auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Global lottery

DATED : 18 March, 2024

MANUAL TESTING

Centralization – The owner can Pause the token.

Severity: High

Function: pause

Status: Open

Overview:

The owner can pause the token for an unlimited period of time which can lock the user's token.

```
function pause() public onlyOwner {
        _pause();
    };
```

Suggestion:

It is recommended that there should be a locking period.

AUDIT SUMMARY

Project name – Global lottery

Date: 18 March, 2024

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Passed With High Risk

Note: The minting will be possible in the contract but not more than the max total supply which is mentioned in the contract i.e; 10000000000

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	0	0	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0x7cf119b909583125be43c8a8514996e890a6585a#code

Token Information

Token Name : Global lottery

Token Symbol: GLOT

Decimals: 11

Token Supply: 10000000000

Network: BscScan

Token Type: BEP-20

Token Address:
0x59DC4965BdA44B13dEA7F4e6aD5FCb78DC7231eF

Checksum:
Ae1c3a4fbb6e83e8393a57617b5a5b221

Owner:
0x9E572e320f8B5Dd2305A4e7a3A7c4F7d0F8c09cb
(at time of writing the audit)

Deployer:
0x9E572e320f8B5Dd2305A4e7a3A7c4F7d0F8c09cb

TOKEN OVERVIEW

Fees:

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fees Privilege: Owner

Ownership: Owned

Minting: Yes

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

AUDIT METHODOLOGY

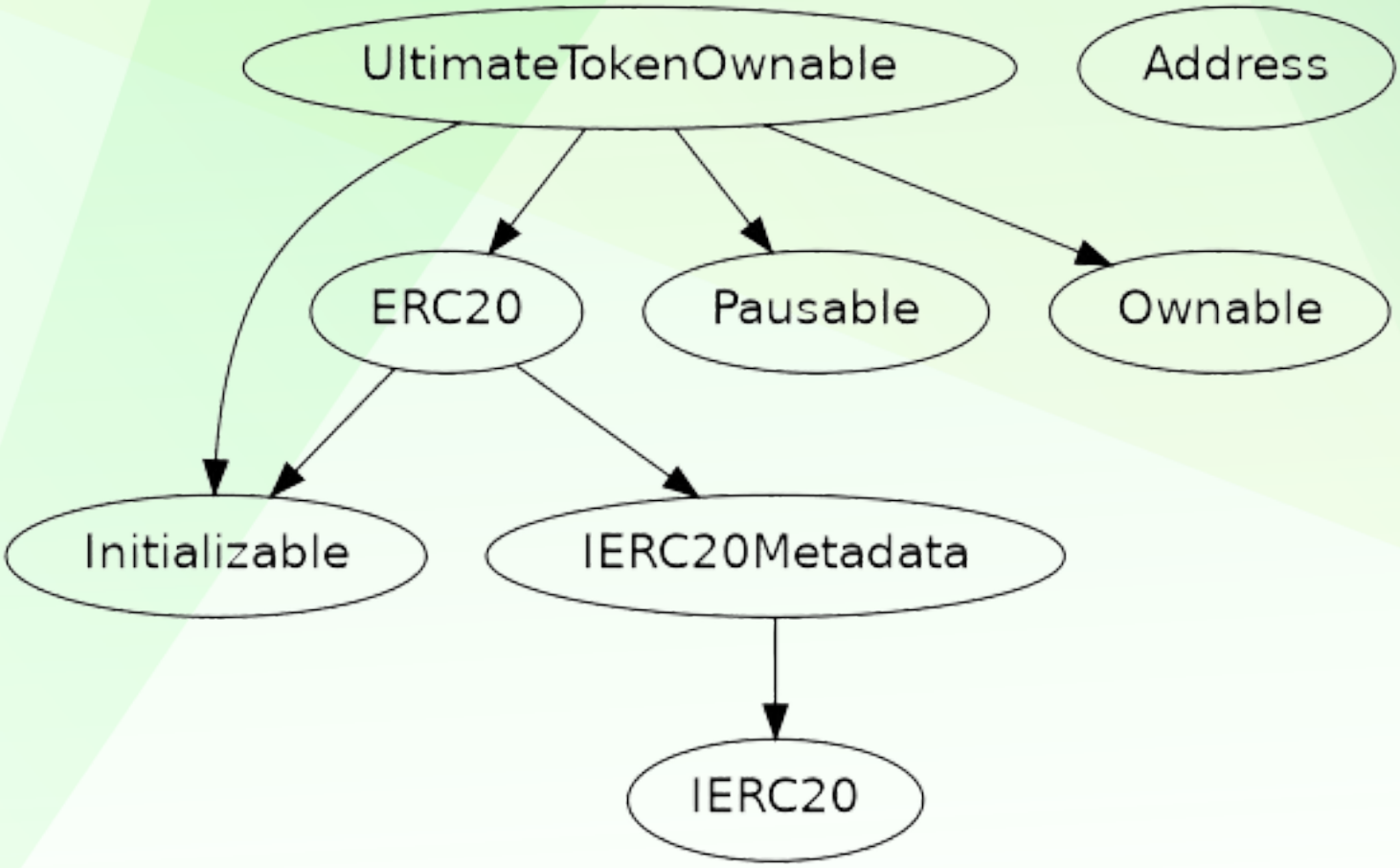
The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

INHERITANCE TREE



STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._owner (UltimateTokenOwnable.sol#775) shadows:
        - Ownable._owner (UltimateTokenOwnable.sol#112) (state variable)
UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._name (UltimateTokenOwnable.sol#776) shadows:
        - ERC20._name (UltimateTokenOwnable.sol#605) (state variable)
UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._symbol (UltimateTokenOwnable.sol#777) shadows:
        - ERC20._symbol (UltimateTokenOwnable.sol#606) (state variable)
UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._decimals (UltimateTokenOwnable.sol#778) shadows:
        - ERC20._decimals (UltimateTokenOwnable.sol#607) (state variable)
UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._maxSupply (UltimateTokenOwnable.sol#780) shadows:
        - ERC20._maxSupply (UltimateTokenOwnable.sol#603) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address._revert(bytes,string) (UltimateTokenOwnable.sol#372-384) uses assembly
        - INLINE ASM (UltimateTokenOwnable.sol#377-380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._revert(bytes,string) (UltimateTokenOwnable.sol#372-384) is never used and should be removed
Address.functionCall(address,bytes) (UltimateTokenOwnable.sol#230-232) is never used and should be removed
Address.functionCall(address,bytes,string) (UltimateTokenOwnable.sol#240-246) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (UltimateTokenOwnable.sol#259-261) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (UltimateTokenOwnable.sol#269-278) is never used and should be removed
Address.functionDelegateCall(address,bytes) (UltimateTokenOwnable.sol#311-313) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (UltimateTokenOwnable.sol#321-328) is never used and should be removed
Address.functionStaticCall(address,bytes) (UltimateTokenOwnable.sol#286-288) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (UltimateTokenOwnable.sol#296-303) is never used and should be removed
Address.sendValue(address,uint256) (UltimateTokenOwnable.sol#205-210) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (UltimateTokenOwnable.sol#360-370) is never used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (UltimateTokenOwnable.sol#336-352) is never used and should be removed
Initializable._getInitializedVersion() (UltimateTokenOwnable.sol#561-563) is never used and should be removed
Initializable._isInitializing() (UltimateTokenOwnable.sol#568-570) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.19 (UltimateTokenOwnable.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (UltimateTokenOwnable.sol#205-210):
        - (success) = recipient.call{value: amount}() (UltimateTokenOwnable.sol#208)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (UltimateTokenOwnable.sol#269-278):
        - (success,returndata) = target.call{value: value}(data) (UltimateTokenOwnable.sol#276)
Low level call in Address.functionStaticCall(address,bytes,string) (UltimateTokenOwnable.sol#296-303):
        - (success,returndata) = target.staticcall(data) (UltimateTokenOwnable.sol#301)
Low level call in Address.functionDelegateCall(address,bytes,string) (UltimateTokenOwnable.sol#321-328):
        - (success,returndata) = target.delegatecall(data) (UltimateTokenOwnable.sol#326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

STATIC ANALYSIS

```
INFO:Detectors:
Pragma version^0.8.19 (UltimateTokenOwnable.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (UltimateTokenOwnable.sol#205-210):
        - (success) = recipient.call{value: amount}() (UltimateTokenOwnable.sol#208)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (UltimateTokenOwnable.sol#269-278):
        - (success,returndata) = target.call{value: value}(data) (UltimateTokenOwnable.sol#276)
Low level call in Address.functionStaticCall(address,bytes,string) (UltimateTokenOwnable.sol#296-303):
        - (success,returndata) = target.staticcall(data) (UltimateTokenOwnable.sol#301)
Low level call in Address.functionDelegateCall(address,bytes,string) (UltimateTokenOwnable.sol#321-328):
        - (success,returndata) = target.delegatecall(data) (UltimateTokenOwnable.sol#326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._owner (UltimateTokenOwnable.sol#775) is not in mixedCase
Parameter UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._name (UltimateTokenOwnable.sol#776) is not in mixedCase
Parameter UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._symbol (UltimateTokenOwnable.sol#777) is not in mixedCase
Parameter UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._decimals (UltimateTokenOwnable.sol#778) is not in mixedCase
Parameter UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._initialSupply (UltimateTokenOwnable.sol#779) is not in mixedCase
Parameter UltimateTokenOwnable.initialize(address,string,string,uint8,uint256,uint256)._maxSupply (UltimateTokenOwnable.sol#780) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:UltimateTokenOwnable.sol analyzed (8 contracts with 93 detectors), 32 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):
https://testnet.bscscan.com/tx/0x419211777af7201984724b23c4a4a18984b626fa79820b56488687702aa0debe

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0xb09df272d43e114d483dc61788190d95fbe3594921eafc49a16abb480976c05b

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x2173c0ad803914830d905a552c17ad3a1630dc29eeeed04805bac53faea2f9de

4- Pause (passed):
https://testnet.bscscan.com/tx/0xea3053be474f9cdf82c91ecea35134337523b598537c5a3218f954096fc98b77

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can pause/unpause token.
- The owner can mint token.

CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	1
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – The owner can Pause the token.

Severity: High

Function: pause

Status: Open

Overview:

The owner can pause the token for an unlimited period of time which can lock the user's token.

```
function pause() public onlyOwner {
        _pause();
    };
```

Suggestion:

It is recommended that there should be a locking period.

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them

```
function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may have re-
verted");
    }
modifier reinitializer(uint8 version) {
        require(!_initializing && _initialized < version, "Initializable: contract
is already initialized");
        _initialized = version;
        _initializing = true;
        _;
        _initializing = false;
        emit Initialized(version);
    }
function _getInitializedVersion() internal view returns (uint8) {
        return _initialized;
    }
function _isInitializing() internal view returns (bool) {
        return _initializing;
    }
 function functionCall(address target, bytes memory data) internal returns (bytes
memory) {
        return functionCallWithValue(target, data, 0, "Address: low-level call
failed");
    }
function functionCallWithValue(address target, bytes memory data, uint256 value)
```

MANUAL TESTING

```
internal returns (bytes memory) {
        return functionCallWithValue(target, data, value, "Address: low-level call
with value failed");
    }
function functionStaticCall(address target, bytes memory data) internal view re-
turns (bytes memory) {
        return functionStaticCall(target, data, "Address: low-level static call
failed");
    }
function functionDelegateCall(address target, bytes memory data) internal returns
(bytes memory) {
        return functionDelegateCall(target, data, "Address: low-level delegate call
failed");
    }
modifier reinitializer(uint8 version) {
        require(!_initializing && _initialized < version, "Initializable: contract
is already initialized");
        _initialized = version;
        _initializing = true;
        _;
        _initializing = false;
        emit Initialized(version);
    }
function _getInitializedVersion() internal view returns (uint8) {
        return _initialized;
    }
function _isInitializing() internal view returns (bool) {
        return _initializing;
    }
```

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace