Cyberscope

Audit Report

ACHIVX

August 2024

Files: Token.sol, MultisigManager.sol, ManagedToken.sol

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

Overview

The ACHIVX project completed an audit of its core smart contracts, specifically focusing on Token.sol, MultisigManager.sol, and ManagedToken.sol. ACHIVX stands out as a promising initiative with a steadily growing community. The audit covered key aspects of the project's smart contracts including the token contract itself, its interface, and a multi-signature contract that enables authorised entities to execute administrative actions on the token.

Specifically, the multi-signature contract introduces a majority-based mechanism, where eligible participants can submit requests to perform administrative tasks on the token contract. These tasks may include minting new tokens, blacklisting addresses and burning their tokens, pausing the token contract, or modifying its ownership. A request is approved only when a majority of the designated voters consent to the action.

Through these contracts, the ACHIVX project showcases an innovative approach to managing token contracts by leveraging a multi-signature system for enhanced decentralisation.

Review

Testing Deploys	https://sepolia.etherscan.io/address/0xd6abb1751e0fb60de1735cd81434f9a1f27b7e48 https://sepolia.etherscan.io/address/0x0634BA97Aa6b16dDF40b21c0F3d0f8CdCd191cEe

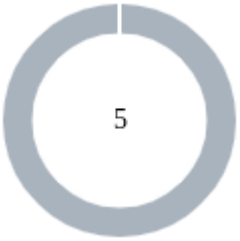
Audit Updates

Initial Audit	15 Aug 2024 https://github.com/cyberscope-io/audits/blob/main/achivx/v1/audit.pdf
Corrected Phase 2	30 Aug 2024

Source Files

Filename	SHA256
Token.sol	1b185b9bb150150bd92f6659f20e3921b339a5d00fc75e68aa9ff940e5fe3a6a
MultisigManager.sol	975a07fe621cdd52ac4ead91d3716038facfed34c6bd4f45e4d1559d8d624724
ManagedToken.sol	20b1996bffd2c8515a345274af8ee937d27c71a99321d29aac6161e071d5fe81

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	5	0	0	0

Diagnostics

Severity	Code	Description	Status
●	BC	Blacklists Addresses	Unresolved
●	BBT	Burns Blacklisted Tokens	Unresolved
●	CCR	Contract Centralization Risk	Unresolved
●	MT	Mints Tokens	Unresolved
●	ST	Stops Transactions	Unresolved

BC - Blacklists Addresses

Criticality	Minor / Informative
Location	Token.sol#L297
Status	Unresolved

Description

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

```
function addBlackList(address evilUser) external onlyOwner {
        isBlackListed[evilUser] = true;
        emit AddedBlackList(evilUser);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account.In the current context, the team should ensure that the ownership of the Token contract is securely assigned to the `MultisigManager.sol` contract at the time of deployment. This setup ensures that all critical functions are protected by the multi-signature mechanism, thereby reducing the risk of a single point of failure or unauthorized access.

Additionally, it is advised to validate that the deployment and initialization process of the Token contract does not inadvertently allow the contract to be owned by any account other than the intended `MultisigManager.sol` contract.

Suggested Solutions:

These measures, while improving security, do not eliminate the severity of the finding:

-Ensure that the signers of the multi-signature contract are distinct, independent entities with rigorously managed private keys.

-Ensure the multi-signature contract `MultisigManager.sol` is properly configured and operational from deployment.

-Ensure that the ownership of the Token contract is irrevocably assigned to the `MultisigManager.sol` contract and cannot be modified using the `transferOwnership()` method.

BBT - Burns Blacklisted Tokens

Criticality	Minor / Informative
Location	Token.sol#L313
Status	Unresolved

Description

The contract owner has the authority to burn tokens from a blacklisted address. The owner may take advantage of it by calling the `destroyBlackFunds()` function. As a result, the targeted address will lose the corresponding tokens.

```
function destroyBlackFunds(address blackListedUser) external onlyOwner {
        require(isBlackListed[blackListedUser], "account not
blacklisted");
        uint dirtyFunds = balanceOf(blackListedUser);
        balances[blackListedUser] = 0;
        _totalSupply -= dirtyFunds;
        emit DestroyedBlackFunds(blackListedUser, dirtyFunds);
        emit Transfer(blackListedUser, address(0), dirtyFunds);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account.In the current context, the team should ensure that the ownership of the Token contract is securely assigned to the `MultisigManager.sol` contract at the time of deployment. This setup ensures that all critical functions are protected by the multi-signature mechanism, thereby reducing the risk of a single point of failure or unauthorized access.

Additionally, it is advised to validate that the deployment and initialization process of the Token contract does not inadvertently allow the contract to be owned by any account other than the intended `MultisigManager.sol` contract.

Suggested Solutions:

These measures, while improving security, do not eliminate the severity of the finding:

-Ensure that the signers of the multi-signature contract are distinct, independent entities with rigorously managed private keys.

-Ensure the multi-signature contract `MultisigManager.sol` is properly configured and operational from deployment.

-Ensure that the ownership of the Token contract is irrevocably assigned to the `MultisigManager.sol` contract and cannot be modified using the `transferOwnership()` method.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	Token.sol#L262,270,297,305,313,398,698,711
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.

```
function pause() external onlyOwner whenNotPaused {
        ...
}
function unpause() external onlyOwner whenPaused {
        ...
}
function addBlackList(address evilUser) external onlyOwner {
        ...
}
function removeBlackList(address clearedUser) external onlyOwner {
        ...
}
function destroyBlackFunds(address blackListedUser) external onlyOwner {
        ...
}
function deprecate(address upgradedAddress) external onlyOwner {
        ...
}
function issue(uint amount,address to) external onlyOwner
        whenNotDeprecated {
        ...
}
function redeem(uint amount) external onlyOwner whenNotDeprecated {
        ...
}
```

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.In addition, ownership should be permanently assigned to the `MultisigManager.sol` contract, with independent signers whose interests are distinct and private keys securely managed. Consider disabling ownership transfers and exploring further decentralization mechanisms to minimize reliance on a single entity.

MT - Mints Tokens

Criticality	Minor / Informative
Location	Token.sol#L698
Status	Unresolved

Description

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

```
function issue(
        uint amount,
        address to
    ) external onlyOwner whenNotDeprecated {
        balances[to] += amount;
        _totalSupply += amount;
        emit Issue(amount, to);
        emit Transfer(address(0), to, amount);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account.In the current context, the team should ensure that the ownership of the Token contract is securely assigned to the `MultisigManager.sol` contract at the time of deployment. This setup ensures that all critical functions are protected by the multi-signature mechanism, thereby reducing the risk of a single point of failure or unauthorized access.

Additionally, it is advised to validate that the deployment and initialization process of the Token contract does not inadvertently allow the contract to be owned by any account other than the intended `MultisigManager.sol` contract.

Suggested Solutions:

These measures, while improving security, do not eliminate the severity of the finding:

-Ensure that the signers of the multi-signature contract are distinct, independent entities with rigorously managed private keys.

-Ensure the multi-signature contract `MultisigManager.sol` is properly configured and operational from deployment.

-Ensure that the ownership of the Token contract is irrevocably assigned to the `MultisigManager.sol` contract and cannot be modified using the `transferOwnership()` method.

ST - Stops Transactions

Criticality	Minor / Informative
Location	Token.sol#L262
Status	Unresolved

Description

The contract owner has the authority to stop transactions for all users. The owner may take advantage of it by calling the `pause()` function.

```
function pause() external onlyOwner whenNotPaused {
        paused = true;
        emit Pause();
}
```

Recommendation

The team should carefully manage the private keys of the owner's account.In the current context, the team should ensure that the ownership of the Token contract is securely assigned to the `MultisigManager.sol` contract at the time of deployment. This setup ensures that all critical functions are protected by the multi-signature mechanism, thereby reducing the risk of a single point of failure or unauthorized access.

Additionally, it is advised to validate that the deployment and initialization process of the Token contract does not inadvertently allow the contract to be owned by any account other than the intended `MultisigManager.sol` contract.

Suggested Solutions:

These measures, while improving security, do not eliminate the severity of the finding:

-Ensure that the signers of the multi-signature contract are distinct, independent entities with rigorously managed private keys.

-Ensure the multi-signature contract `MultisigManager.sol` is properly configured and operational from deployment.

-Ensure that the ownership of the Token contract is irrevocably assigned to the `MultisigManager.sol` contract and cannot be modified using the `transferOwnership()` method.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
Ownable	Implementation	ERC173		
		Public	✓	-
	transferOwnership	External	✓	onlyOwner
ERC20Basic	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
ERC20	Interface	ERC20Basic		
	allowance	External		-
	transferFrom	External	✓	-
	approve	External	✓	-

ERC20Extended	Interface	ERC20		
	batchTransfer	External	✓	-
BasicToken	Implementation	Ownable, ERC20Basic		
		Public	✓	-
	transfer	Public	✓	-
	balanceOf	Public		-
StandardToken	Implementation	BasicToken, ERC20		
	transferFrom	Public	✓	-
	approve	Public	✓	-
	allowance	Public		-
ExtendedToken	Implementation	StandardToken, ERC20Extended		
	batchTransfer	Public	✓	-
Pausable	Implementation	Ownable, IPausable		
	pause	External	✓	onlyOwner whenNotPaused
	unpause	External	✓	onlyOwner whenPaused
BlackList	Implementation	Ownable, BasicToken, IBlackList		

	addBlackList	External	✓	onlyOwner
	removeBlackList	External	✓	onlyOwner
	destroyBlackFunds	External	✓	onlyOwner
UpgradedStandardToken	Interface	ERC20		
	transferByLegacy	External	✓	-
	transferFromByLegacy	External	✓	-
	approveByLegacy	External	✓	-
	batchTransferByLegacy	External	✓	-
Deprecateable	Implementation	Ownable, IDeprecatable		
		Public	✓	-
	deprecate	External	✓	onlyOwner
LegacyToken	Interface	ERC20, IDeprecatable		
	legacyBalance	External		-
	legacyAllowance	External		-
	emitTransfer	External	✓	-
	emitApproval	External	✓	-
Token	Implementation	Deprecateable, LegacyToken, Pausable, ExtendedToken, BlackList,		

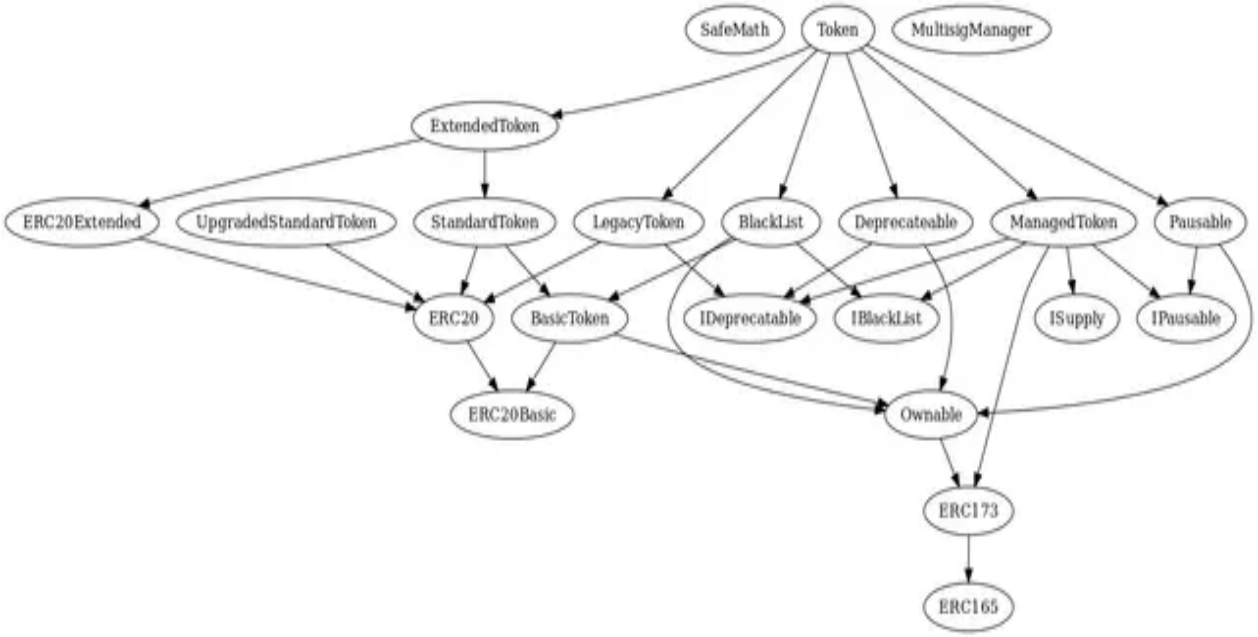
		ManagedToken		
		Public	✓	Ownable BasicToken
	supportsInterface	External		-
	transfer	Public	✓	whenNotPaused whenNotBlackListed
	transferFrom	Public	✓	whenNotPaused whenNotBlackListed
	balanceOf	Public		-
	approve	Public	✓	-
	allowance	Public		-
	batchTransfer	Public	✓	whenNotPaused whenNotBlackListed
	totalSupply	Public		-
	legacyBalance	External		onlyUpgraded
	legacyAllowance	External		onlyUpgraded
	emitTransfer	External	✓	onlyUpgraded
	emitApproval	External	✓	onlyUpgraded
	issue	External	✓	onlyOwner whenNotDeprecated
	redeem	External	✓	onlyOwner whenNotDeprecated
MultisigManager	Implementation			
	_addVotingAccount	Private	✓	

	_removeVotingAccount	Private	✓	
	_makeRequest	Private	✓	
	_approveRequest	Private	✓	
	getMinApprovals	Public		-
		Public	✓	-
	requestOwnerChange	External	✓	-
	approveOwnerChange	External	✓	-
	requestVotersListChange	External	✓	-
	approveVotersListChange	External	✓	-
	requestTokenPause	External	✓	-
	approveTokenPause	External	✓	-
	requestTokenUnpause	External	✓	-
	approveTokenUnpause	External	✓	-
	requestBlacklist	External	✓	-
	approveBlacklist	External	✓	-
	requestUnblacklist	External	✓	-
	approveUnblacklist	External	✓	-
	requestBlackFundsDestruction	External	✓	-
	approveBlackFundsDestruction	External	✓	-
	requestDeprecation	External	✓	-
	approveDeprecation	External	✓	-
	requestIssue	External	✓	-
	approveIssue	External	✓	-
	requestRedeem	External	✓	-
	approveRedeem	External	✓	-

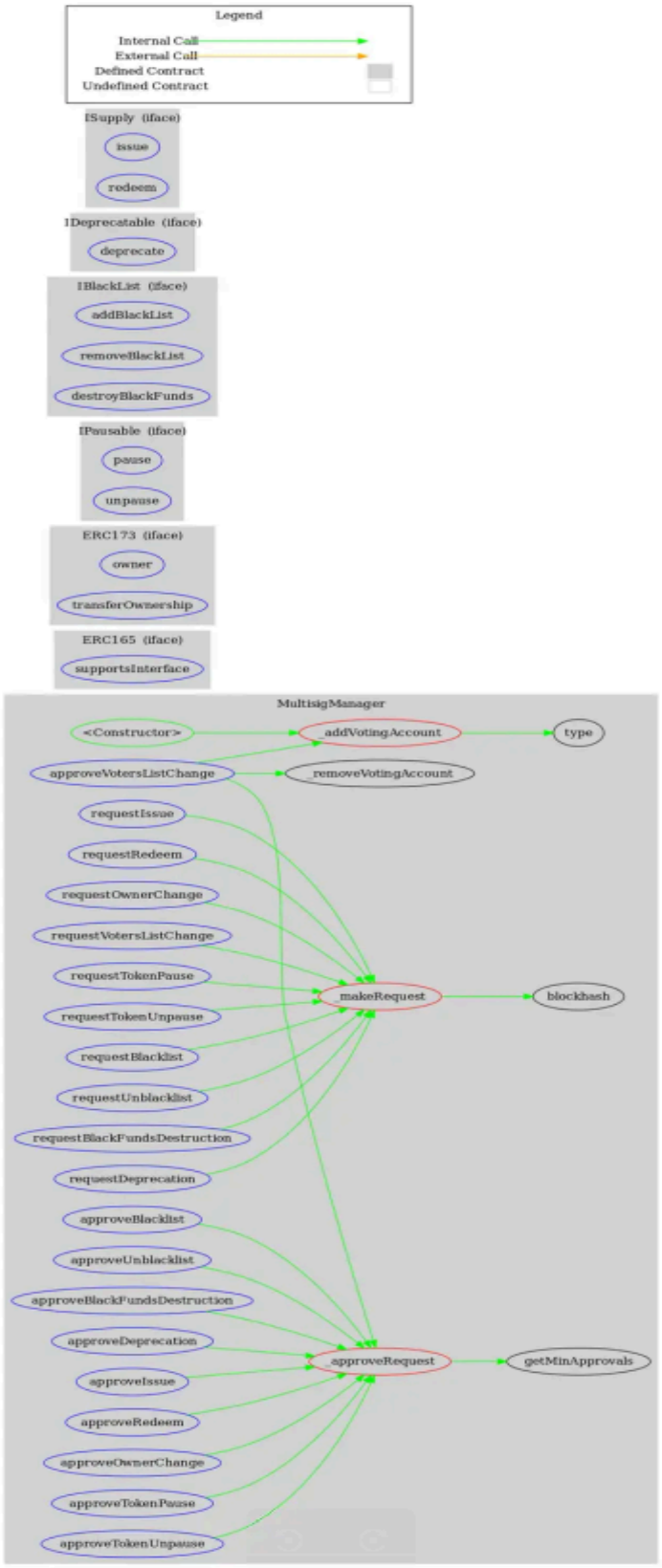
ERC165	Interface			
	supportsInterface	External		-
ERC173	Interface	ERC165		
	owner	External		-
	transferOwnership	External	✓	-
IPausable	Interface			
	pause	External	✓	-
	unpause	External	✓	-
IBlackList	Interface			
	addBlackList	External	✓	-
	removeBlackList	External	✓	-
	destroyBlackFunds	External	✓	-
IDeprecatable	Interface			
	deprecate	External	✓	-
ISupply	Interface			
	issue	External	✓	-
	redeem	External	✓	-

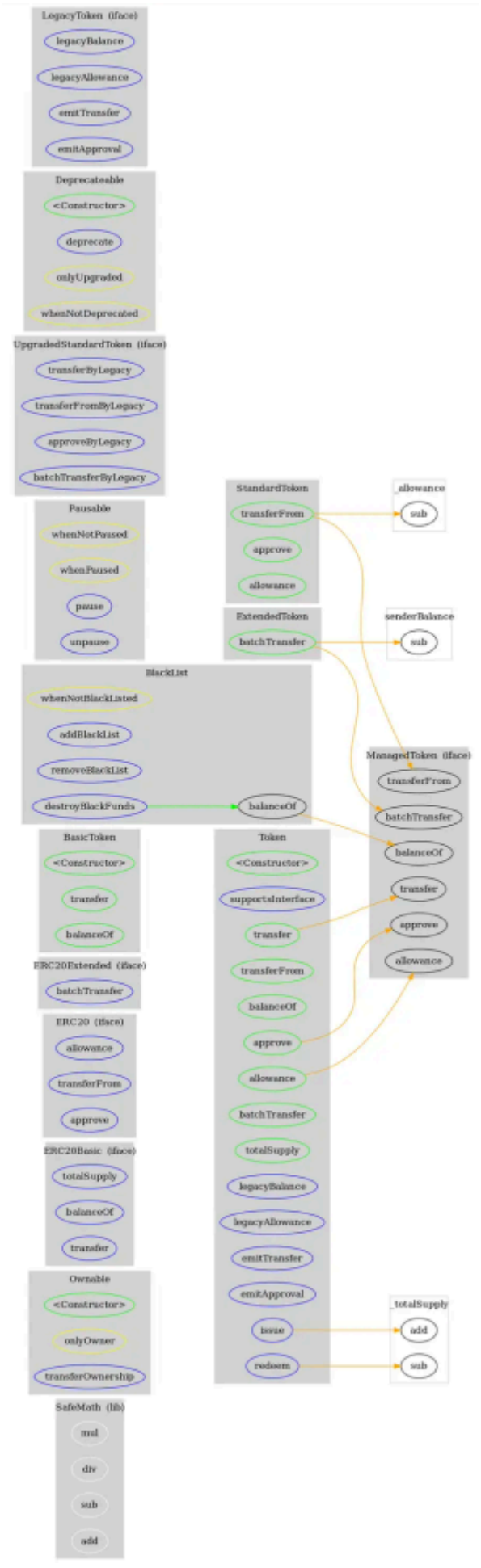
ManagedToken	Interface	ERC173, IPausable, IBlackList, IDeprecatable, ISupply		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ISupply (iface)
issue
redeem
IDeprecatable (iface)
deprecate
IBlackList (iface)
addBlackList
removeBlackList
destroyBlackFunds
IPausable (iface)
pause
unpause
ERC173 (iface)
owner
transferOwnership
ERC165 (iface)
supportsInterface
MultisigManager
<Constructor>
addVotingAccount
type
approveVotersListChange
removeVotingAccount
requestIssue
requestRedeem
requestOwnerChange
requestVotersListChange
requestTokenPause
requestTokenUnpause
requestBlacklist
requestUnblacklist
requestBlackFundsDestruction
requestDeprecation
makeRequest
blockhash
approveBlacklist
approveUnblacklist
approveBlackFundsDestruction
approveDeprecation
approveIssue
approveRedeem
approveOwnerChange
approveTokenPause
approveTokenUnpause
approveRequest
getMinApprovals


LegacyToken (iface)
legacyBalance
legacyAllowance
emitTransfer
emitApproval
Deprecateable
<Constructor>
deprecate
onlyUpgraded
whenNotDeprecated
UpgradedStandardToken (iface)
transferByLegacy
transferFromByLegacy
approveByLegacy
batchTransferByLegacy
Pausable
whenNotPaused
whenPaused
pause
unpause
StandardToken
transferFrom
approve
allowance
allowance
sub
ExtendedToken
batchTransfer
senderBalance
sub
BlackList
whenNotBlackListed
addBlackList
removeBlackList
destroyBlackFunds
balanceOf
ManagedToken (iface)
transferFrom
batchTransfer
balanceOf
transfer
approve
allowance
BasicToken
<Constructor>
transfer
balanceOf
Token
<Constructor>
supportsInterface
transfer
transferFrom
balanceOf
approve
allowance
batchTransfer
totalSupply
legacyBalance
legacyAllowance
emitTransfer
emitApproval
issue
redeem
ERC20Extended (iface)
batchTransfer
ERC20 (iface)
allowance
transferFrom
approve
ERC20Basic (iface)
totalSupply
balanceOf
transfer
totalSupply
add
sub
Ownable
<Constructor>
onlyOwner
transferOwnership
SafeMath (lib)
mul
div
sub
add

Summary

ACHIVX is an innovative project with a rapidly growing community. The audit of its smart contracts revealed no compiler errors or critical issues. However, it is important to note that the contract owner has access to administrative functions that could potentially be misused if not properly managed. To mitigate this risk, the project has integrated a multi-signature contract intended to serve as the owner of the Token contract. This significantly enhances decentralisation by ensuring that all critical functions are secured through the multi-signature mechanism, which reduces the likelihood of a single point of failure or unauthorised access. Nevertheless, for effective decentralisation and security, it is crucial that the multi-signature wallet is correctly configured as the owner of the token contract from the deployment, and that the signers are independent entities with securely managed private keys.

This audit addresses security concerns, evaluates business logic, and suggests potential improvements to strengthen the overall robustness of the ACHIVX project.

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