Cyberscope

Audit Report

Axondao

September 2023

Network	Goerli
Address	0xCb805f61e28f550279F268A0F091A52cbed6c904
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	RSD	Redundant Swap Duplication	Unresolved
●	EIS	Excessively Integer Size	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	FSA	Fixed Swap Address	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	AXGT
Compiler Version	v0.8.18+commit.87f61d96
Optimization	200 runs
Explorer	https://goerli.etherscan.io/address/0xcb805f61e28f550279f268a0f091a52cbed6c904
Address	0xcb805f61e28f550279f268a0f091a52cbed6c904
Network	GOERLI
Symbol	AXGT
Decimals	18
Total Supply	1,000,000,000

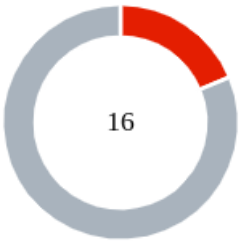
Audit Updates

Initial Audit	20 Sep 2023

Source Files

Filename	SHA256
AXGT.sol	62c28a61045d3bbaa31abaf18c2fc0bddea5967ad37e3be8cde93c3ab9f47f20

Findings Breakdown



Critical	3
Medium	0
Minor / Informative	13

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	3	0	0	0
Medium	0	0	0	0
Minor / Informative	13	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	AXGT.sol#L715,803,809
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. However once the transactions are enable the owner will be able to disable them again and as a result disable the transactions for all users excluding the authorized accounts.

Additionally, the transfers can be disabled if the `balanceLimit` variable is set to a very small value, such as zero.

```
if (!isTradingEnabled) {
    require(_isExcluded[to] || _isExcluded[from], "Trading is
not yet enabled. Be patient!");
}
...
if (!_isLimitExcluded[to]){
    require((balanceOf(to) + amount) <= balanceLimit,
"maxlimit");
}

    function enableTrading() external onlyOwner {
        require(!isTradingEnabled, "already enabled");
        isTradingEnabled = true;
    }

    function disableTrading () external onlyOwner {
        require(isTradingEnabled, "already disabled");
        isTradingEnabled = false;
    }
```

Recommendation

Its is recommended to remove the `disableTrading` function from the contract and invoke the `enableTrading` function. Also 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. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	AXGT.sol#L847
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setFee` function with a high percentage value.

```
  function setFee (uint256 _newLPFee, uint256 _newadminFee,
uint256 _newfundFee) external onlyOwner{
        require (_newLPFee + _newadminFee + _newfundFee <
10000, "need to smaller than 100%");
        liquidityFee = _newLPFee;
        adminFee = _newadminFee;
        fundFee = _newfundFee;
    }
```

Recommendation

The contract could embody a check for the maximum acceptable value. 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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

ZD - Zero Division

Criticality	Critical
Location	AXGT.sol#L734,797
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. This can lead to unpredictable and potentially harmful results, such as a transaction revert.

The `liquidityFee`, `fundFee` and `adminFee` variables could be set to zero.

```
uint256 swapTokens = contractTokenBalance * (liquidityFee) /
(liquidityFee + fundFee + adminFee);
uint256 adminETH = ethBalance * adminFee / (adminFee +
fundFee);
```

Recommendation

It is important to handle division by zero appropriately in the code to avoid unintended behavior and to ensure the reliability and safety of the contract. The contract should ensure that the divisor is always non-zero before performing a division operation. It should prevent the variables to be set to zero, or should not allow the execution of the corresponding statements.

RSD - Redundant Swap Duplication

Criticality	Minor / Informative
Location	AXGT.sol#L763,795
Status	Unresolved

Description

The contract contains multiple swap methods that individually perform token swaps and transfer promotional amounts to specific addresses and features. This redundant duplication of code introduces unnecessary complexity and increases dramatically the gas consumption. By consolidating these operations into a single swap method, the contract can achieve better code readability, reduce gas costs, and improve overall efficiency.

Specifically the contract is using the `swapAndLiquify` method both within the `swapAndLiquify` and `swapAndSendDividends` functions.

```
...
swapping = true;
uint256 swapTokens = contractTokenBalance * (liquidityFee) /
(liquidityFee + fundFee + adminFee);
swapAndLiquify(swapTokens);
uint256 sellTokens = balanceOf(address(this));
swapAndSendDividends(sellTokens);
swapping = false;
...

function swapAndLiquify(uint256 tokens) private {
    uint256 half = tokens / 2;
    ...
    swapTokensForEth(half);
    ...
}

function swapAndSendDividends(uint256 tokens) private {
    swapTokensForEth(tokens);
    ...
}
```

Recommendation

A more optimized approach could be adopted to perform the token swap operation once for the total amount of tokens and distribute the proportional amounts to the corresponding addresses, eliminating the need for separate swaps.

EIS - Excessively Integer Size

Criticality	Minor / Informative
Location	AXGT.sol#L676,847
Status	Unresolved

Description

The contract is using a bigger unsigned integer data type that the maximum size that is required. By using an unsigned integer data type larger than necessary, the smart contract consumes more storage space and requires additional computational resources for calculations and operations involving these variables. This can result in higher transaction costs, longer execution times, and potential scalability bottlenecks.

Specifically the variables `liquidity` , `FeeadminFee` and `fundFee` can be represent by using a `uint16` , since the maximum possible vallue is `10000` .

```
uint256 public liquidityFee    = 133;
uint256 public adminFee        = 133;
uint256 public fundFee         = 133;

    function setFee (uint256 _newLPFee, uint256 _newadminFee,
uint256 _newfundFee) external onlyOwner{
        require (_newLPFee + _newadminFee + _newfundFee <
10000, "need to smaller than 100%");
        ...
    }
```

Recommendation

To address the inefficiency associated with using an oversized unsigned integer data type, it is recommended to accurately determine the required size based on the range of values the variable needs to represent. The team is advised to use a `uint16` data type for the variables `liquidityFee` , `adminFee` and `fundFee` .

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	AXGT.sol#L813,817,823,829,835,839,843
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

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.

```
  function setSwapAtAmount (uint256 amount) external onlyOwner
{
        swapTokensAtAmount = amount;
    }

    function setadminWallet (address _address) external
onlyOwner {
        _isExcluded[adminWallet] = false;
        adminWallet = _address;
        _isExcluded[adminWallet] = true;
    }

    function setFundWAllet (address _address) external
onlyOwner {
        _isExcluded[fundWallet] = false;
        adminWallet = _address;
        _isExcluded[fundWallet] = true;
    }

    function changeOwner (address _address) external onlyOwner{
        _isExcluded[_address] = false;
        transferOwnership(_address);
        _isExcluded[_address] = true;
    }

    function setExcludeWallet (address _address, bool _value)
external onlyOwner{
        _isExcluded[_address] = _value;
    }

    function setExcludeLimitWallet (address _address, bool
_value) external onlyOwner{
        _isLimitExcluded[_address] = _value;
    }

    function setLimit (uint256 _limit) external onlyOwner {
        balanceLimit = _limit;
    }
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	AXGT.sol#L799
Status	Unresolved

Description

The contract sends funds to the `adminWallet` and `fundWallet` addresses as part of the transfer flow. These addresses can either be a wallet address or a contract. If the addresses belong to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
payable(adminWallet).transfer(adminETH);
payable(fundWallet).transfer(fundETH);
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be achieved by not allowing set contract addresses or by sending the funds in a non-revertable way.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	AXGT.sol#L687
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
constructor()  ERC20("AxonDAO Governance Token", "AXGT") {
        IUniswapV2Router02 _UniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
        address _UniswapV2Pair =
IUniswapV2Factory(_UniswapV2Router.factory())
            .createPair(address(this),
_UniswapV2Router.WETH());
        UniswapV2Router = _UniswapV2Router;
        UniswapV2Pair = _UniswapV2Pair;
    ...
    }
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	AXGT.sol#L724,813
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTokensAtAmount` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function _transferaddress from,  address to,  uint256 amount)
internal override {
    ...
        uint256 contractTokenBalance =
balanceOf(address(this));
        bool canSwap = contractTokenBalance >=
swapTokensAtAmount;
        if(
            canSwap &&
            !swapping &&
            from != UniswapV2Pair &&
            from != owner() &&
            to != owner()
        ) {
            swapping = true;
            uint256 swapTokens = contractTokenBalance *
(liquidityFee) / (liquidityFee + fundFee + adminFee);
            swapAndLiquify(swapTokens);
            uint256 sellTokens = balanceOf(address(this));
            swapAndSendDividends(sellTokens);
            swapping = false;
        }
        ...
    }
    function setSwapAtAmount (uint256 amount) external
onlyOwner {
        swapTokensAtAmount = amount;
    }
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	AXGT.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	AXGT.sol#L686
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.

```
UniswapV2Router
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	AXGT.sol#L666,687
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
IUniswapV2Router02 public UniswapV2Router;
...
IUniswapV2Router02 _UniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	AXGT.sol#L106,107,124,160,666,667,680,677,710,813,823,829,831,835,843,847
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.

```
...
address _address
uint256 _limit
uint256 _newLPFee, uint256 _newadminFee, uint256 _newfundFee
...
```

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	AXGT.sol#L805,814,844,849
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.

```
isTradingEnabled = true;
...
swapTokensAtAmount = amount;
...
balanceLimit = _limit;
...
liquidityFee = _newLPFee;
adminFee = _newadminFee;
fundFee = _newfundFee;
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	AXGT.sol#L819,825
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
adminWallet = _address;
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	AXGT.sol#L18
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.7;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Context	Implementation			
	_msgSender	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner

IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-

	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-

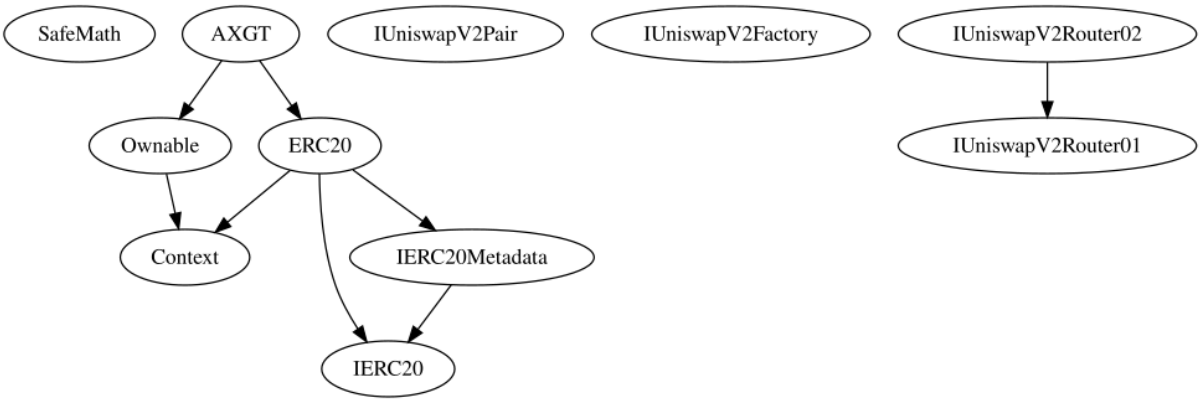
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2 Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-

IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-

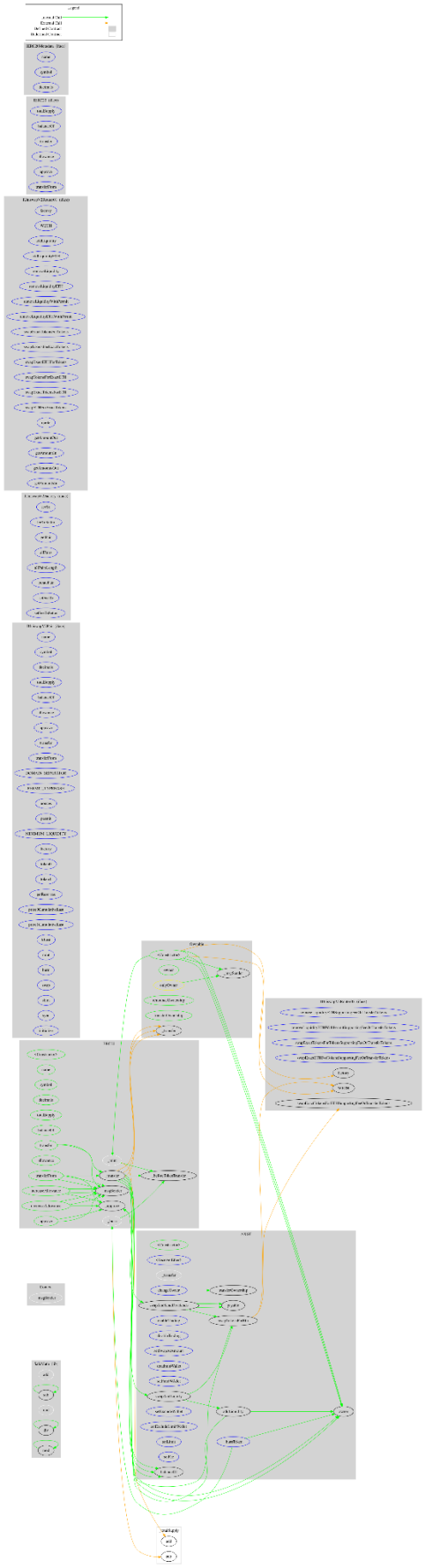
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
AXGT	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	_burnToken	External	✓	onlyOwner
	_transfer	Internal	✓	
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapAndSendDividends	Private	✓	
	enableTrading	External	✓	onlyOwner
	disableTrading	External	✓	onlyOwner
	setSwapAtAmount	External	✓	onlyOwner
	setadminWallet	External	✓	onlyOwner

	setFundWAllet	External	✓	onlyOwner
	changeOwner	External	✓	onlyOwner
	setExcludeWallet	External	✓	onlyOwner
	setExcludeLimitWallet	External	✓	onlyOwner
	setLimit	External	✓	onlyOwner
	setFee	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

Axondao contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions and manipulate the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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