AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

HappyNewYear

DATED : 26 Dec 23'

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
Function: EnableTrading
Status: Open

Overview:
The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner {
    tradingEnabled = true;
  }
```

Suggestion
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

AUDIT SUMMARY

Project name – HappyNewYear

Date: 26 Dec, 2023

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

Audit Status: **Passed with high risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	0	2	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/0xb4ead26abee2607c2d46a6099c16dc6e68ea991c#code

Token Information

Token Address:
0xF543F9613CC6b6246751D0937b71b1747884B2a5

Name: HappyNewYear

Symbol: HPNY

Decimals: 18

Network: Etherscan

Token Type: ERC-20

Owner: 0x92277699Bf4bD613286E6dD0C76DBB4d539d85Ac

Deployer:
0x92277699Bf4bD613286E6dD0C76DBB4d539d85Ac

Token Supply: 366000000

Checksum: Ae032c616934aeb47e6039f76b20d2v5

Testnet:
https://testnet.bscscan.com/address/0xb4ead26abee2607c2d46a6099c16dc6e68ea991c#code

TOKEN OVERVIEW

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

Blacklist: No

Other Privileges:

-Whitelist to transfer without enabling trades

- Enabling trades

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

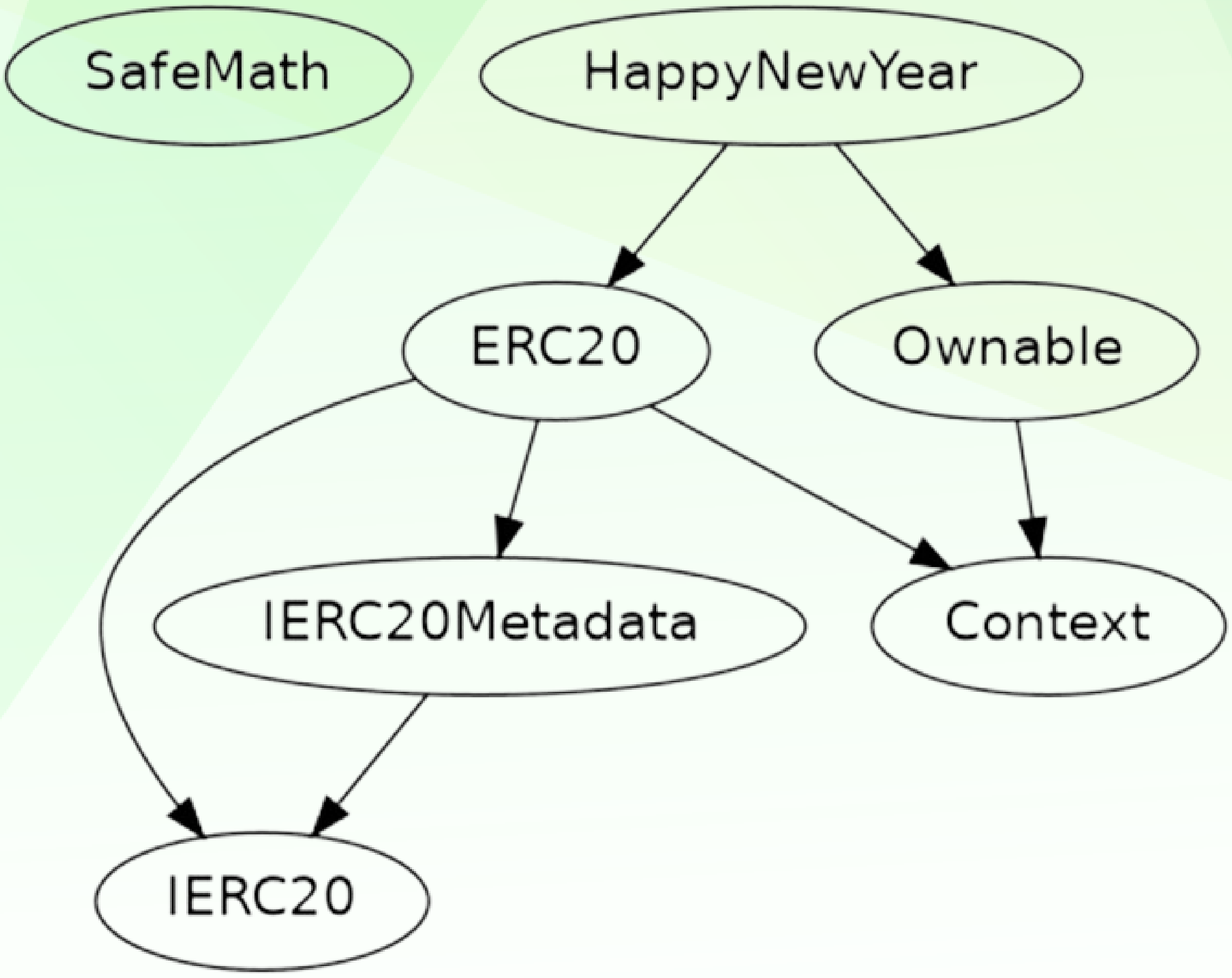
CLASSIFICATION OF RISK

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	2
◆ Gas Optimization / Suggestions	2

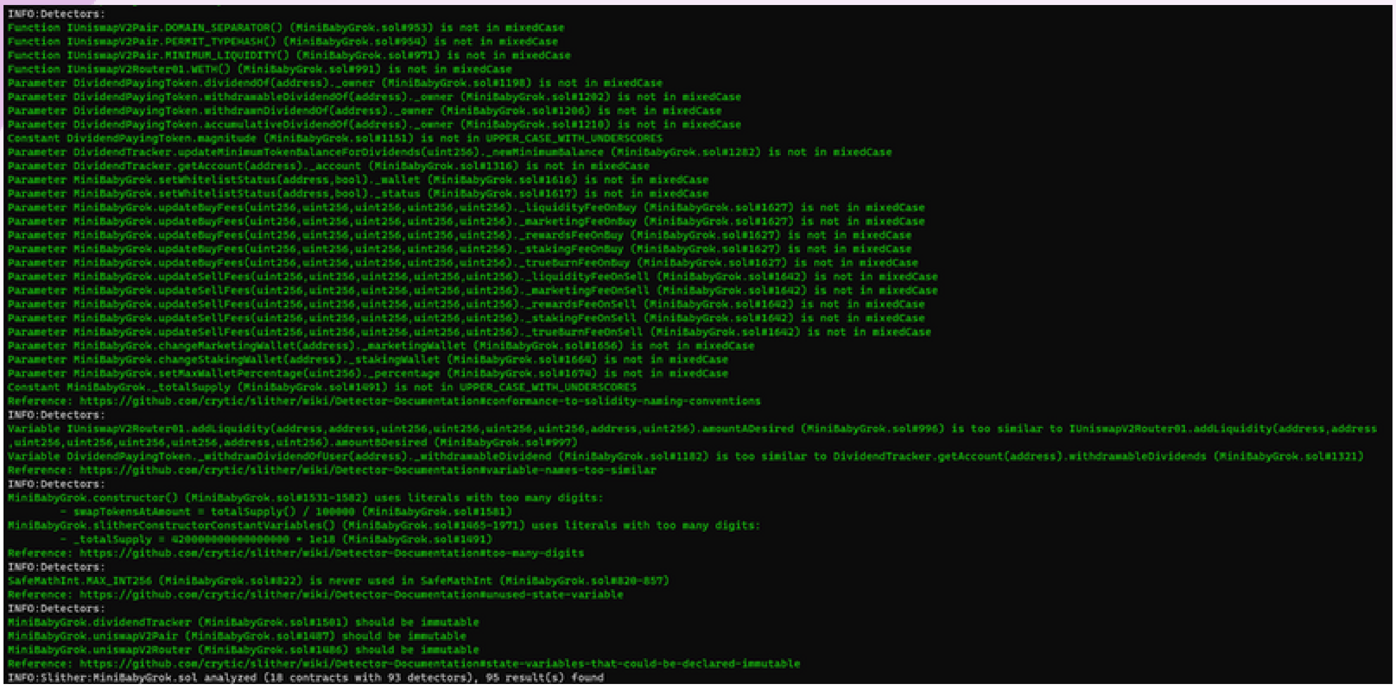
INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can set the pre-launch address.

STATIC ANALYSIS



```
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (MiniBabyGrok.sol#953) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (MiniBabyGrok.sol#954) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (MiniBabyGrok.sol#971) is not in mixedCase
Function IUniswapV2Router01.WETH() (MiniBabyGrok.sol#991) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (MiniBabyGrok.sol#1198) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (MiniBabyGrok.sol#1202) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (MiniBabyGrok.sol#1206) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (MiniBabyGrok.sol#1210) is not in mixedCase
Constant DividendPayingToken.magnitude (MiniBabyGrok.sol#1151) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.updateMinimumTokenBalanceForDividends(uint256)._newMinimumBalance (MiniBabyGrok.sol#1282) is not in mixedCase
Parameter DividendTracker.getAccount(address)._account (MiniBabyGrok.sol#1316) is not in mixedCase
Parameter MiniBabyGrok.setWhitelistStatus(address,bool)._wallet (MiniBabyGrok.sol#1616) is not in mixedCase
Parameter MiniBabyGrok.setWhitelistStatus(address,bool)._status (MiniBabyGrok.sol#1617) is not in mixedCase
Parameter MiniBabyGrok.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._liquidityFeeOnBuy (MiniBabyGrok.sol#1627) is not in mixedCase
Parameter MiniBabyGrok.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._marketingFeeOnBuy (MiniBabyGrok.sol#1627) is not in mixedCase
Parameter MiniBabyGrok.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._rewardsFeeOnBuy (MiniBabyGrok.sol#1627) is not in mixedCase
Parameter MiniBabyGrok.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._stakingFeeOnBuy (MiniBabyGrok.sol#1627) is not in mixedCase
Parameter MiniBabyGrok.updateBuyFees(uint256,uint256,uint256,uint256,uint256)._trueBurnFeeOnBuy (MiniBabyGrok.sol#1627) is not in mixedCase
Parameter MiniBabyGrok.updateSellFees(uint256,uint256,uint256,uint256,uint256)._liquidityFeeOnSell (MiniBabyGrok.sol#1642) is not in mixedCase
Parameter MiniBabyGrok.updateSellFees(uint256,uint256,uint256,uint256,uint256)._marketingFeeOnSell (MiniBabyGrok.sol#1642) is not in mixedCase
Parameter MiniBabyGrok.updateSellFees(uint256,uint256,uint256,uint256,uint256)._rewardsFeeOnSell (MiniBabyGrok.sol#1642) is not in mixedCase
Parameter MiniBabyGrok.updateSellFees(uint256,uint256,uint256,uint256,uint256)._stakingFeeOnSell (MiniBabyGrok.sol#1642) is not in mixedCase
Parameter MiniBabyGrok.updateSellFees(uint256,uint256,uint256,uint256,uint256)._trueBurnFeeOnSell (MiniBabyGrok.sol#1642) is not in mixedCase
Parameter MiniBabyGrok.changeMarketingWallet(address)._marketingWallet (MiniBabyGrok.sol#1656) is not in mixedCase
Parameter MiniBabyGrok.changeStakingWallet(address)._stakingWallet (MiniBabyGrok.sol#1664) is not in mixedCase
Parameter MiniBabyGrok.setMaxWalletPercentage(uint256)._percentage (MiniBabyGrok.sol#1674) is not in mixedCase
Constant MiniBabyGrok._totalSupply (MiniBabyGrok.sol#1491) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (MiniBabyGrok.sol#996) is too similar to IUniswapV2Router01.addLiquidity(address,address
,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (MiniBabyGrok.sol#997)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (MiniBabyGrok.sol#1182) is too similar to DividendTracker.getAccount(address).withdrawableDividends (MiniBabyGrok.sol#1321)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
MiniBabyGrok.constructor() (MiniBabyGrok.sol#1531-1582) uses literals with too many digits:
        - swapTokensAtAmount = totalSupply() / 100000 (MiniBabyGrok.sol#1581)
MiniBabyGrok.slitherConstructorConstantVariables() (MiniBabyGrok.sol#1465-1971) uses literals with too many digits:
        - _totalSupply = 420000000000000000 * 1e18 (MiniBabyGrok.sol#1491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
SafeMathInt.MAX_INT256 (MiniBabyGrok.sol#822) is never used in SafeMathInt (MiniBabyGrok.sol#820-857)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
MiniBabyGrok.dividendTracker (MiniBabyGrok.sol#1501) should be immutable
MiniBabyGrok.uniswapV2Pair (MiniBabyGrok.sol#1487) should be immutable
MiniBabyGrok.uniswapV2Router (MiniBabyGrok.sol#1486) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:MiniBabyGrok.sol analyzed (18 contracts with 93 detectors), 95 result(s) found
```

Result => A static analysis of contract’s source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0xf542c7be1b885451aae8dd27bef39a738c902a24b5fe5a3b45adca74e3584590

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0x37f80d7f3b396dd38b127a97f35b561ea19bbbfb17585526dcc7988f53b67c1c

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0x5fddbafc3cd7426f6e9f08aad8c4150b81722cd74bdbaa1859c8eb09ebc79dbb

4- Set Pre-Launch Address (passed):

https://testnet.bscscan.com/tx/0xe3ae5a89a0d95e279600111aa145bbee6bb1aaa97cf4e8e8a38492a675f0dfbb

5- Enable Trading (passed):

https://testnet.bscscan.com/tx/0x941525db16eabb5879cfe434272eccc659e775a37769f130617905ffc42e4171

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
Function: EnableTrading
Status: Open

Overview:
The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner {
    tradingEnabled = true;
  }
```

Suggestion
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

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 setPreLaunchAddress(
address _address,
bool state
  ) external onlyOwner {
    presaleAddress[_address] = state;
  }
```

MANUAL TESTING

Centralization – Local Variable Shadowing

Severity: Low

Status: Open

Subject: Shadowing Local

Overview:

```
constructor() ERC20("Happy New Year", "HPNY") {
    uint256 totalSupply = 366_000_000 * 10 ** 18;
    presaleAddress[msg.sender] = true;
    _mint(msg.sender, totalSupply);
  }
```

Suggestion:
Rename the local variable that shadows another component.

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 _msgData() internal view virtual returns (bytes
calldata) {
    return msg.data;
  }
```

MANUAL TESTING

Optimization

Severity: Informational

subject: Remove Safe Math

Status: Open

Line: 10-235

Overview:

compiler version above 0.8.0 can control arithmetic overflow/underflow, It is recommended to remove the unwanted code to avoid high gas fees.

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