Cyberscope

Audit Report

SuperWhale

March 2024

Network BSC_TESTNET

Address 0x6e83c8ded0415c9b90078d46ea363ad68583322b

Audited by © cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Findings Breakdown 4
Diagnostics 5
IMC - Insufficient MaxHold Check 6
Description 6
Recommendation 6
BC - Blacklists Addresses 8
Description 8
Recommendation 8
CCR - Contract Centralization Risk 10
Description 10
Recommendation 11
IDI - Immutable Declaration Improvement 12
Description 12
Recommendation 12
MEM - Missing Error Messages 13
Description 13
Recommendation 13
RSW - Redundant Storage Writes 14
Description 14
Recommendation 14
ST - Stops Transactions 15
Description 15
Recommendation 15
L04 - Conformance to Solidity Naming Conventions 17
Description 17
Recommendation 17
L07 - Missing Events Arithmetic 19
Description 19
Recommendation 19
Functions Analysis 20
Inheritance Graph 22
Flow Graph 23
Summary 24
Disclaimer 25
About Cyberscope 26

Review

Contract Name	SuperWallyAvatar
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	200 runs
Explorer	https://testnet.bscscan.com/address/0x6e83c8ded0415c9b90078d46ea363ad68583322b
Address	0x6e83c8ded0415c9b90078d46ea363ad68583322b
Network	BSC_TESTNET
Symbol	SWA

Audit Updates

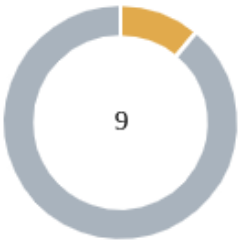
Initial Audit	09 Mar 2024

Source Files

Filename	SHA256
erc721a/contracts/IERC721A.sol	6bcabae29d620c91ee24de234cf53de46bffcb87631caab20874ee1365fb81ad
erc721a/contracts/ERC721A.sol	5c90c48b1aa37e8702ddc416697649c2a3f5d8d6c58b366f591fbc8baf6384a1
erc721a/contracts/extensions/IERC721AQueryable.sol	cb7a737553421cbdae1345a4350a0a9d19c955e09a5144b88fe92a243fc63a68
erc721a/contracts/extensions/ERC721AQueryable.sol	14513c51d915a6ec0917321f6f93767d610591570f0cc5f4fe5f8f01b215873f

contracts/SuperWallyAvatar.sol	d8ec967f45b3591f4a0733ef91b6b9a732b9962ea2aa1170165d77257e415a7e
@openzeppelin/contracts/utils/Strings.sol	f81f11dca62dcd3e0895e680559676f4ba4f2e12a36bb0291d7ecbb6b983141f
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/Address.sol	8160a4242e8a7d487d940814e5279d934e81f0436689132a4e73394bab084a6d
@openzeppelin/contracts/utils/math/Math.sol	8059d642ec219d0b9b62fbc76912079529cf494cac988abe5e371f1168b29b0f
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
@openzeppelin/contracts/utils/introspection/ERC165.sol	8806a632d7b656cadb8133ff8f2acae4405b3a64d8709d93b0fa6a216a8a6154
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	77f0f7340c2da6bb9edbc90ab6e7d3eb8e2ae18194791b827a3e8c0b11a09b43
@openzeppelin/contracts/token/ERC721/IERC721.sol	c7703068bac02fe1cdf109e38faf10399c66eb411e4c9ae0d70c009eca4bf5ef
@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol	f16b861aa1f623ccc5e173f1a82d8cf45b678a7fb81e05478fd17eb2ccb7b37e
@openzeppelin/contracts/security/ReentrancyGuard.sol	3b30604df38d0f9b2b281a3e6661eb1b9cd577579e66225c674df21ca5b89b2c
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	1	0	0	0
Minor / Informative	8	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Medium	IMC	Insufficient MaxHold Check	Unresolved
Minor / Informative	BC	Blacklists Addresses	Unresolved
Minor / Informative	CCR	Contract Centralization Risk	Unresolved
Minor / Informative	IDI	Immutable Declaration Improvement	Unresolved
Minor / Informative	MEM	Missing Error Messages	Unresolved
Minor / Informative	RSW	Redundant Storage Writes	Unresolved
Minor / Informative	ST	Stops Transactions	Unresolved
Minor / Informative	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor / Informative	L07	Missing Events Arithmetic	Unresolved

IMC - Insufficient MaxHold Check

Criticality	Medium
Location	contracts/SuperWallyAvatar.sol#L156
Status	Unresolved

Description

The contract includes a critical check in the `_beforeTokenTransfers` function, aiming to ensure that the balance of the recipient does not exceed a predefined maximum holding (`maxHold`). However, this check is prematurely executed before the actual token transfer takes place, resulting in an evaluation against the recipient's current balance rather than their future balance post-transfer. Since the recipient's balance has not yet been updated to include the new tokens, this check may inadvertently allow transfers that result in the recipient exceeding the `maxHold` limit. This oversight means that even if the recipient's balance is just below the `maxHold` threshold before the transfer, they could receive an amount that puts them over the limit without the contract enforcing the intended restriction.

```
    function _beforeTokenTransfers(
        address from,
        address to,
        uint256 startTokenId,
        uint256 stopTokenId
    ) internal override {
        uint256 toBalance = balanceOf(to);
        require(!blacklist[from] && !blacklist[to], "Address is
blacklisted!");
        // whitelist addresses and owner can bypass the limit
        if (!whitelist[to] && to != owner()) {
            require(toBalance < maxHold, "You can't hold more than 3
tokens!");
        }
        super._beforeTokenTransfers(from, to, startTokenId,
stopTokenId);
    }
```

Recommendation

It is recommended to adjust the logic to account for the tokens being transferred when conducting the `maxHold` check. This can be achieved by including the number of tokens being transferred in the balance check calculation. Specifically, the contract should calculate the recipient's prospective balance by adding the amount of tokens to be transferred to their current balance and then compare this sum against the `maxHold` limit. This approach ensures that the check accurately reflects the recipient's balance post-transfer, effectively enforcing the maximum holding policy. Implementing this change will require modifying the condition within the `_beforeTokenTransfers` function to include the calculation of the incoming tokens as part of the `toBalance` variable or a similar comparison. This adjustment will safeguard against unintentional limit breaches and align the contract's functionality with its intended protective measures regarding maximum token holdings.

BC - Blacklists Addresses

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L68,163
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `setBlacklist` function.

```
   function setBlacklist(address _address, bool _state)
external onlyOwner {
        blacklist[_address] = _state;
    }

    function _beforeTokenTransfers(
        address from,
        address to,
        uint256 startTokenId,
        uint256 stopTokenId
    ) internal override {
        uint256 toBalance = balanceOf(to);
        require(!blacklist[from] && !blacklist[to], "Address is
blacklisted!");
        // whitelist addresses and owner can bypass the limit
        if (!whitelist[to] && to != owner()) {
            require(toBalance < maxHold, "You can't hold more
than 3 tokens!");
        }
        super._beforeTokenTransfers(from, to, startTokenId,
stopTokenId);
    }
```

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.

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.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L72,76,99,125,129,135,139
Status	Unresolved

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 contract is designed with a high level of centralization, granting the owner comprehensive control over several critical aspects of the NFT ecosystem. This includes the ability to whitelist addresses, determine the maximum number of tokens an address can hold (`maxHold`), mint tokens for specific addresses, set the base URI for the NFT metadata, define the cost of tokens, limit the number of tokens mintable per transaction (`maxMintAmountPerTx`), and toggle the minting process on or off (`paused`). Such centralized control introduces risks including potential manipulation, unfair advantage, or even censorship by the contract owner. While centralization might be intended for administrative convenience or to enforce certain rules during the early stages of project deployment, it can undermine the security in the NFT ecosystem.

```
    function setWhitelist(address _address, bool _state)
external onlyOwner {
        whitelist[_address] = _state;
    }

    function setMaxHold(uint256 _maxHold) external onlyOwner {
        maxHold = _maxHold;
    }

    function mintForAddress(
        uint256 _mintAmount,
        address _receiver
    ) public mintCompliance(_mintAmount) onlyOwner {
        _safeMint(_receiver, _mintAmount);
    }

    function setCost(uint256 _cost) public onlyOwner {
        cost = _cost;
    }

    function setMaxMintAmountPerTx(
        uint256 _maxMintAmountPerTx
    ) public onlyOwner {
        maxMintAmountPerTx = _maxMintAmountPerTx;
    }

    function setUriPrefix(string memory _uriPrefix) public
onlyOwner {
        uriPrefix = _uriPrefix;
    }

    function setPaused(bool _state) public onlyOwner {
        paused = _state;
    }
```

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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L47
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
maxSupply
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L148
Status	Unresolved

Description

The contract is using missing error messages. These are no error messages to accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(os)
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L139
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
    function setCost(uint256 _cost) public onlyOwner {
        cost = _cost;
    }

    function setMaxMintAmountPerTx(
        uint256 _maxMintAmountPerTx
    ) public onlyOwner {
        maxMintAmountPerTx = _maxMintAmountPerTx;
    }

    function setPaused(bool _state) public onlyOwner {
        paused = _state;
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

ST - Stops Transactions

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L76,165
Status	Unresolved

Description

The contract owner can effectively halt the transfer of NFT tokens by setting the `maxHold` parameter to zero. This parameter is intended to limit the number of tokens an address can hold, presumably to ensure fair distribution or to prevent hoarding. However, the ability to set `maxHold` to zero goes beyond these purposes, as it can completely prevent all users from receiving tokens, effectively freezing the transferability of the NFTs within the ecosystem. This mechanism can be used to exert control over the asset, potentially undermining the liquidity and utility of the NFTs for all participants.

```
    function setMaxHold(uint256 _maxHold) external onlyOwner {
        maxHold = _maxHold;
    }

    function _beforeTokenTransfers(
        address from,
        address to,
        uint256 startTokenId,
        uint256 stopTokenId
    ) internal override {
        ...
        // whitelist addresses and owner can bypass the limit
        if (!whitelist[to] && to != owner()) {
            require(toBalance < maxHold, "You can't hold more
than 3 tokens!");
        }
        ...
    }
```

Recommendation

It is recommended to implement safeguards against the misuse of the maxHold function to ensure the continuity and fairness of token transfers. One approach could be to set a

reasonable minimum limit for `maxHold` that cannot be exceeded by the owner, ensuring that the feature cannot be used to completely halt the transfers of the NFT tokens.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/SuperWallyAvatar.sol#L68,72,76,80,87,100,101,111,125,130,135,139
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.

```
bool _state
address _address
uint256 _maxHold
uint256 _mintAmount
address _receiver
uint256 _tokenId
uint256 _cost
uint256 _maxMintAmountPerTx
string memory _uriPrefix
```

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	contracts/SuperWallyAvatar.sol#L70,73,77,126
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.

```
blacklist[_address] = _state;
whitelist[_address] = _state;
maxHold = _maxHold;
cost = _cost;
```

Recommendation

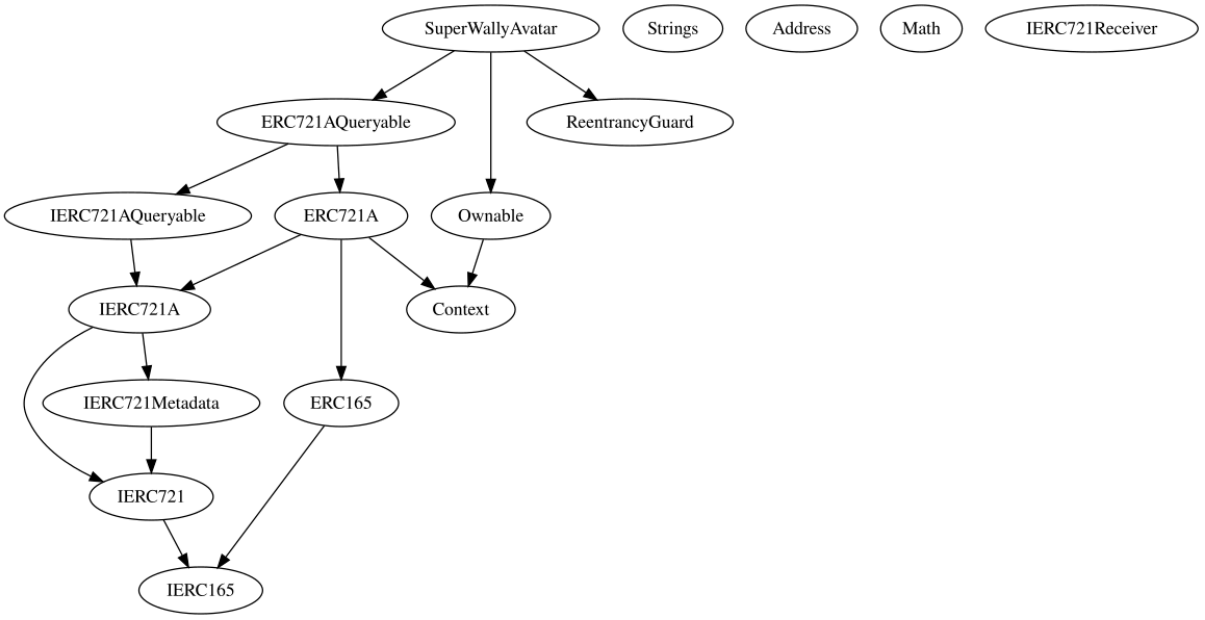
By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

Functions Analysis

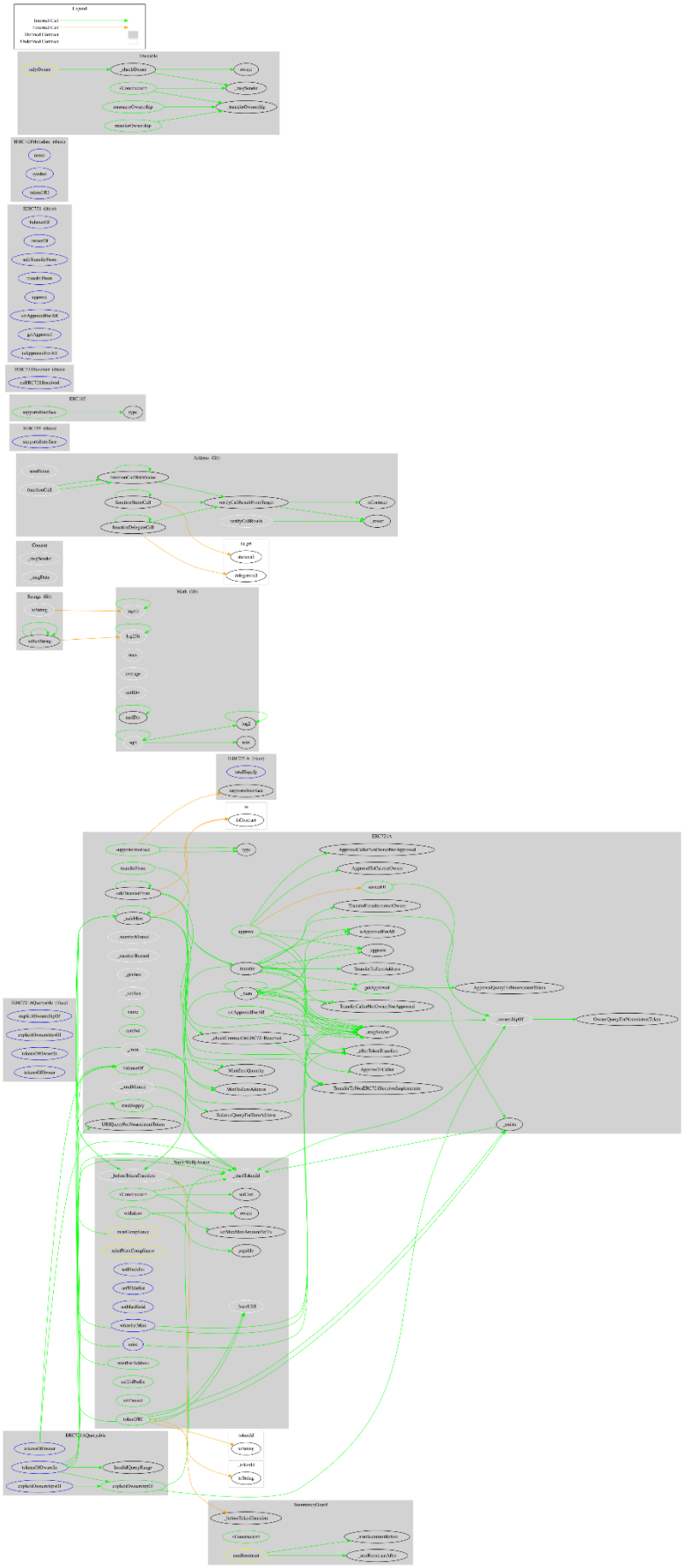
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SuperWallyAvatar	Implementation	ERC721AQueryable, Ownable, ReentrancyGuard		
		Public	✓	ERC721A
	setBlacklist	External	✓	onlyOwner
	setWhitelist	External	✓	onlyOwner
	setMaxHold	External	✓	onlyOwner
	whitelistMint	External	✓	-
	mint	External	Payable	mintCompliance mintPriceCompliance
	mintForAddress	Public	✓	mintCompliance onlyOwner
	_startTokenId	Internal		
	tokenURI	Public		-
	setCost	Public	✓	onlyOwner
	setMaxMintAmountPerTx	Public	✓	onlyOwner
	setUriPrefix	Public	✓	onlyOwner
	setPaused	Public	✓	onlyOwner
	withdraw	Public	✓	onlyOwner nonReentrant
	_baseURI	Internal		

	_beforeTokenTransfers	Internal	✓	

Inheritance Graph



Flow Graph



Summary

SuperWhale contract implements an NFT 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