Cyberscope

Audit Report

Golduck Token

March 2023

Network	ETH
Address	0x366A07A2164e627E4994FE1F8d97cec4087a65B2
Audited by	© cyberscope

Table of Contents

Review

Contract Name	GolduckDAOToken
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization	200 runs
Explorer	https://etherscan.io/address/0x366a07a2164e627e4994fe1f8d97cec4087a65b2
Address	0x366a07a2164e627e4994fe1f8d97cec4087a65b2
Network	ETH
Decimals	18

Audit Updates

Initial Audit	27 Mar 2023

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	da66c17044345dc892d85bd7ddc9745d25df0b3dacfba8f84eb87c60d6e40fe3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	cd823c76cbf5f5b6ef1bda565d58be66c843c37707cd93eb8fb5425deebd6756
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	36a6477c6263d9441dab59861e0ca97a201caf2843598af2a8e04e897a738c2f
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	68bcca423fc72ec9625e219c9e36306c726a347e43f3711467c579bd3f6500c8
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	4e09a7479aa3e7c313f8fc141c4c8fc04e0abfeb8754615ef7d78ec94c298b07
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	35fb271561f3dc72e91b3a42c6e40c2bb2e788cd8ca58014ac43f6198b8d32ca
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
contracts/RPD_V1/GolduckDAOToken.sol	d6f3d398b92f13b017a7edf2c253c4b6eedaf8a7182f51e103f0236d661f8c05
contracts/RPD_V1/interfaces/IRewardDistributor.sol	babacd04aa2e629f7c51178dee6587babac3189ee6adc8590bd1aa8867c39e1a
contracts/RPD_V1/interfaces/IRewardPool.sol	d3b22fba6f8d5355828bd836f690ad84fd35792537067ed8273db91153cb5d98

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

Severity	Code	Description	Status
🔵	ST	Stops Transactions	Passed
🔵	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	Resolved
🔵	BT	Burns Tokens	Passed
🔵	BC	Blacklists Addresses	Passed

MT - Mints Tokens

Criticality	Minor / Informative
Location	contracts/RPD_V1/GolduckDAOToken.sol#L42
Status	Resolved

Description

The contract users have the authority to mint tokens. The users may take advantage of it by calling the `config` function. As a result, the contract tokens will be highly inflated.

```
function config() external {
    require(!_config, "Already called");

    _config = true;
    isRewardEnabled = false;
    _burn(msg.sender,balanceOf(msg.sender));
    _mint(msg.sender, 100000000 * (10 ** 18));
    isRewardEnabled = true;
}
```

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. 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.

Team Update

Currently, the contract is pointed by the `0xe585e1878856868d9657aab815a8b8ba6a7a960d` address that works as a proxy contract.

The proxy contract has executed the config method, hence in the proxy's state, it cannot be executed again.

For reference, the transaction URL associated with the function's execution is: https://etherscan.io/tx/0x1f8a05399e9b10bcf55f7d258e57e5c29bf88013c500e0dbba10bd3e1f74d179.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	FPP	Function Public Permissions	Resolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L22	Potential Locked Ether	Unresolved

FPP - Function Public Permissions

Criticality	Minor / Informative
Location	contracts/RPD_V1/GolduckDAOToken.sol#L42
Status	Resolved

Description

The `config` function allows users to mint tokens to their balance, with the amount being equal to the total supply. The function is marked as external and can be accessed by any user. As a result, this can lead to inflation and devalue the token's worth.

```
function config() external {
    require(!_config, "Already called");

    _config = true;
    isRewardEnabled = false;
    _burn(msg.sender,balanceOf(msg.sender));
    _mint(msg.sender, 100000000 * (10 ** 18));
    isRewardEnabled = true;
}
```

Recommendation

The team is advised to add proper access controls and checks to prevent such vulnerabilities and ensure the security of the contract.

Team Update

Currently, the contract is pointed by the `0xe585e1878856868d9657aab815a8b8ba6a7a960d` address that works as a proxy contract.

The proxy contract has executed the config method, hence in the proxy's state, it cannot be executed again.

For reference, the transaction URL associated with the function's execution is: https://etherscan.io/tx/0x1f8a05399e9b10bcf55f7d258e57e5c29bf88013c500e0dbba10bd3e1f74d179.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/RPD_V1/GolduckDAOToken.sol#L16
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 _rewardPool
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, and 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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/RPD_V1/GolduckDAOToken.sol#L2
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 ^0.8.4;
```

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.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	contracts/RPD_V1/GolduckDAOToken.sol#L25
Status	Unresolved

Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
receive() external payable {}
```

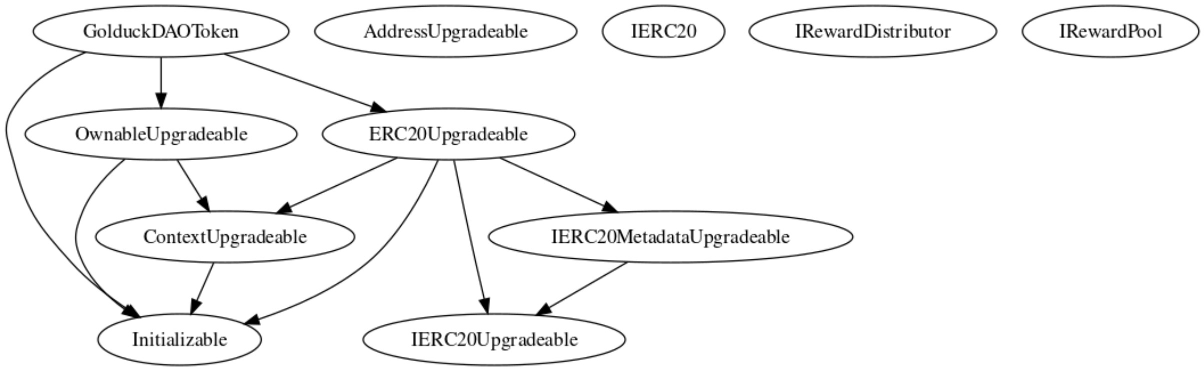
Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

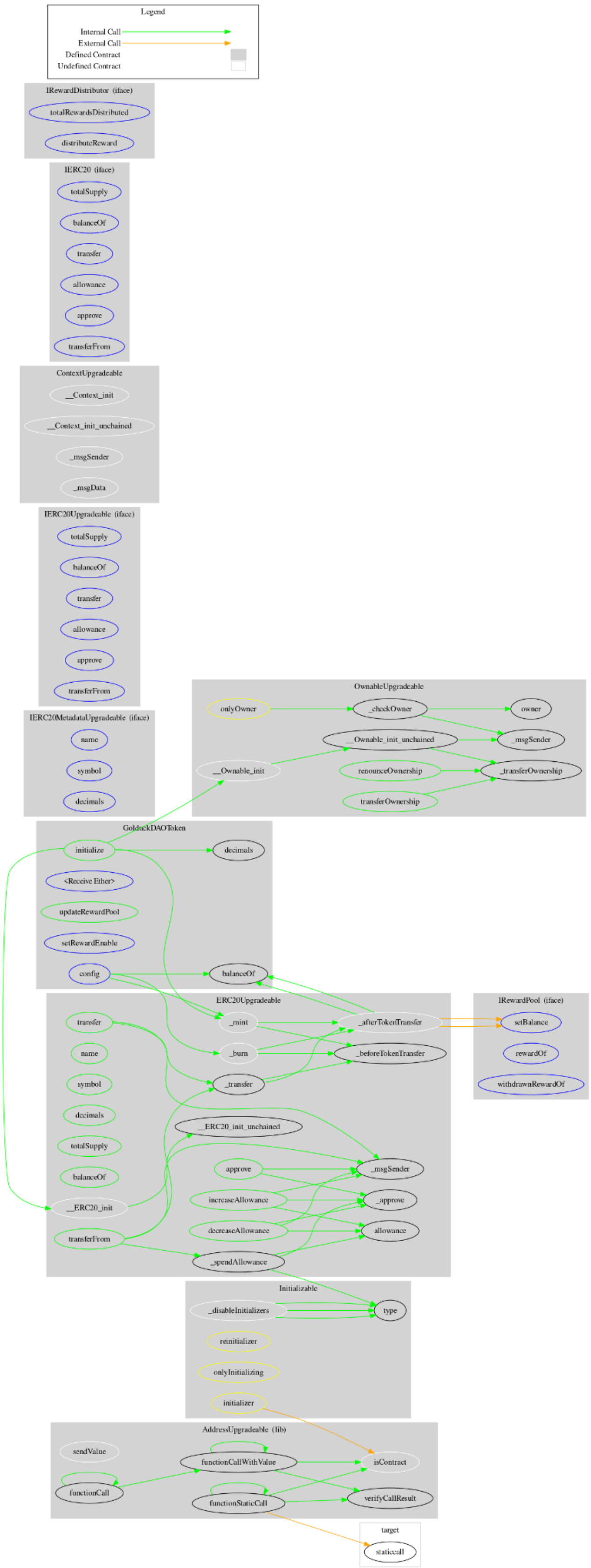
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
GolduckDAOToken	Implementation	Initializable, ERC20Upgradeable, OwnableUpgradeable		
	initialize	Public	✓	initializer
		External	Payable	-
	updateRewardPool	Public	✓	onlyOwner
	setRewardEnable	External	✓	onlyOwner
	_afterTokenTransfer	Internal	✓	
	config	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IRewardDistributor (iface)
totalRewardsDistributed
distributeReward
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
ContextUpgradeable
__Context_init
__Context_init_unchained
_msgSender
_msgData
IERC20Upgradeable (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
OwnableUpgradeable
onlyOwner
_checkOwner
owner
__Ownable_init_unchained
_msgSender
__Ownable_init
renounceOwnership
_transferOwnership
transferOwnership
IERC20MetadataUpgradeable (iface)
name
symbol
decimals
GolduckDAOToken
initialize
decimals
<Receive Ether>
updateRewardPool
setRewardEnable
config
balanceOf
ERC20Upgradeable
transfer
_mint
_afterTokenTransfer
name
_burn
_beforeTokenTransfer
symbol
_transfer
decimals
__ERC20_init_unchained
totalSupply
approve
_msgSender
balanceOf
increaseAllowance
_approve
__ERC20_init
decreaseAllowance
allowance
transferFrom
_spendAllowance
IRewardPool (iface)
setBalance
rewardOf
withdrawnRewardOf
Initializable
_disableInitializers
type
reinitializer
onlyInitializing
initializer
AddressUpgradeable (lib)
sendValue
functionCallWithValue
isContract
functionCall
functionStaticCall
verifyCallResult
target
staticcall

Summary

Golduck contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

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