Cyberscope

Audit Report

V8Coin

January 2025

Network	SEPOLIA
Address	0x3a25f0aAAE5eE47c26AA3647C9C4bb0C98074060
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

Explorer	https://sepolia.etherscan.io/address/0x3a25f0aaae5ee47c26aa3647c9c4bb0c98074060

Audit Updates

Initial Audit	23 Dec 2024 https://github.com/cyberscope-io/audits/blob/main/v8/v1/airdropClaim.pdf
Corrected Phase 2	27 Jan 2025

Source Files

Filename	SHA256
V8AirdropClaim.sol	c039bcf21be56953728d30559fe18975aff9eca69b75ecb332be88aa768015b6

Overview

The V8AirdropClaim contract is designed to facilitate and manage token airdrop claims with a focus on security and efficiency. It allows eligible users to claim rewards by validating their claims through a Merkle proof mechanism while enforcing strict role-based access controls for administrative functions. The contract also supports flexible management of airdrop sessions, claim fees, and rewards.

Claim Functionality

The claim functionality is the core of the contract, enabling users to securely receive airdrop rewards by providing a valid Merkle proof to prove their eligibility. The `claimReward` function ensures that each claim is valid by verifying the user's proof against the current Merkle root. Users are required to pay a predefined claim fee (`claimFeeAmount`), which is transferred to the designated `claimFeeReceiver` , ensuring proper fee handling. All users receive the same fixed reward amount of V8COIN tokens, which are transferred to the specified receiver address upon successful claim validation. Claims can only be made during active airdrop sessions, defined by the `claimSessionStart` and `claimSessionEnd` timestamps.

Roles

Admin

The admin can interact with the following functions:

- `setRoot`
- `rescueERC20`
- `updateClaimFeeReceiver`
- `updateClaimFeeAmount`
- `updateRewardAmount`

AIRDROP_SESSION_OPENER_ROLE

The AIRDROP_SESSION_OPENER_ROLE can interact with the following functions:

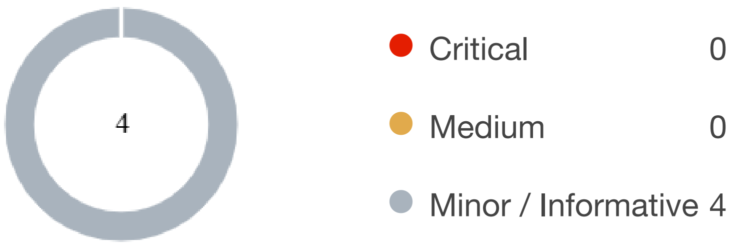
- `openAirdropSessionNow`
- `openAirdropSessionAt`

Users

The users can interact with the following functions:

- `claimReward`
- `isAirdropSessionOpen`
- `claimSessionStartHuman`
- `claimSessionEndHuman`
- `isProofValid`

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	0	4	0	0

Diagnostics

🔴 Critical 🟠 Medium ⚪ Minor / Informative

Severity	Code	Description	Status
⚪	TSI	Tokens Sufficiency Insurance	Acknowledged
⚪	MPC	Merkle Proof Centralization	Acknowledged
⚪	CCR	Contract Centralization Risk	Acknowledged
⚪	L04	Conformance to Solidity Naming Conventions	Acknowledged

TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Location	contracts/V8AirdropClaim.sol#L176
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.

```
uint256 rewardAmountCache = rewardAmount;
V8COIN.safeTransfer(_receiver, rewardAmountCache);
```

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 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 has acknowledged that this is not a security issue and states: *This is by design. We want the airdrop contract to hold a minimal amount of tokens, so in case of hack nothing bad happens. There is no incentive for attacks, tokens are not held in contract. It is a plus and best for the community.*

MPC - Merkle Proof Centralization

Criticality	Minor / Informative
Location	contracts/V8AirdropClaim.sol#L227,300
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.

```
function setRoot(bytes32 _root) external
onlyRole(DEFAULT_ADMIN_ROLE) {
    if (_root == 0x0) {
        revert ZeroRoot();
    }
    root = _root;
    emit SetRoot(_root);
}

function isProofValid(address _owner, bytes32[] memory _proof)
public view returns(bool) {
    bytes32 leaf =
keccak256(bytes.concat(keccak256(abi.encode(_owner,
CHAIN_ID))));
}
```

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: *Merkel tree is by design a very centralized solution, a lot of projects use it. It's efficient for cost effectiveness for project and airdrop participants.*

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	contracts/V8AirdropClaim.sol#L194,209,227,255,270,284
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 openAirdropSessionNow(uint256
sessionDurationInMinutes) external
onlyRole(AIRDROP_SESSION_OPENER_ROLE) {/*...*/}


    function openAirdropSessionAt(
        uint256 startYear,
        uint256 startMonth,
        uint256 startDay,
        uint256 startHours,
        uint256 startMinutes,
        uint256 sessionDurationInMinutes
    ) external onlyRole(AIRDROP_SESSION_OPENER_ROLE) {/*...*/}

    function setRoot(bytes32 _root) external
onlyRole(DEFAULT_ADMIN_ROLE) {/*...*/}

    function updateClaimFeeReceiver(address payable
_claimFeeReceiver) external onlyRole(DEFAULT_ADMIN_ROLE)
{/*...*/}

    function updateClaimFeeAmount(uint256 _claimFeeAmount)
external onlyRole(DEFAULT_ADMIN_ROLE) {/*...*/}

    function updateRewardAmount(uint256 _rewardAmount) external
onlyRole(DEFAULT_ADMIN_ROLE) {/*...*/}
```

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: *We opted for cold storage hardware wallets. Multisignature is not necessarily a guarantee of decentralization. It just means different signatures permissions.Project has not been launched so it is normal to see huge wallets during distribution and before project launch.*

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	V8AirdropClaim.sol#L82,86,162,227,240,255,270,284,300
Status	Acknowledged

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.

```
IERC20Metadata immutable public V8COIN
uint256 immutable public CHAIN_ID
address payable _receiver
bytes32[] memory _proof
bytes32 _root
IERC20Metadata _token
address payable _claimFeeReceiver
uint256 _claimFeeAmount
uint256 _rewardAmount
address _owner
```

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.

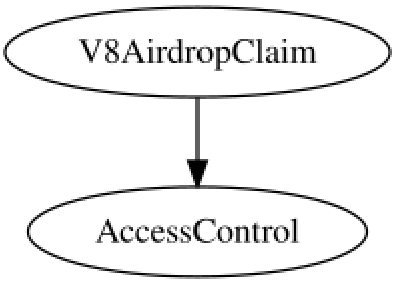
Team Update

The team has acknowledged that this is not a security issue and states: *As we understand from your report, you see problem in V8COIN constant name and internal/private variables with leading underscore (that are listed in the report). V8COIN is the ticker and full name of the coin, it can't be used with underscore (like V8_COIN, this is unacceptable), if we got your recommendation correctly. Regarding private/internal functions it's allowed and recommended to use leading underscore in style guides:*
https://docs.soliditylang.org/en/v0.8.24/style-guide.html#underscore-prefix-for-non-external-functions-and-variables .

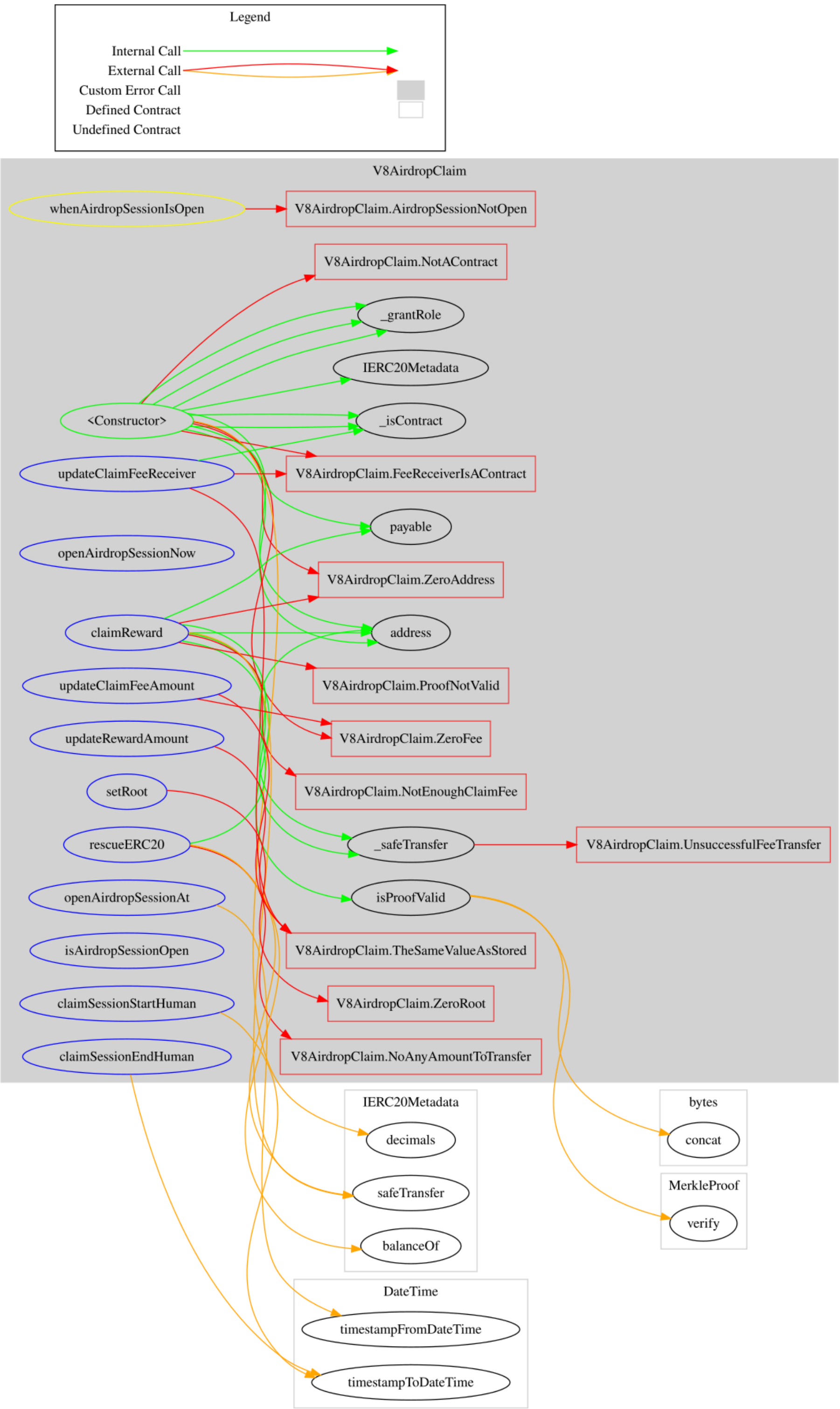
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
V8AirdropClaim	Implementation	AccessControl		
		Public	✓	-
	claimReward	External	Payable	whenAirdropSessionIsOpen
	openAirdropSessionNow	External	✓	onlyRole
	openAirdropSessionAt	External	✓	onlyRole
	setRoot	External	✓	onlyRole
	rescueERC20	External	✓	onlyRole
	updateClaimFeeReceiver	External	✓	onlyRole
	updateClaimFeeAmount	External	✓	onlyRole
	updateRewardAmount	External	✓	onlyRole
	isProofValid	Public		-
	isAirdropSessionOpen	External		-
	claimSessionStartHuman	External		-
	claimSessionEndHuman	External		-
	_safeTransfer	Internal	✓	
	_isContract	Internal		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
V8AirdropClaim
whenAirdropSessionIsOpen
V8AirdropClaim.AirdropSessionNotOpen
V8AirdropClaim.NotAContract
_grantRole
IERC20Metadata
<Constructor>
_isContract
updateClaimFeeReceiver
V8AirdropClaim.FeeReceiverIsAContract
payable
openAirdropSessionNow
V8AirdropClaim.ZeroAddress
claimReward
address
updateClaimFeeAmount
V8AirdropClaim.ProofNotValid
updateRewardAmount
V8AirdropClaim.ZeroFee
setRoot
V8AirdropClaim.NotEnoughClaimFee
rescueERC20
_safeTransfer
V8AirdropClaim.UnsuccessfulFeeTransfer
openAirdropSessionAt
isProofValid
isAirdropSessionOpen
V8AirdropClaim.TheSameValueAsStored
claimSessionStartHuman
V8AirdropClaim.ZeroRoot
claimSessionEndHuman
V8AirdropClaim.NoAnyAmountToTransfer
IERC20Metadata
decimals
safeTransfer
balanceOf
bytes
concat
MerkleProof
verify
DateTime
timestampFromDateTime
timestampToDateTime

Summary

The V8AirdropClaim contract implements a secure and efficient mechanism for managing token airdrop claims, incorporating features such as Merkle proof validation, role-based access control, and session-based claim management. This audit investigates potential security vulnerabilities, evaluates the correctness of business logic, and identifies areas for improvement to enhance the contract’s robustness and functionality.

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