auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Hydra

DATED : 22 Jan, 2024

MANUAL TESTING

Centralization – Owner Can Mint Tokens

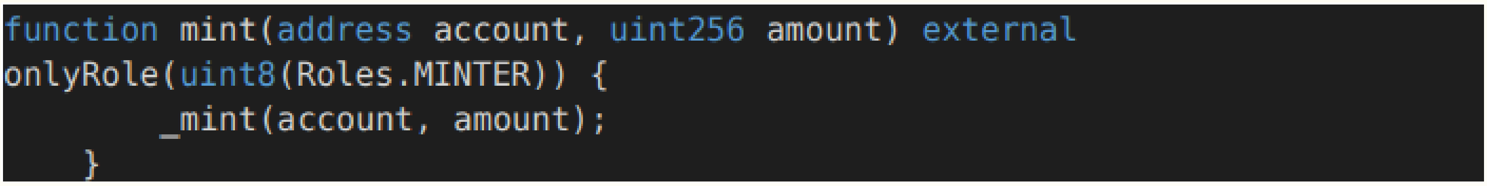
Severity: High

Function: Mint

Status: Open

Overview:

The owner can mint unlimited tokens which is not recommended as this functionality can cause the token to lose it's value and the owner can also use it to manipulate the price of the token.



```
function mint(address account, uint256 amount) external
onlyRole(uint8(Roles.MINTER)) {
        _mint(account, amount);
    }
```

Suggestion:

It is recommended that the total supply of the. Tokens should not be changed after initial deployment.

MANUAL TESTING

Centralization – Owner Can Burn Tokens

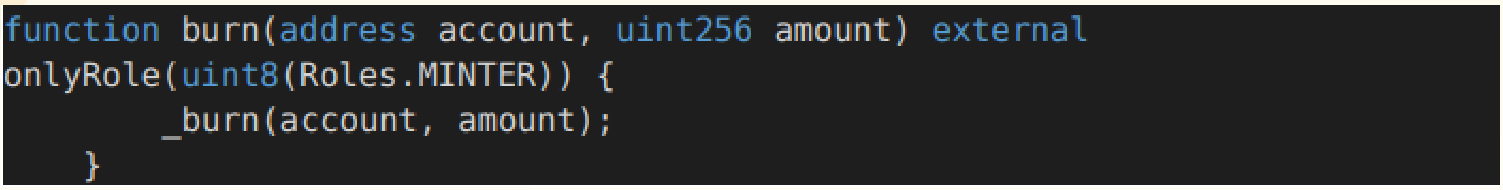
Severity: High

Function: burn

Status: Open

Overview:

The owner can burn tokens without approval from any wallet,



```
function burn(address account, uint256 amount) external
onlyRole(uint8(Roles.MINTER)) {
        _burn(account, amount);
    }
```

Suggestion:

There should not be any burning without any allowance from the user. Otherwise user can loose his tokens

AUDIT SUMMARY

Project name – Hydra

Date: 22 Jan, 2024

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

Audit Status: **High Risk Major Flag**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	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/0xd3ff98e80cef07d79549217ea08fe312969e1728#code

Token Information

Token Name : Hydra

Token Symbol: Hydra

Decimals: 18

Token Supply: 1000,000,000

Network: EtherScan

Token Type: ERC-20

Token Address:
0xAC11a6166D01F9Ac28f708F9C4a973ED0e434877

Checksum:
Ae1c3a4fbb6e83e8393a57617b5a5b32

Owner:
0xe7Cc235bbdA30EEaAaEEa3F2DdE198405ef35c3f
(at time of writing the audit)

Deployer:
0xe7Cc235bbdA30EEaAaEEa3F2DdE198405ef35c3f

TOKEN OVERVIEW

Fees:

Buy Fee: 0%

Sell Fee: 0%

Transfer Fee: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: Yes

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Burn: Yes

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.

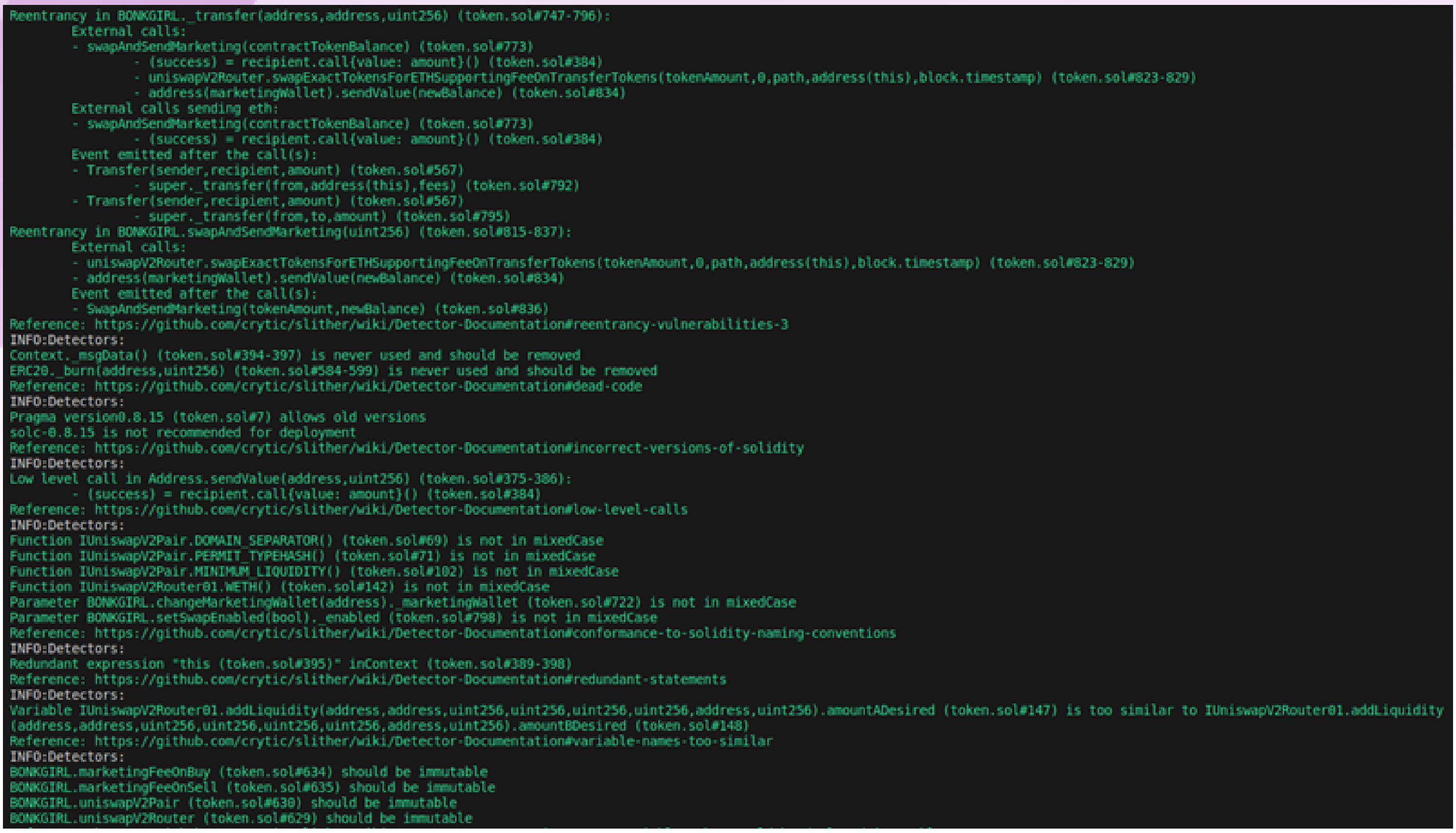
VULNERABILITY CHECKLIST

- ✓ 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

STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.



```
Reentrancy in BONKGIRL._transfer(address,address,uint256) (token.sol#747-796):
        External calls:
        - swapAndSendMarketing(contractTokenBalance) (token.sol#773)
                - (success) = recipient.call{value: amount}() (token.sol#384)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (token.sol#823-829)
                - address(marketingWallet).sendValue(newBalance) (token.sol#834)
        External calls sending eth:
        - swapAndSendMarketing(contractTokenBalance) (token.sol#773)
                - (success) = recipient.call{value: amount}() (token.sol#384)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (token.sol#567)
                - super._transfer(from,address(this),fees) (token.sol#792)
        - Transfer(sender,recipient,amount) (token.sol#567)
                - super._transfer(from,to,amount) (token.sol#795)
Reentrancy in BONKGIRL.swapAndSendMarketing(uint256) (token.sol#815-837):
        External calls:
        - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (token.sol#823-829)
        - address(marketingWallet).sendValue(newBalance) (token.sol#834)
        Event emitted after the call(s):
        - SwapAndSendMarketing(tokenAmount,newBalance) (token.sol#836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (token.sol#394-397) is never used and should be removed
ERC20._burn(address,uint256) (token.sol#584-599) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.15 (token.sol#7) allows old versions
solc-0.8.15 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) (token.sol#375-386):
        - (success) = recipient.call{value: amount}() (token.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (token.sol#69) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (token.sol#71) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (token.sol#102) is not in mixedCase
Function IUniswapV2Router01.WETH() (token.sol#142) is not in mixedCase
Parameter BONKGIRL.changeMarketingWallet(address)._marketingWallet (token.sol#722) is not in mixedCase
Parameter BONKGIRL.setSwapEnabled(bool)._enabled (token.sol#798) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (token.sol#395)" inContext (token.sol#389-398)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (token.sol#147) is too similar to IUniswapV2Router01.addLiquidity
(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (token.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
BONKGIRL.marketingFeeOnBuy (token.sol#634) should be immutable
BONKGIRL.marketingFeeOnSell (token.sol#635) should be immutable
BONKGIRL.uniswapV2Pair (token.sol#630) should be immutable
BONKGIRL.uniswapV2Router (token.sol#629) should be immutable
```

Functional Tests

1- Approve (passed):

https://testnet.bscscan.com/tx/0xf91f0f8bf7367e4e43322a55f822de08f103e956ee8a31b2e6c2a6fd67ee978a

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0xe937e3c4905e358ae3038974239da1a01f742957c40c5698bed2c5e7141ef9d4

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0xe37a56f0e1a3de4196f1ea9e179fdc7cbcf5b1cdd85de948c7f2e086c366d1c1

4- marketing Wallet(passed):

https://testnet.bscscan.com/tx/0x195281882f13a9b4fdc7d65cdb94001db5d503ece6237abede60eab5b99f54bd

5- exclude from fees (passed):

https://testnet.bscscan.com/tx/0x5653a8016e996312c9b151f28d8df276983df2bdb98c054aba77e0bd61d367dc

6- Enable Trading (passed):

https://testnet.bscscan.com/tx/0x97ac01049caf1d375c5c5bffb9e57b8567508f26cd5fdcc8c369828288032989

7- transfer (passed):

https://testnet.bscscan.com/tx/0x24bf3aeff77fecc714912473f3493d0089f87f7d03d2edb662878834e43de8a4

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	2
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Owner Can Mint Tokens

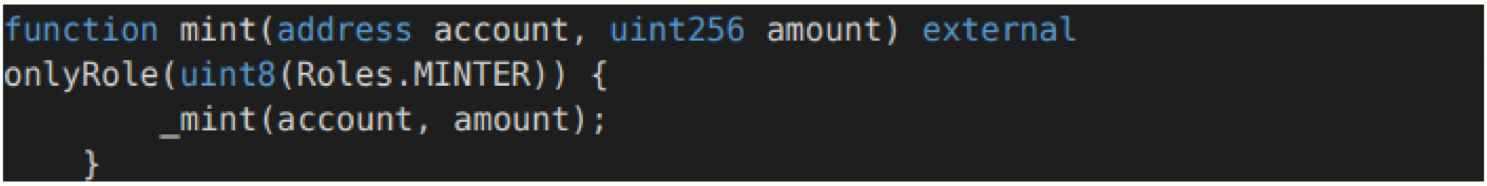
Severity: High

Function: Mint

Status: Open

Overview:

The owner can mint unlimited tokens which is not recommended as this functionality can cause the token to lose it's value and the owner can also use it to manipulate the price of the token.



```
function mint(address account, uint256 amount) external
onlyRole(uint8(Roles.MINTER)) {
        _mint(account, amount);
    }
```

Suggestion:

It is recommended that the total supply of the. Tokens should not be changed after initial deployment.

MANUAL TESTING

Centralization – Owner Can Burn Tokens

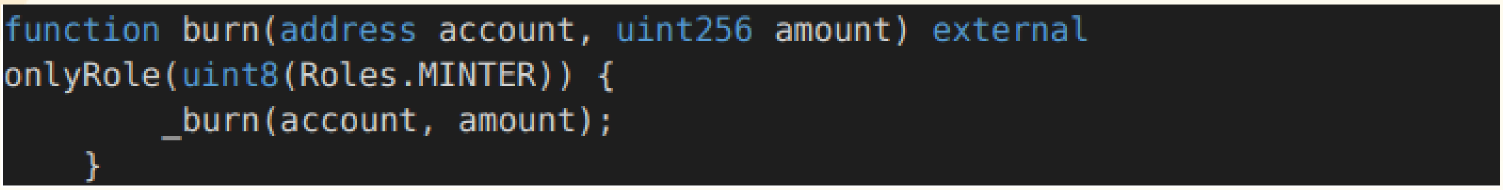
Severity: High

Function: burn

Status: Open

Overview:

The owner can burn tokens without approval from any wallet,



```
function burn(address account, uint256 amount) external
onlyRole(uint8(Roles.MINTER)) {
        _burn(account, amount);
    }
```

Suggestion:

There should not be any burning without any allowance from the user. Otherwise user can loose his tokens

MANUAL TESTING

Optimization

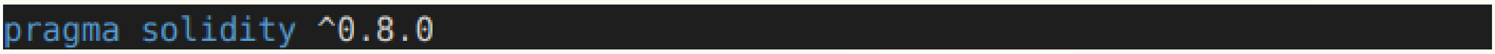
Severity: Informational

Function: Floating Pragma Solidity version

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 ^0.8.0
```

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