
	unstake	Public	✓	nonReentrant
	withdrawResidualBalance	Public	✓	onlyOwner nonReentrant
	claimGainedInterest	Public	✓	nonReentrant
	claimETHAndReLock	Public	✓	nonReentrant

	compoundETHAndReLock	Public	✓	nonReentrant
	depositETHRewards	Public	Payable	nonReentrant
	_cumulativeETHRewards	Private		
	_depositETHRewards	Private	✓	
	_compoundETHRewards	Private	✓	
	_distributeETHRewards	Private	✓	
	_stake	Private	✓	
	_calculateInterestGainedBy	Private		
	_claimGainedInterest	Private	✓	
	_updateClaimableInterestOf	Private	✓	
	_updateUnrealizedETHRewardsOf	Private	✓	

Inheritance Graph

See the detailed images in the github repository.

Flow Graph

See the detailed images in the github repository.

Summary

Type AI contracts implement a token contract and a locking token staking contract. 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