AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Ignite

DATED : 23 Feb, 2024

AUDIT SUMMARY

Project name – Ignite

Date: 23 Feb, 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	1	1	2
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/0xc931654259b134fad20c64f2608be44c62692907#code

Token Information

Token Name : Ignite

Token Symbol: GNC

Decimals: 18

Token Supply: 21000000

Network: Binance smart chain

Token Type: BEP-20

Token Address:
0x9E863CECA9c36903979A43C2edea19AD16FEc5E7

Checksum:
Ae1c3a4fbb6e83e8393a57617b5a221

Owner:
0x000000000000000000000000000000000000dEaD
(at time of writing the audit)

Deployer:
0x0c49acfA69E6478fCd1541f1EcEAf918f2660b89

TOKEN OVERVIEW

Fees:
Total Buy Tax: 6%
Total Sell Tax: 6%
Transfer Fee: 0-0%

Fees Privilege: Owner

Ownership: Renounced

Minting: No

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

AUDIT METHODOLOGY

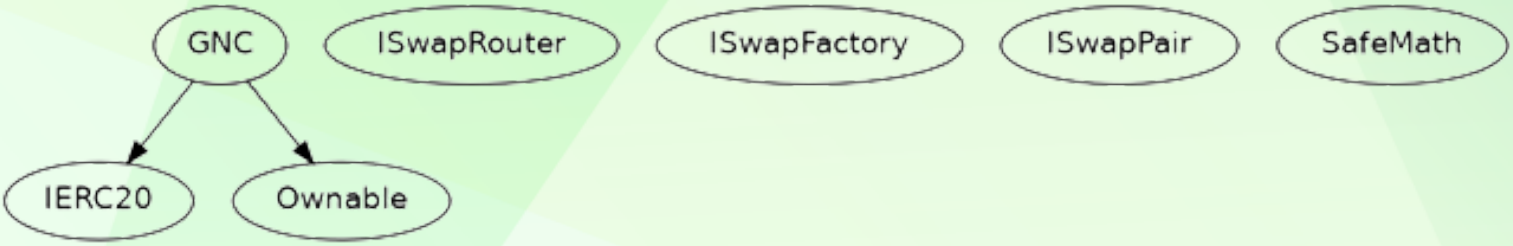
The auditing process will follow a routine as special considerations by Auditace:

- 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

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

INHERITANCE TREE



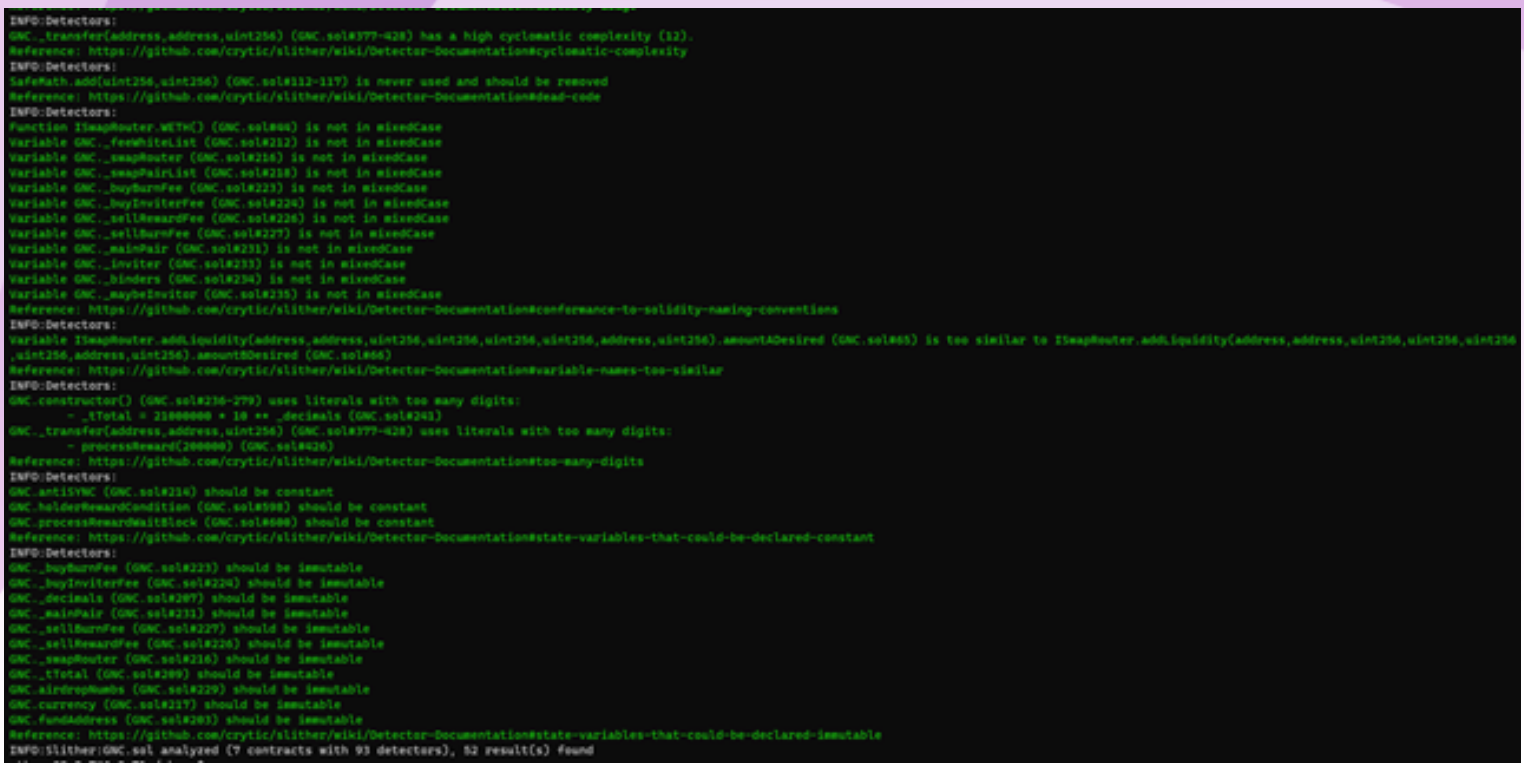
STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
GNC._tokenTransfer(address,address,uint256,bool,bool) (GNC.sol#430-519) performs a multiplication on the result of a division:
        - feeAmount = tAmount.div(10000).mul(swapFee) (GNC.sol#517)
GNC._tokenTransfer(address,address,uint256,bool,bool) (GNC.sol#430-519) performs a multiplication on the result of a division:
        - burnAmount_scope_0 = tAmount.div(10000).mul(_buyBurnFee) (GNC.sol#483)
GNC._tokenTransfer(address,address,uint256,bool,bool) (GNC.sol#430-519) performs a multiplication on the result of a division:
        - inviterAmount = tAmount.div(10000).mul(_buyInviterFee) (GNC.sol#491)
GNC._tokenTransfer(address,address,uint256,bool,bool) (GNC.sol#430-519) performs a multiplication on the result of a division:
        - burnAmount = tAmount.div(10000).mul(_sellBurnFee) (GNC.sol#462)
GNC._tokenTransfer(address,address,uint256,bool,bool) (GNC.sol#430-519) performs a multiplication on the result of a division:
        - rewardAmount = tAmount.div(10000).mul(_sellRewardFee) (GNC.sol#471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
GNC (GNC.sol#197-653) has incorrect ERC20 function interface:IERC20.approve(address,uint256) (GNC.sol#29)
GNC (GNC.sol#197-653) has incorrect ERC20 function interface:IERC20.transferFrom(address,address,uint256) (GNC.sol#31)
GNC (GNC.sol#197-653) has incorrect ERC20 function interface:GNC.approve(address,uint256) (GNC.sol#319-325)
GNC (GNC.sol#197-653) has incorrect ERC20 function interface:GNC.transferFrom(address,address,uint256) (GNC.sol#327-339)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
GNC._transfer(address,address,uint256).isRemove (GNC.sol#384) is a local variable never initialized
GNC._transfer(address,address,uint256).takeFee (GNC.sol#382) is a local variable never initialized
GNC._transfer(address,address,uint256).isAdd (GNC.sol#385) is a local variable never initialized
GNC._tokenTransfer(address,address,uint256,bool,bool).swapFee (GNC.sol#439) is a local variable never initialized
GNC._transfer(address,address,uint256).isSell (GNC.sol#383) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
GNC._isAddLiquidity() (GNC.sol#346-360) ignores return value by (r0,r1) = mainPair.getReserves() (GNC.sol#348)
GNC._isRemoveLiquidity() (GNC.sol#362-376) ignores return value by (r0,r1) = mainPair.getReserves() (GNC.sol#364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
GNC.allowance(address,address).owner (GNC.sol#313) shadows:
        - Ownable.owner() (GNC.sol#177-179) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GNC._bindInvitor(address,address) (GNC.sol#545-555) uses assembly
        - INLINE ASM (GNC.sol#548)
GNC.addHolder(address) (GNC.sol#581-595) uses assembly
        - INLINE ASM (GNC.sol#583-585)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GNC._transfer(address,address,uint256) (GNC.sol#377-428) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
SafeMath.add(uint256,uint256) (GNC.sol#112-117) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

STATIC ANALYSIS



FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x5ce3311117717969d4b53960b661df442ea821b60f11a265a3ca1673ae5186d5

2- Transfer (passed):

https://testnet.bscscan.com/tx/0x522c625b0454578d771ef473e6f16c31090d0ff8a490705eecd8ac128fee04c5

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	1
Low-Risk	1
Gas Optimization / Suggestions	2

MANUAL TESTING

Centralization – Divide before multiply

Severity: Medium

Function:

Status: Open

Overview:

Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.

```
function mint(address _to, uint256 _amount) public onlyOwner {
        _mint(_to, _amount);
        _moveDelegates(address(0), _delegates[_to], _amount);
    }
```

Suggestion:

Consider ordering multiplication before division.

MANUAL TESTING

Centralization – Local Variable Shadowing

Severity: Low

Function: _approve and allowance

Status: Open

Overview:

```
function allowance(
        address owner,
        address spender
    ) public view override returns (uint256) {
        return _allowances[owner][spender];
    }
```

Suggestion:

Rename the local variable that shadows another component.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Remove Safe Math

Status: Open

Line: 458 – 599

Overview:

Unused variables are allowed in Solidity, and they do. not pose a compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma

Status: Open

Overview:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.18;
```

Suggestion:

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

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