Cyberscope

Audit Report

OpenGames

February 2023

Commit a1b0e5af45f485507236f462e3f3f412f7a824f6

Audited by © cyberscope

Table of Contents

Review

Repository	https://github.com/ammagtech/OGB-ICO-SmartContract
Commit	a1b0e5af45f485507236f462e3f3f412f7a824f6

Audit Updates

Initial Audit	10 Feb 2023

Source Files

Filename	SHA256
OGBToken.sol	e779c4916224c6e26912fdd1de41b2ce254a5e7e3112ef18cf362694bdbd1eec

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	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CTD	Contract's Tokens Drain	Unresolved
●	CO	Code Optimization	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

CTD - Contract's Tokens Drain

Criticality	Critical
Location	OGBToken.sol#L22
Status	Unresolved

Description

The contract users have the ability to claim all the balance of the contract. The users may take advantage of it by calling the `distribute` function.

```
function distribute(address _devContract) public {
    //2.5%
    uint256 devPercentage = 25 * 10**17;

    //2.5% of totalSupply (1 Billion)
    uint256 _devAmount = (totalSupply().mul(devPercentage)) / 10**20;

    //Approve devContract to send 2.5% of 1 Billion
    IERC20(address(this)).approve(_devContract, _devAmount);

    //2.5% send to devContract and start slicing
    IDevSupply(_devContract).deposit(_devAmount, address(this));

    IERC20(address(this)).transfer(
      msg.sender,
      IERC20(address(this)).balanceOf(address(this))
    );
}
```

Recommendation

The team is advised to revisit the implementation of the `distribute()` function and limit the ability to call the function to a specific address, such as the contract owner. This can be achieved by adding an access control mechanism, such as a modifier, that checks the caller's address before executing the function.

CO - Code Optimization

Criticality	Minor / Informative
Location	OGBToken.sol#L30,35
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 extends the `ERC20` contract, so the `transfer()` and `approve()` functions are already implemented. That means there is not reason to use `IERC20` interface to get access to these function.

```
IERC20(address(this)).approve(_devContract, _devAmount);
...
IERC20(address(this)).transfer(
    msg.sender,
    IERC20(address(this)).balanceOf(address(this))
);
```

Recommendation

The team is advised to take into consideration these segments 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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	OGBToken.sol#L12,22,41
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 _OGBAddress
address _devContract
uint256 _amount
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	OGBToken.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 ^0.8.7;
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	OGBToken.sol#L35
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(address(this)).transfer(
      msg.sender,
      IERC20(address(this)).balanceOf(address(this))
    )
```

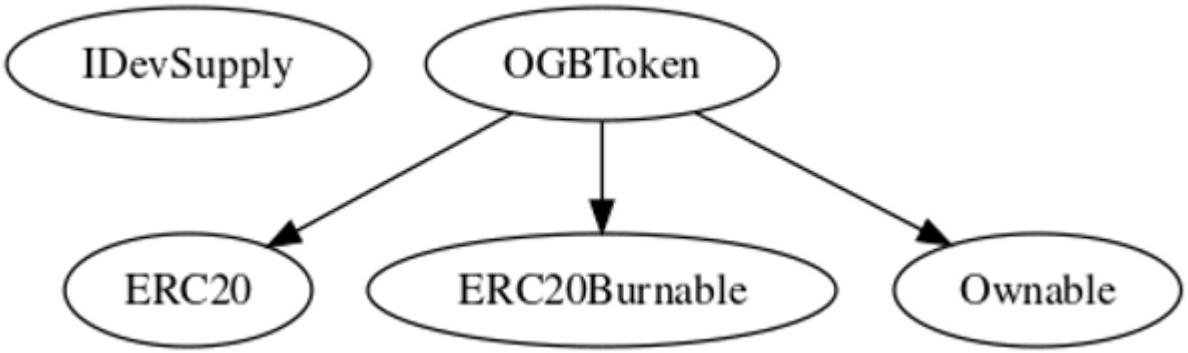
Recommendation

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.

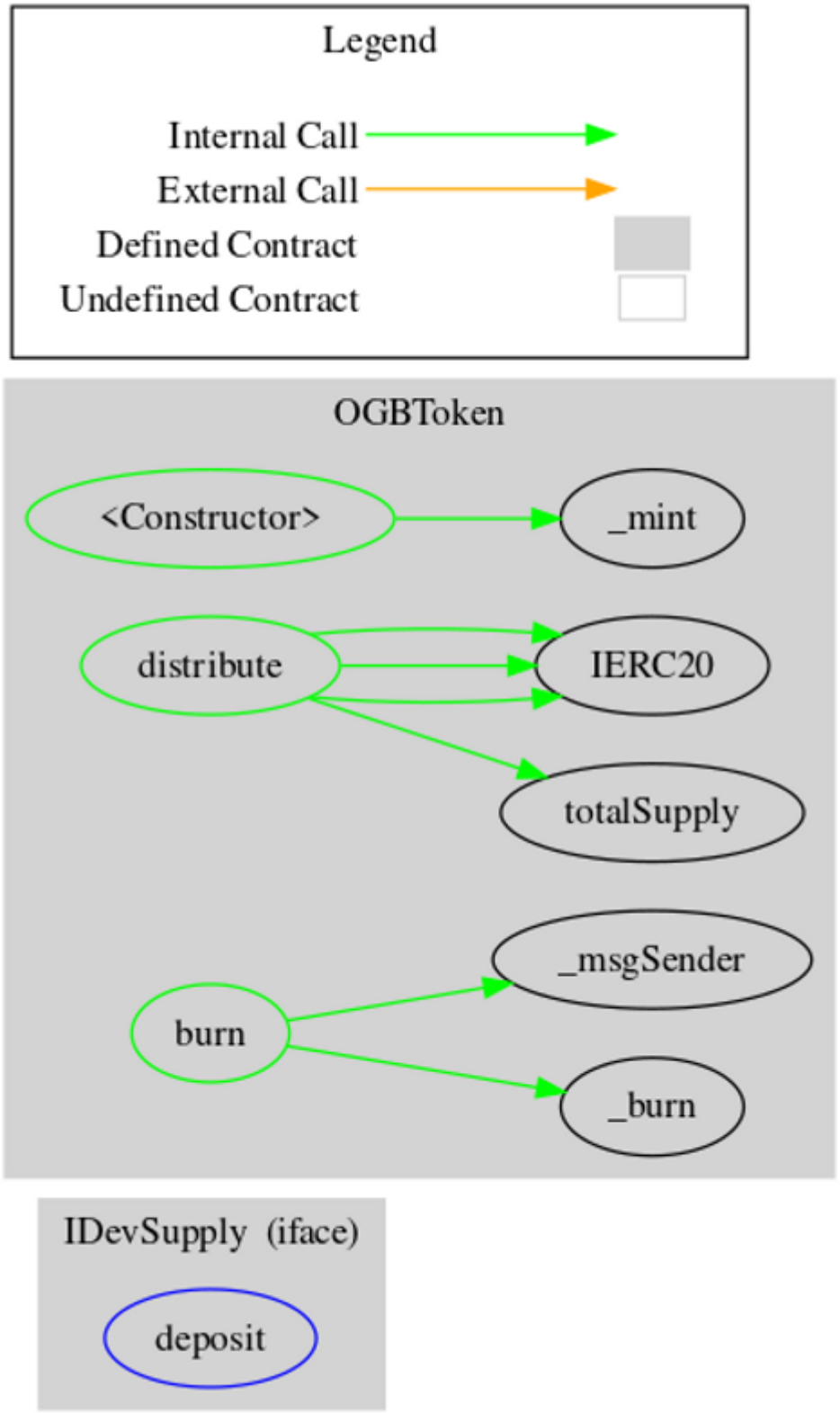
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IDevSupply	Interface			
	deposit	External	✓	-
OGBToken	Implementation	ERC20, ERC20Burnable, Ownable		
		Public	✓	ERC20
	distribute	Public	✓	-
	burn	Public	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
OGBToken
<Constructor>
_mint
distribute
IERC20
totalSupply
_msgSender
burn
_burn
IDevSupply (iface)
deposit

Summary

OpenGames is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors and one critical issue. As described at the Diagnostics section, users can drain the contract's balance by calling the distribute() function.

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