Cyberscope

Audit Report

AI-GPT

February 2023

Type	ERC20
Network	ARBITRUM
Address	0x2a9b6b293375e30fb5050b9c5d2c588918d50570
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Analysis 3
ST - Stops Transactions 4
Description 4
Recommendation 4
Diagnostics 5
PTRP - Potential Transfer Revert Propagation 6
Description 6
Recommendation 6
PVC - Price Volatility Concern 7
Description 7
Recommendation 7
RSML - Redundant SafeMath Library 8
Description 8
Recommendation 8
L04 - Conformance to Solidity Naming Conventions 9
Description 9
Recommendation 10
L14 - Uninitialized Variables in Local Scope 11
Description 11
Recommendation 11
L16 - Validate Variable Setters 12
Description 12
Recommendation 12
L19 - Stable Compiler Version 13
Description 13
Recommendation 13
Functions Analysis 14
Inheritance Graph 17
Flow Graph 18
Summary 19
Disclaimer 20
About Cyberscope 21

Review

Contract Name	AIGPT
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://arbiscan.io/address/0x2a9b6b293375e30fb5050b9c5d2c588918d50570
Address	0x2a9b6b293375e30fb5050b9c5d2c588918d50570
Network	ARBITRUM
Symbol	AG
Decimals	9
Total Supply	1,000,000,000

Audit Updates

Initial Audit	27 Feb 2023

Source Files

Filename	SHA256
AIGPT.sol	a21e2743c9793847c3d66d7a2fd423b64885b1b135c5e6938c70d0070e02bebc

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

ST - Stops Transactions

Criticality	Medium
Location	AIGPT.sol#L425,447
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting either the `_txLimitAmount` or `_walletLimitAmount` to zero.

```
if(!isTxLimitExempt[sender] && !isTxLimitExempt[recipient]) {
    require(smallOrEqual(amount, _txLimitAmount));
}

...

if(walletLimitEnable && !isWalletLimitExempt[recipient])
    require(smallOrEqual(balanceOf(recipient).add(finalAmount),
_walletLimitAmount));
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. 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.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	PTRP	Potential Transfer Revert Propagation	Unresolved
	PVC	Price Volatility Concern	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L14	Uninitialized Variables in Local Scope	Unresolved
	L16	Validate Variable Setters	Unresolved
	L19	Stable Compiler Version	Unresolved

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	AIGPT.sol#L481
Status	Unresolved

Description

The contract sends funds to a `marketingWallet` as part of the transfer flow. This address can either be a wallet address or a contract. If the address belongs 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.

```
if(amountBNBMarketing > 0)
    transferToAddressETH(MarketingWallet, amountBNBMarketing);

if(amountBNBTeam > 0)
    transferToAddressETH(TreasuryWallet, amountBNBTeam);
```

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 archived by not allowing set contract addresses or by sending the funds in a non-revertable way.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	AIGPT.sol#L319
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 minimumTokensBeforeSwapAmount() public view returns (uint256) {
    return minimumTokensBeforeSwap;
}
```

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	AIGPT.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 on 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 unnecessarily the gas consumption.

```
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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	AIGPT.sol#L126,154,155,158,167,168,169,171,172,173,175,176,177,179,180,181,184,185,379
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.

```
function WETH() external pure returns (address);
address payable public MarketingWallet
address payable public TreasuryWallet
mapping (address => uint256) _balances
uint256 public _buyLiquidityFee
uint256 public _buyMarketingFee
uint256 public _buyTeamFee
uint256 public _sellLiquidityFee
uint256 public _sellMarketingFee
uint256 public _sellTeamFee
uint256 public _liquidityShare
uint256 public _marketingShare
uint256 public _teamShare
uint256 public _totalTaxIfBuying

...
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	AIGPT.sol#L346
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 i
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	AIGPT.sol#L372,376
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.

```
MarketingWallet = payable(newAddress)
TreasuryWallet = payable(newAddress)
```

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	AIGPT.sol#L12
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 ^0.8.0;
```

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.

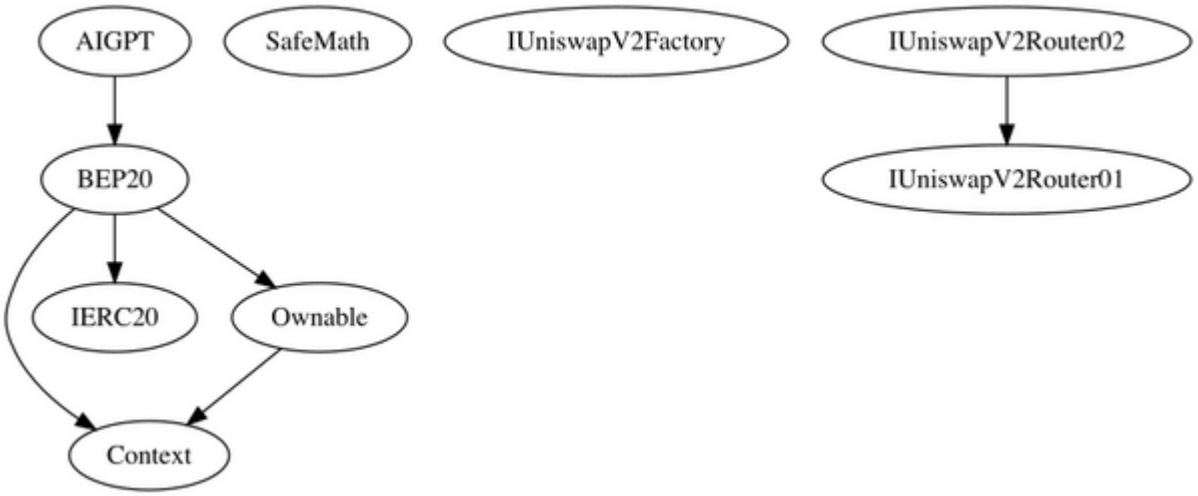
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Ownable	Implementation	Context		
		Public	✓	-

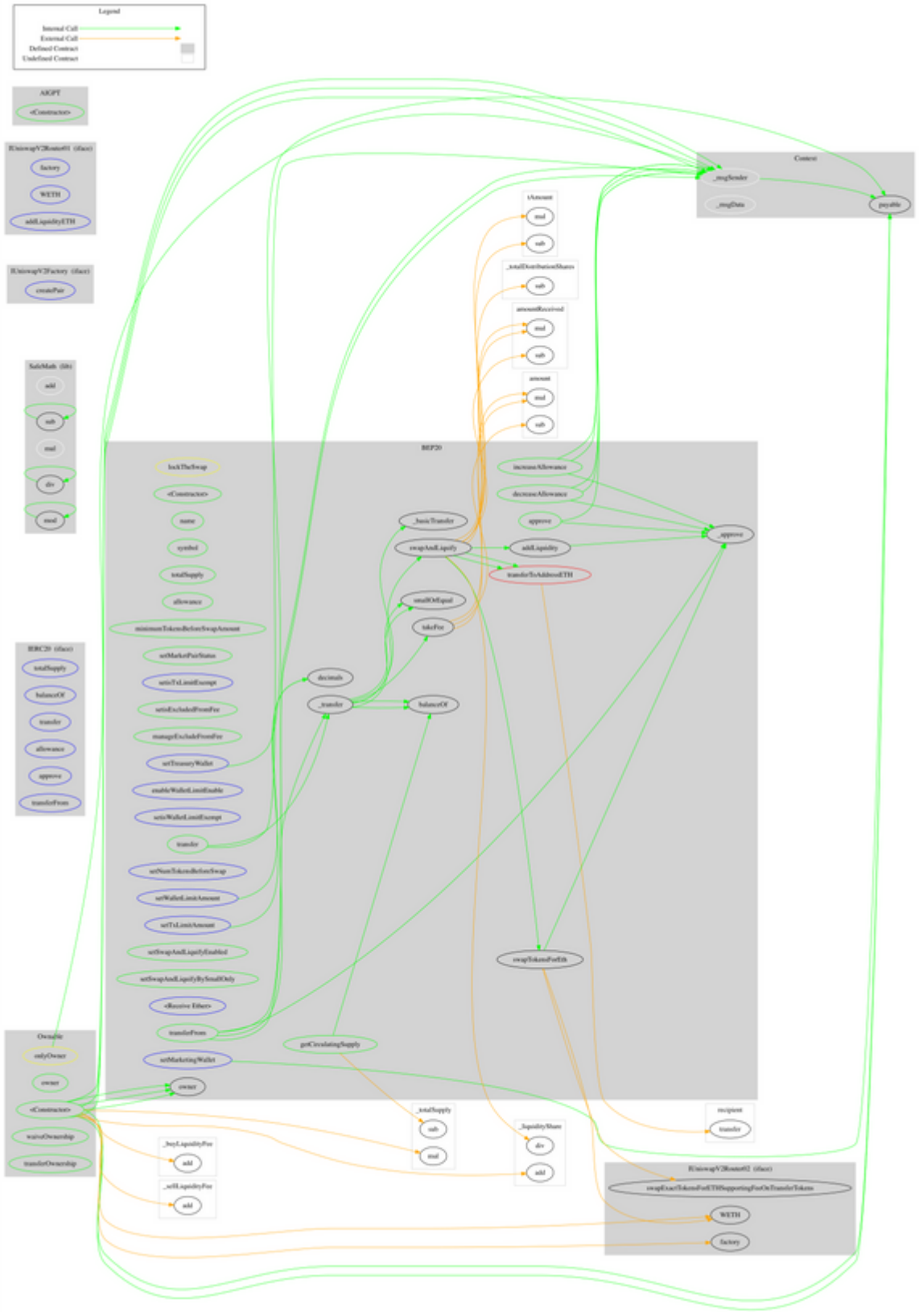
	owner	Public		-
	waiveOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IUniswapV2Factory	Interface			
	createPair	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
BEP20	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	minimumTokensBeforeSwapAmount	Public		-
	approve	Public	✓	-

	_approve	Private	✓	
	setMarketPairStatus	Public	✓	onlyOwner
	setisTxLimitExempt	External	✓	onlyOwner
	setisExcludedFromFee	Public	✓	onlyOwner
	manageExcludeFromFee	Public	✓	onlyOwner
	setTxLimitAmount	External	✓	onlyOwner
	enableWalletLimitEnable	External	✓	onlyOwner
	setisWalletLimitExempt	External	✓	onlyOwner
	setWalletLimitAmount	External	✓	onlyOwner
	setNumTokensBeforeSwap	External	✓	onlyOwner
	setMarketingWallet	External	✓	onlyOwner
	setTreasuryWallet	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
	setSwapAndLiquifyBySmallOnly	Public	✓	onlyOwner
	getCirculatingSupply	Public		-
	transferToAddressETH	Private	✓	
		External	Payable	-
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Private	✓	
	smallOrEqual	Public		-
	_basicTransfer	Internal	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	takeFee	Internal	✓	
AIGPT	Implementation	BEP20		
		Public	✓	BEP20

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
AIGPT
<Constructor>
IUniswapV2Router01 (iface)
factory
WETH
addLiquidityETH
IUniswapV2Factory (iface)
createPair
SafeMath (lib)
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
onlyOwner
owner
<Constructor>
waiveOwnership
transferOwnership
Context
_msgSender
_msgData
payable
tAmount
mul
sub
_totalDistributionShares
sub
amountReceived
mul
sub
amount
mul
sub
BEP20
lockTheSwap
<Constructor>
name
symbol
totalSupply
allowance
minimumTokensBeforeSwapAmount
setMarketPairStatus
setIsTxLimitExempt
setIsExcludedFromFee
manageExcludeFromFee
setTreasuryWallet
enableWalletLimitEnable
setIsWalletLimitExempt
transfer
setNumTokensBeforeSwap
setWalletLimitAmount
setTxLimitAmount
setSwapAndLiquifyEnabled
setSwapAndLiquifyBySmallOnly
<Receive Ethers>
transferFrom
setMarketingWallet
owner
_basicTransfer
swapAndLiquify
addLiquidity
transferToAddressETH
takeFee
decimals
_transfer
balanceOf
increaseAllowance
decreaseAllowance
approve
_approve
swapTokensForEth
getCirculatingSupply
_totalSupply
sub
mul
_buyLiquidityFee
add
_sellLiquidityFee
add
_liquidityShare
div
add
recipient
transfer
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory

Summary

AI-GPT 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. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. The fees are fixed t0 3%.

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