AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Dinoerc404

DATED : 13 March, 2024

AUDIT SUMMARY

Project name – Dinoerc404

Date: 13 March, 2024

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

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	1	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/0x3dd7cf5176bb285f1bb9dc182b17d4b1cb61ff1f#code

Token Information

Token Name : dinoerc404

Token Symbol: DINOERC404

Decimals: -

Token Supply: 600000000000000000000

Network: -

Token Type: -

Token Address: -

Checksum:
A9fe9e9c341ee7973b99e3c799ce8f6f

Owner:
-
(at time of writing the audit)

Deployer: -

TOKEN OVERVIEW

Fees:

Buy Fee: 0%

Sell Fee: 0%

Transfer Fee: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

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

STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse = (3 * denominator) ^ 2 (DINOERC404.flattened.sol#308)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse *= 2 - denominator * inverse (DINOERC404.flattened.sol#312)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse *= 2 - denominator * inverse (DINOERC404.flattened.sol#313)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse *= 2 - denominator * inverse (DINOERC404.flattened.sol#314)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse *= 2 - denominator * inverse (DINOERC404.flattened.sol#315)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse *= 2 - denominator * inverse (DINOERC404.flattened.sol#316)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - denominator = denominator / twos (DINOERC404.flattened.sol#293)
        - inverse *= 2 - denominator * inverse (DINOERC404.flattened.sol#317)
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) performs a multiplication on the result of a division:
        - prod0 = prod0 / twos (DINOERC404.flattened.sol#296)
        - result = prod0 * inverse (DINOERC404.flattened.sol#323)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
DINOERC404 (DINOERC404.flattened.sol#1944-2071) has incorrect ERC721 function interface:ERC404.approve(address,uint256) (DINOERC404.flattened.sol#1159-1170)
DINOERC404 (DINOERC404.flattened.sol#1944-2071) has incorrect ERC721 function interface:ERC404.transferFrom(address,address,uint256) (DINOERC404.flattened.sol#1218-1231)
DINOERC404 (DINOERC404.flattened.sol#1944-2071) has incorrect ERC721 function interface:IERC404.approve(address,uint256) (DINOERC404.flattened.sol#763-766)
DINOERC404 (DINOERC404.flattened.sol#1944-2071) has incorrect ERC721 function interface:IERC404.transferFrom(address,address,uint256) (DINOERC404.flattened.sol#773-777)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc721-interface
INFO:Detectors:
DINOERC404.constructor(string,string,uint8,address,uint256,address)._recipient (DINOERC404.flattened.sol#1951) lacks a zero-check on :
                - recipient = _recipient (DINOERC404.flattened.sol#1954)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
ERC404.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (DINOERC404.flattened.sol#1358-1414) uses timestamp for comparisons
        Dangerous comparisons:
        - deadline_ < block.timestamp (DINOERC404.flattened.sol#1367)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Math.mulDiv(uint256,uint256,uint256) (DINOERC404.flattened.sol#247-326) uses assembly
        - INLINE ASM (DINOERC404.flattened.sol#254-257)
        - INLINE ASM (DINOERC404.flattened.sol#278-285)
        - INLINE ASM (DINOERC404.flattened.sol#291-300)
Strings.toString(uint256) (DINOERC404.flattened.sol#597-617) uses assembly
        - INLINE ASM (DINOERC404.flattened.sol#603-605)
        - INLINE ASM (DINOERC404.flattened.sol#609-611)
ERC404._getOwnerOf(uint256) (DINOERC404.flattened.sol#1770-1778) uses assembly
        - INLINE ASM (DINOERC404.flattened.sol#1775-1777)
ERC404._setOwnerOf(uint256,address) (DINOERC404.flattened.sol#1780-1791) uses assembly
        - INLINE ASM (DINOERC404.flattened.sol#1783-1788)
ERC404._getOwnedIndex(uint256) (DINOERC404.flattened.sol#1793-1801) uses assembly
        - INLINE ASM (DINOERC404.flattened.sol#1798-1800)
ERC404._setOwnedIndex(uint256,uint256) (DINOERC404.flattened.sol#1803-1818) uses assembly
        - INLINE ASM (DINOERC404.flattened.sol#1810-1815)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Context._contextSuffixLength() (DINOERC404.flattened.sol#27-29) is never used and should be removed
Context._msgData() (DINOERC404.flattened.sol#23-25) is never used and should be removed
DoubleEndedQueue.back(DoubleEndedQueue.Uint256Deque) (DINOERC404.flattened.sol#936-943) is never used and should be removed
DoubleEndedQueue.clear(DoubleEndedQueue.Uint256Deque) (DINOERC404.flattened.sol#968-971) is never used and should be removed
DoubleEndedQueue.front(DoubleEndedQueue.Uint256Deque) (DINOERC404.flattened.sol#924-929) is never used and should be removed
DoubleEndedQueue.popFront(DoubleEndedQueue.Uint256Deque) (DINOERC404.flattened.sol#907-917) is never used and should be removed
DoubleEndedQueue.pushBack(DoubleEndedQueue.Uint256Deque,uint256) (DINOERC404.flattened.sol#861-868) is never used and should be removed
Math.average(uint256,uint256) (DINOERC404.flattened.sol#220-223) is never used and should be removed
Math.ceilDiv(uint256,uint256) (DINOERC404.flattened.sol#231-239) is never used and should be removed
Math.log10(uint256,Math.Rounding) (DINOERC404.flattened.sol#483-488) is never used and should be removed
Math.log2(uint256) (DINOERC404.flattened.sol#392-428) is never used and should be removed
Math.log2(uint256,Math.Rounding) (DINOERC404.flattened.sol#434-439) is never used and should be removed
Math.log256(uint256) (DINOERC404.flattened.sol#496-520) is never used and should be removed
Math.log256(uint256,Math.Rounding) (DINOERC404.flattened.sol#526-531) is never used and should be removed
Math.max(uint256,uint256) (DINOERC404.flattened.sol#205-207) is never used and should be removed
Math.min(uint256,uint256) (DINOERC404.flattened.sol#212-214) is never used and should be removed
```

STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
DINOERC404.erc1151 (DINOERC404.flattened.sol#1962) is set pre-construction with a non-constant function or state variable:
        - IERC1155(VOYAGE)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Pragma version^0.8.20 (DINOERC404.flattened.sol#2) 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 DINOERC404.mintERC20() (DINOERC404.flattened.sol#2049-2065):
        - (success) = address(recipient).call{value: transferAmount}() (DINOERC404.flattened.sol#2057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IERC404.DOMAIN_SEPARATOR() (DINOERC404.flattened.sol#798) is not in mixedCase
Function ERC404.DOMAIN_SEPARATOR() (DINOERC404.flattened.sol#1417-1422) is not in mixedCase
Variable ERC404._INITIAL_CHAIN_ID (DINOERC404.flattened.sol#1035) is not in mixedCase
Variable ERC404._INITIAL_DOMAIN_SEPARATOR (DINOERC404.flattened.sol#1038) is not in mixedCase
Parameter DINOERC404.updateTransferAmount(uint256)._newAmount (DINOERC404.flattened.sol#2067) is not in mixedCase
Variable DINOERC404.MAX_MINT (DINOERC404.flattened.sol#1960) is not in mixedCase
Variable DINOERC404.VOYAGE (DINOERC404.flattened.sol#1961) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
DINOERC404.VOYAGE (DINOERC404.flattened.sol#1961) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
DINOERC404.MAX_MINT (DINOERC404.flattened.sol#1960) should be immutable
DINOERC404.erc1151 (DINOERC404.flattened.sol#1962) should be immutable
DINOERC404.recipient (DINOERC404.flattened.sol#1964) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:DINOERC404.flattened.sol analyzed (14 contracts with 93 detectors), 74 result(s) found
```

FUNCTIONAL TESTING

1- Add To Whitelist (passed):

https://testnet.bscscan.com/tx/0xc358a0edf3e3a44f833a7fa8a05aeb73e2a4c62859c9ef7a99bbe186d0c353f1

2- Set ERC721 Transfer Exempt (passed):

https://testnet.bscscan.com/tx/0xfeaea6f8d3b62e4ee8c04a54d3deee2f129fa83b71386bf18887ce35705e52a2

3- Approve (passed):

https://testnet.bscscan.com/tx/0x2b4f1e842d55ffbc4a080fab80be58a9472b116114d30c437e4e17cb1c1955c2

4- Set Self ERC721 Transfer Exempt (passed):

https://testnet.bscscan.com/tx/0x569aec4f5aa7d512502b65d2208a9df861f984396bbf6029ba5c8f3e846cd3d6

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set ERC721 Transfer Exempt.
- The owner can add/remove address to whitelist.
- The owner can update the transfer amount.

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	0
Medium-Risk	0
Low-Risk	1
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Subject: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setERC721TransferExempt(
      address account_,
      bool value_
    ) external onlyOwner {
      _setERC721TransferExempt(account_, value_);
    }
function updateTransferAmount(uint256 _newAmount) public onlyOwner {
        transferAmount = _newAmount;
    }
```

Suggestion:

Emit an event for critical changes.

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 to avoid them.

```
function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
function _contextSuffixLength() internal view virtual returns (uint256) {
        return 0;
    }
function clear(Uint256Deque storage deque) internal {
    deque._begin = 0;
    deque._end = 0;
  }
function front(
    Uint256Deque storage deque
  ) internal view returns (uint256 value) {
    if (empty(deque)) revert QueueEmpty();
    return deque._data[deque._begin];
  }
function popFront(
    Uint256Deque storage deque
  ) internal returns (uint256 value) {
    unchecked {
      uint128 frontIndex = deque._begin;
      if (frontIndex == deque._end) revert QueueEmpty();
      value = deque._data[frontIndex];
      delete deque._data[frontIndex];
      deque._begin = frontIndex + 1;
    }
  }
```

MANUAL TESTING

```
function pushBack(Uint256Deque storage deque, uint256 value) internal {
    unchecked {
      uint128 backIndex = deque._end;
      if (backIndex + 1 == deque._begin) revert QueueFull();
      deque._data[backIndex] = value;
      deque._end = backIndex + 1;
    }
  }
function equal(string memory a, string memory b) internal pure returns (bool) {
        return bytes(a).length == bytes(b).length && keccak256(bytes(a)) ==
keccak256(bytes(b));
    }
function toHexString(address addr) internal pure returns (string memory) {
        return toHexString(uint256(uint160(addr)), ADDRESS_LENGTH);
    }
function toHexString(uint256 value) internal pure returns (string memory) {
        unchecked {
            return toHexString(value, Math.log256(value) + 1);
        }
function toStringSigned(int256 value) internal pure returns (string memory) {
        return string.concat(value < 0 ? "-" : "", toString(Signed-
Math.abs(value)));
    }
function average(int256 a, int256 b) internal pure returns (int256) {
        // Formula from the book "Hacker's Delight"
        int256 x = (a & b) + ((a ^ b) >> 1);
        return x + (int256(uint256(x) >> 255) & (a ^ b));
    }
function min(int256 a, int256 b) internal pure returns (int256) {
        return a < b ? a : b;
    }
function max(int256 a, int256 b) internal pure returns (int256) {
        return a > b ? a : b;
    }
function log256(uint256 value, Rounding rounding) internal pure returns (uint256) {
        unchecked {
            uint256 result = log256(value);
            return result + (unsignedRoundsUp(rounding) && 1 << (result << 3) <
value ? 1 : 0);
        }
    }
```

MANUAL TESTING

```
function log10(uint256 value, Rounding rounding) internal pure returns (uint256) {
        unchecked {
            uint256 result = log10(value);
            return result + (unsignedRoundsUp(rounding) && 10 ** result < value ? 1
: 0);
        }
    }
function log2(uint256 value, Rounding rounding) internal pure returns (uint256) {
        unchecked {
            uint256 result = log2(value);
            return result + (unsignedRoundsUp(rounding) && 1 << result < value ? 1
: 0);
        }
    }
function sqrt(uint256 a, Rounding rounding) internal pure returns (uint256) {
        unchecked {
            uint256 result = sqrt(a);
            return result + (unsignedRoundsUp(rounding) && result * result < a ? 1
: 0);
        }
    }
function mulDiv(uint256 x, uint256 y, uint256 denominator, Rounding rounding) in-
ternal pure returns (uint256) {
        uint256 result = mulDiv(x, y, denominator);
        if (unsignedRoundsUp(rounding) && mulmod(x, y, denominator) > 0) {
            result += 1;
        }
        return result;
    }
function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
        if (b == 0) {
            // Guarantee the same behavior as in a regular Solidity division.
            return a / b;
        }

        // (a + b - 1) / b can overflow on addition, so we distribute.
        return a == 0 ? 0 : (a - 1) / b + 1;
    }
function average(uint256 a, uint256 b) internal pure returns (uint256) {
        // (a + b) / 2 can overflow.
        return (a & b) + (a ^ b) / 2;
```

MANUAL TESTING

```
    }
function max(uint256 a, uint256 b) internal pure returns (uint256) {
        return a > b ? a : b;
    }
function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        unchecked {
            if (b == 0) return (false, 0);
            return (true, a % b);
        }
    }
function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        unchecked {
            if (b == 0) return (false, 0);
            return (true, a / b);
        }
    }
function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        unchecked {
            // Gas optimization: this is cheaper than requiring 'a' not being zero,
but the
            // benefit is lost if 'b' is also tested.
            // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
            if (a == 0) return (true, 0);
            uint256 c = a * b;
            if (c / a != b) return (false, 0);
            return (true, c);
        }
    }
function trySub(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        unchecked {
            if (b > a) return (false, 0);
            return (true, a - b);
        }
    }
function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
        unchecked {
            uint256 c = a + b;
            if (c < a) return (false, 0);
            return (true, c);
        }
    }
```

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