Cyberscope

Audit Report

Tea-Fi presaleClaim

August 2024

SHA256 ecffce9791bc4af9e77144b356643df8741f1077755b5c1fba84a69bea5ba236

Audited by © cyberscope

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

Testing Deploy	https://testnet.bscscan.com/address/0xeA89d0f265bD69CE82934C09A7aCdfF0A6710385

Audit Updates

Initial Audit	13 Aug 2024 https://github.com/cyberscope-io/audits/blob/main/tea-fi/v1/presaleClaim.pdf
Corrected Phase 2	21 Aug 2024

Source Files

Filename	SHA256
ClaimEarningFees.sol	ecffce9791bc4af9e77144b356643df8741f1077755b5c1fba84a69bea5ba236

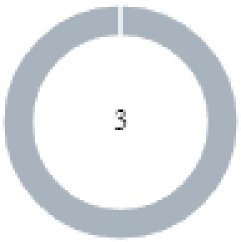
Overview

The `ClaimEarningFees` smart contract is designed to manage the distribution of funds to users based on a Merkle tree proof mechanism. The contract utilizes a role-based access control system, where specific roles such as DEFAULT_ADMIN_ROLE and OPERATOR_ROLE have distinct permissions to manage the contract's functionalities. The primary function of the contract is to allow eligible users to claim tokens or Ether based on a provided Merkle proof, which validates their entitlement to the specified amounts.

The contract includes mechanisms to pause and unpause the claiming process, either globally or for individual accounts. This ensures that the contract can be securely managed during its operation, allowing the administrator or operators to control the flow of claims. The unpause function also resets the claim period, establishing a new deadline for claims based on the current block timestamp.

Additionally, the contract provides functionality for administrators to withdraw Ether or ERC20 tokens held within the contract, directing the funds to a designated multisig wallet. This ensures that the contract can manage and distribute funds securely and efficiently. The contract also includes standard safeguards such as reentrancy protection and the ability to check if accounts are paused or have already claimed their entitled funds.

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	0	3	0	0

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CCR	Contract Centralization Risk	Acknowledged
●	MPC	Merkle Proof Centralization	Acknowledged
●	TSI	Tokens Sufficiency Insurance	Acknowledged

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	ClaimEarningFees.sol#L217,244,281
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. Specifically, the `pause` and `unpause` functions grant administrative roles the ability to pause or unpause the claims globally or for specific addresses. Furthermore, the `unpause` function also enables to modify the Merkle root used for claims. Additionally, the admin role has the authority to withdraw the contract's balance of both native currency (ETH) and ERC20 tokens.

```
function pause(address account) external onlyAdminAndOperatorRoles
{
    if (account == _msgSender()) {
        revert PauseYourselfError();
    } else if (account == address(0)) {
        _pause();
        emit ClaimPaused();
        return;
    }

    AccountData storage _accountData = accountData[account];

    if (_accountData.isPaused) {
        revert AccountHasAlreadyBeenPausedError();
    }

    accountData[account].isPaused = true;
    emit AccountIsPausedFromClaim(account);
}

function unpause(
    bytes32 newRoot,
    address account,
    uint256 customPeriod
) external onlyAdminAndOperatorRoles {
    if (account == address(0)) {
        uint256 durationNormal = customPeriod != 0
            ? customPeriod
            : CLAIM_PERIOD_DURATION;
        unchecked {
            claimPeriodDeadline = block.timestamp + durationNormal;
        }

        if (paused()) {
            _unpause();
        }

        claimRoot = newRoot;

        emit ClaimUnpaused();
        return;
    }

    AccountData storage _accountData = accountData[account];

    if (!_accountData.isPaused) {
        revert AccountHasNotBeenPausedError();
    }

    _accountData.isPaused = false;
    emit AccountIsUnpausedFromClaim(account);
}
```

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.

Team Update

The team has acknowledged that this is not a security issue and states: *The admin and operators are EOAs that are owned by project owners, and their private keys are well protected. The purpose of these roles is to enhance the security of the contract by making hand-on validations of each claim (determined by the algorithm that parses the blockchain events of the Presale smart contract) and submitting the root hash in accordance with the claim. We have an admin panel where all information related to claims is displayed and validated. By using such logic, the backend does not need to store any private keys. Also, the tokens will be stored in a contract only while the claim is active, and since each claim has a deadline, these funds can be withdrawn to the multisig wallet after each claim. As a result, the assets are provided with an additional level of security, preventing them from remaining in the contract for a long period of time or accumulating large amounts of money.*

MPC - Merkle Proof Centralization

Criticality	Minor / Informative
Location	ClaimEarningFees.sol#L172,244
Status	Acknowledged

Description

The contract uses a Merkle Proof mechanism in order to define many applicable addresses. The verification process is based on an off-chain configuration. The contract owner is responsible for updating the in-chain “Merkle Root” in order to validate correctly the provided message.

```
MerkleProof.verify(
    proof,
    claimRoot,
    keccak256(abi.encodePacked(account, nonce, tokens, amounts))
);

function unpause(
    bytes32 newRoot,
    address account,
    uint256 customPeriod
) external onlyAdminAndOperatorRoles {
    if (account == address(0)) {
        uint256 durationNormal = customPeriod != 0
            ? customPeriod
            : CLAIM_PERIOD_DURATION;
        unchecked {
            claimPeriodDeadline = block.timestamp + durationNormal;
        }

        if (paused()) {
            _unpause();
        }

        claimRoot = newRoot;

        emit ClaimUnpaused();
        return;
    }

...
```

Recommendation

We state that the Merkle Proof algorithm is required for proper protocol operations and gas consumption decrease. Thus, we emphasize that the Merkle proof algorithm is based on an off-chain mechanism. Any off-chain mechanism could potentially be compromised and affect the on-chain state unexpectedly. 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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

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

Permanent Solution:

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

Team Update

The team has acknowledged that this is not a security issue and states: *The admin and operators are EOAs that are owned by project owners, and their private keys are well protected. The purpose of these roles is to enhance the security of the contract by making hand-on validations of each claim (determined by the algorithm that parses the blockchain events of the Presale smart contract) and submitting the root hash in accordance with the claim. We have an admin panel where all information related to claims is displayed and validated. By using such logic, the backend does not need to store any private keys. Also, the tokens will be stored in a contract only while the claim is active, and since each claim has a deadline, these funds can be withdrawn to the multisig wallet after each claim. As a result, the assets are provided with an additional level of security, preventing them from remaining in the contract for a long period of time or accumulating large amounts of money.*

TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Location	ClaimEarningFees.sol#L135
Status	Acknowledged

Description

The tokens and native currency are not held within the contract itself. Instead, the contract is designed to provide them 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.

```
if (tokens[i] != address(0)) {
    IERC20(tokens[i]).safeTransfer(sender, amounts[i]);
} else {
    _transferEther(sender, amounts[i]);
}
```

Recommendation

It is recommended to consider implementing a more decentralized and automated approach for handling the contract tokens and native currency. One possible solution is to hold them 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.

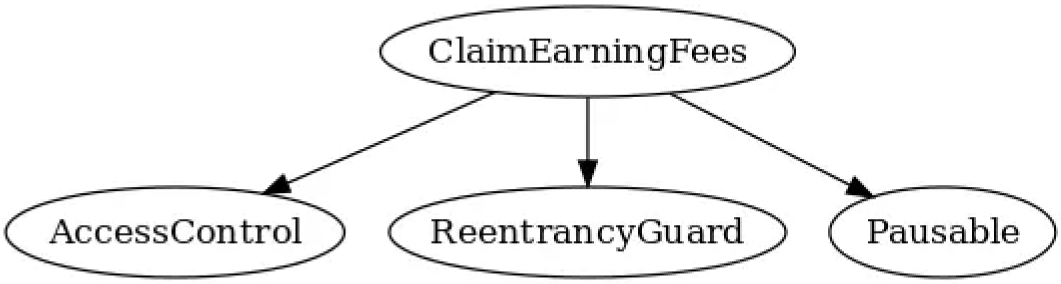
Team Update

The team has acknowledged that this is not a security issue and states: *The contract logic is built in such a way that tokens must be on the contract balance (the admin will send them to the contract before submitting the root hash). Multisig wallet is needed only to withdraw the remaining tokens there after the claim (only the admin can do this).*

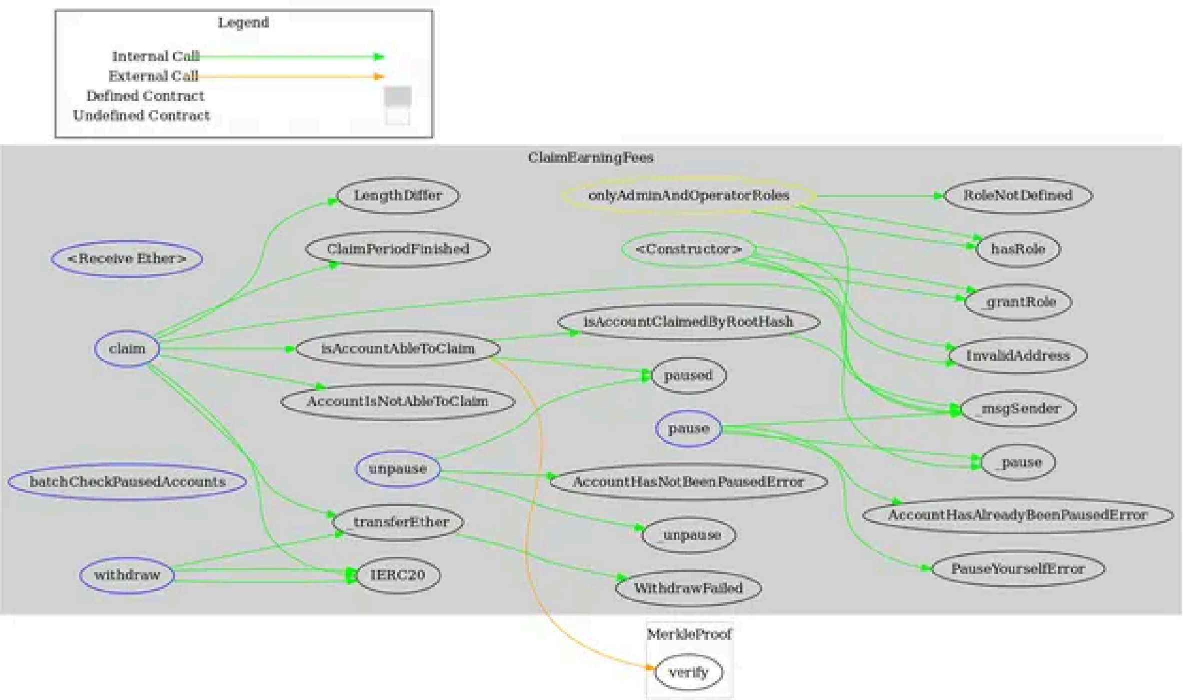
Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
ClaimEarningFees	Implementation	AccessControl, ReentrancyGuard, Pausable		
		Public	✓	-
		External	Payable	-
	claim	External	✓	whenNotPaused nonReentrant
	isAccountAbleToClaim	Public		-
	isAccountClaimedByRootHash	Public		-
	batchCheckPausedAccounts	External		-
	pause	External	✓	onlyAdminAndOperatorRoles
	unpause	External	✓	onlyAdminAndOperatorRoles
	withdraw	External	✓	onlyRole
	_transferEther	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ClaimEarningFees
LengthDiffer
onlyAdminAndOperatorRoles
RoleNotDefined
<Receive Ether>
ClaimPeriodFinished
<Constructor>
hasRole
_grantRole
isAccountClaimedByRootHash
claim
isAccountAbleToClaim
InvalidAddress
paused
AccountIsNotAbleToClaim
_msgSender
pause
_pause
batchCheckPausedAccounts
unpause
AccountHasNotBeenPausedError
AccountHasAlreadyBeenPausedError
transferEther
_unpause
PauseYourselfError
withdraw
IERC20
WithdrawFailed
MerkleProof
verify

Summary

Tea-Fi ClaimEarningFees contract implements a presale claim 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

cyberscope.io