Cyberscope

Audit Report

Golden Inu

August 2023

Network ETH

Address 0xd87996ff3d06858bfc20989aef50cc5fcd4d84ca

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	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	EPC	Existing Pair Creation	Unresolved
	PVC	Price Volatility Concern	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L14	Uninitialized Variables in Local Scope	Unresolved
	L16	Validate Variable Setters	Unresolved
	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	StandardTokenWithAntibot
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization	200 runs
Explorer	https://etherscan.io/address/0xd87996ff3d06858bfc20989aef50cc5fcd4d84ca
Address	0xd87996ff3d06858bfc20989aef50cc5fcd4d84ca
Network	ETH
Symbol	GOLDEN
Decimals	9
Total Supply	100,000,000,000,000,000

Audit Updates

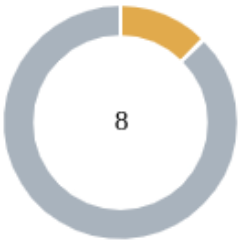
Initial Audit	20 Aug 2023

Source Files

Filename	SHA256
contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol	fa4a33be231fee1dcc2757bb898a2927f16425d25f9257a2f1073c03e801b0f0
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/token/ERC20/IERC20.sol	c2b06bb4572bb4f84bfc5477dadc0fcc497cb66c3a1bd53480e68bedc2e154a6

@openzeppelin/contracts/token/ERC20/ERC20.sol	f7831910f2ed6d32acff6431e5998baf50e4a00121303b27e974aab0ec637d79
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	1	0	0	0
Minor / Informative	7	0	0	0

ST - Stops Transactions

Criticality	Medium
Location	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L710,716
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the `isExcludedFromMaxTransactionAmount` addresses. The owner may take advantage of it by setting the `maxTransactionAmount` or the `maxWallet` to zero. As a result, the contract may operate as a honeypot.

```
 if (!isExcludedFromMaxTransactionAmount[from]) {
      require(
          amount < maxTransactionAmount,
          "ERC20: exceeds transfer limit"
      );
  }
  if (!isExcludedFromMaxTransactionAmount[to]) {
      require(
          balanceOf(to) < maxWallet,
          "ERC20: exceeds max wallet limit"
      );
  }

   function updateMaxWallet(uint256 _maxWallet) external
onlyOwner {
        require(_maxWallet>0, "maxWallet > 0");
        emit UpdateMaxWallet(_maxWallet, maxWallet);
        maxWallet = _maxWallet;
    }

    function updateMaxTransactionAmount(uint256
_maxTransactionAmount)
        external
        onlyOwner
    {
        require(_maxTransactionAmount>0, "maxTransactionAmount
> 0");
        emit UpdateMaxTransactionAmount(_maxTransactionAmount,
maxTransactionAmount);
        maxTransactionAmount = _maxTransactionAmount;
    }
```

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.

EPC - Existing Pair Creation

Criticality	Minor / Informative
Location	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L527
Status	Unresolved

Description

The contract is using the `updateUniswapV2Router` function to update Uniswap V2 Router and create a new pair on the exchange. However, the function does not handle the scenario where a pair already exists prior to its execution. If a pair for the given tokens has already been established, the createPair function will revert and not proceed with the creation of a new pair. As a result, if a pair has been previously set up before the `updateUniswapV2Router` function is invoked, the contract will encounter an error when trying to call the createPair function within the `updateUniswapV2Router` function. This will prevent the successful execution of the `updateUniswapV2Router` function, effectively locking the contract to the old pair and preventing it from updating to the new pair address.

```
    function updateUniswapV2Router(address newAddress) public
onlyOwner {
        require(
            newAddress != address(mainRouter),
            "The router already has that address"
        );
        emit UpdateUniswapV2Router(newAddress,
address(mainRouter));
        mainRouter = IUniswapV2Router02(newAddress);
        address _mainPair =
IUniswapV2Factory(mainRouter.factory()).createPair(
            address(this),
            baseTokenForPair
        );
        mainPair = _mainPair;
        _setAutomatedMarketMakerPair(mainPair, true);
    }
```

Recommendation

To mitigate the risks associated with the absence of address validation in the pair address argument, it is recommended to implement comprehensive address validation mechanisms. A recommended approach could be to verify pair existence in the decentralized application. Prior to interacting with the pair address contract, perform checks to verify the existence and validity of the contract at the provided address. This can be achieved by querying the Ethereum blockchain or utilizing external libraries that provide contract verification services.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L614,672
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `minAmountToTakeFee` 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 _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        ...
        bool overMinimumTokenBalance = contractTokenBalance >=
            minAmountToTakeFee;

        // Take Fee
        if (
            !inSwapAndLiquify &&
            overMinimumTokenBalance &&
            automatedMarketMakerPairs[to]
        ) {
            takeFee();
        }
    ...
    }

    function updateMinAmountToTakeFee(uint256
_minAmountToTakeFee)
        external
        onlyOwner
    {
        require(_minAmountToTakeFee > 0, "minAmountToTakeFee >
0");
        emit UpdateMinAmountToTakeFee(_minAmountToTakeFee,
minAmountToTakeFee);
        minAmountToTakeFee = _minAmountToTakeFee;
    }
```

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 exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L456,463
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.

```
_decimals
gemAntiBot
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L10,246,248,279,351,518,542,548,565,566,584,585,603,604,614
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);
function DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint256);
function INIT_CODE_PAIR_HASH() external view returns (bytes32);
address _baseTokenForPair
uint256 _maxWallet
uint256 _maxTransactionAmount
uint16 _sellLiquidityFee
uint16 _buyLiquidityFee
uint16 _sellMarketingFee
uint16 _buyMarketingFee
address _marketingWallet
bool _isMarketingFeeBaseToken

...
```

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	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L683,684
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 _liquidityFee
uint256 _marketingFee
```

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	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L519,538
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.

```
baseTokenForPair = _baseTokenForPair
mainPair = _mainPair
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/Tokens/StandardTokens/StandardTokenWithAntibot.sol#L753
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(baseTokenForPair).transfer(
                        marketingWallet,
                        baseTokenForMarketing
                    )
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
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	✓	-
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		-

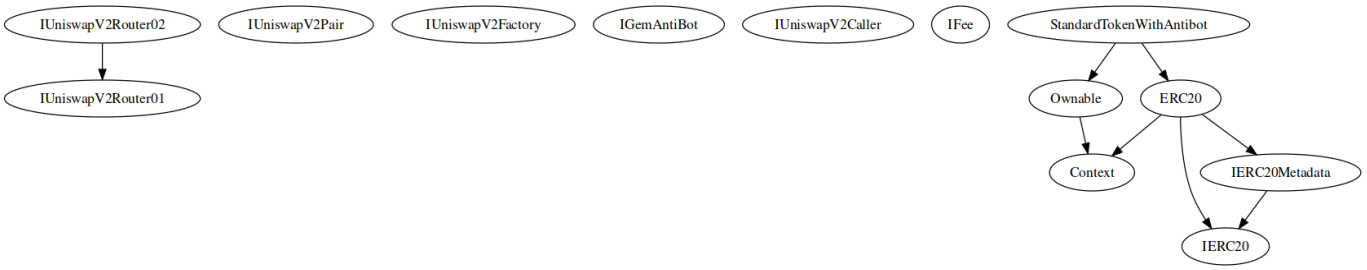
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
	INIT_CODE_PAIR_HASH	External		-
IGemAntiBot	Interface			
	setTokenOwner	External	✓	-
	onPreTransferCheck	External	✓	-
IUniswapV2Caller	Interface			
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
IFee	Interface			
	payFee	External	Payable	-
StandardTokenWithAntibot	Implementation	ERC20, Ownable		
		Public	Payable	ERC20
	decimals	Public		-
	setUsingAntiBot	External	✓	onlyOwner
	updateUniswapV2Pair	External	✓	onlyOwner
	updateUniswapV2Router	Public	✓	onlyOwner
	updateMaxWallet	External	✓	onlyOwner
	updateMaxTransactionAmount	External	✓	onlyOwner

	updateLiquidityFee	External	✓	onlyOwner
	updateMarketingFee	External	✓	onlyOwner
	updateMarketingWallet	External	✓	onlyOwner
	updateMinAmountToTakeFee	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	excludeFromFee	External	✓	onlyOwner
	excludeFromMaxTransactionAmount	External	✓	onlyOwner
	_transfer	Internal	✓	
	takeFee	Private	✓	lockTheSwap
	swapTokensForBaseToken	Private	✓	
	addLiquidity	Private	✓	
		External	Payable	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-

	approve	External	✓	-
	transferFrom	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	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	

	_afterTokenTransfer	Internal	✓	
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

Inheritance Graph



Flow Graph

Summary

Golden Inu 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. There is also a limit of max 20% fees.

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