AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PENGS

DATED : 31 May 23'

AUDIT SUMMARY

Project name – PENGS

Date: 31 May, 2023

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	2	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:
Contract has been tested on binance smart chain testnet which can be found in below link:
https://testnet.bscscan.com/token/0xf24867426bfbfdbe966fce0720e1484bf2d75b57

Token Information

Token Name : Penguins

Token Symbol: PENGS

Decimals: 9

Token Supply: 100,000,000,000

Token Address:
0xd31036B1e9b88eBb04816AC32c10E3bd909c29b9

Checksum:
697239e1834e140a2eea5013d85badb967d6a846

Owner:
0x50d8229c70a75700a583085332759862 4031De3f
(at time of writing the audit)

Deployer:
0x50d8229c70a75700a583085332759862 4031De3f

TOKEN OVERVIEW

Fees:

Buy Fees: 5%

Sell Fees: 5%

Transfer Fees: 5%

Fees Privilege: Immutable fees

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Changing swap threshold - toggling internal swap - including in fees - excluding from fees

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

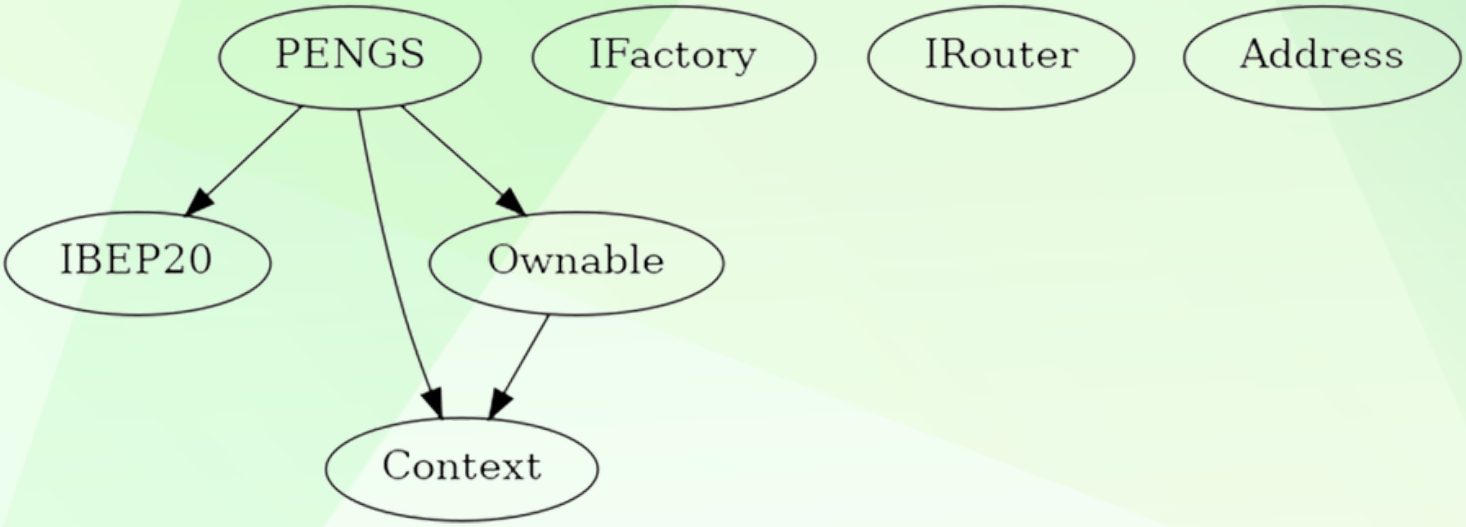
CLASSIFICATION OF RISK

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

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change buy/sell fees (static fees)**
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface			
└	createPair	External ❗	🛑	NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
Address	Library			
└	sendValue	Internal 🔒	🛑	
PENGS	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗

CONTRACT ASSESMENT

└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	isExcludedFromReward	Public ❗		NO ❗
└	reflectionFromToken	Public ❗		NO ❗
└	tokenFromReflection	Public ❗		NO ❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		NO ❗
└	_reflectRfi	Private 🔐	🛑	
└	_takeMarketing	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForBNB	Private 🔐	🛑	
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in PENGS.transferFrom(address,address,uint256) (contracts/Token.sol#274-289):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#279)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#131)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#560-566)
                - address(marketingWallet).sendValue(deltaBalance) (contracts/Token.sol#547)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#279)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#131)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#485)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/Token.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

PENGS.includeInReward(address) (contracts/Token.sol#364-376) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#45-48) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

PENGS._rTotal (contracts/Token.sol#159) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
PENGS.swapTokensAtAmount (contracts/Token.sol#161) is set pre-construction with a non-constant function or state variable:
        - _tTotal / 5000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#125-136):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#101) is not in mixedCase
Struct PENGS.valuesFromGetValues (contracts/Token.sol#188-196) is not in CapWords
Parameter PENGS.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#603) is not in mixedCase
Parameter PENGS.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#604) is not in mixedCase
Parameter PENGS.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#605) is not in mixedCase
Constant PENGS._decimals (contracts/Token.sol#155) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PENGS._name (contracts/Token.sol#171) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PENGS._symbol (contracts/Token.sol#172) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#46)" inContext (contracts/Token.sol#40-49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PENGS._tTotal (contracts/Token.sol#158) should be constant
PENGS.deadWallet (contracts/Token.sol#168) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PENGS.pair (contracts/Token.sol#153) should be immutable
PENGS.router (contracts/Token.sol#152) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x07f7a2be8c6002bd74717b847e0d11eaddd284af9b011f086879551f2d571c1b

2- Buying when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x397300e53ca5290427194382ab91463c69d71e59d1d6853e5c67f5b599c7a8d9

3- Selling when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xa51e210d2ece5765af4b6db2bd6c0c6f666ecbecb325264268a3dd72313dc1f5

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x1b34190d7d4abbf0a3a6536a02ddbe57cb1b364ab49d81e8df08022265f7c7b8

5- Buying when not excluded from fees (5% tax) (passed):
https://testnet.bscscan.com/tx/0x8efced2393bc223937d547966028fd0115307abea3f3bb3b98d18253d3484ad1

6- Selling when not excluded from fees (5% tax) (passed):
https://testnet.bscscan.com/tx/0x021288efa3962ff89062ec8796695b1b776de6a633e74acaec96158ece0e09eb

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (5% tax) (passed):

https://testnet.bscscan.com/tx/0x6bb93c5e0f77bc81000a0419e61179c22d614c86dd2d43bd753ea7475a6504c6

8- Internal swap (marketing wallet received bnb) (passed):

https://testnet.bscscan.com/address/0x0d679353abcf229f9af4734ff4aa819004dc7f12#internaltx

MANUAL TESTING

Logical – Incorrect condition at updateSwapTokensAtAmount

Severity: Medium

function: updateSwapTokensAtAmount

Status: Not Resolved

Overview:

Error message indicates that new swapthreshold must be less than 1% of total supply, while 1e15 (1,000,000,000,000,000) is 100x of total supply

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(
        amount <= 1e15,
        "Cannot set swap threshold amount higher than 1% of tokens"
    );
    emit SwapTokensAtAmountUpdated(
        swapTokensAtAmount,
        amount * 10 ** _decimals
    );
    swapTokensAtAmount = amount * 10 ** _decimals;
}
```

Suggestion

To mitigate this Logical issue, ensure that swapTokensAtAmount is less than 1e9 which is 1% of total supply

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(
        amount <= 1e9,
        "Cannot set swap threshold amount higher than 1% of tokens"
    );
    emit SwapTokensAtAmountUpdated(
        swapTokensAtAmount,
        amount * 10 ** _decimals
    );
    swapTokensAtAmount = amount * 10 ** _decimals;
}
```

MANUAL TESTING

Logical – setting swapthreshold to 0 may disable some sells

Severity: Medium

function: **_transfer** and **swapAndLiquify**

Status: Not Resolved

Overview:

swapTokensAtAmount can be set to 0, since contract tries to perform an internal swap (I.e convert token balance of the contract to bnb and then send it to marketing wallet) when balance of the contract is equal or more than 0, if balance of the contract is 0 and swapTokensAtAmount is also equal to 0 the sell or transfer function will be reverted.

```
bool canSwap = balanceOf(address(this)) >= swapTokensAtAmount;
if (
   !swapping &&
   canSwap &&
   from != pair &&
   !_isExcludedFromFee[from] &&
   !_isExcludedFromFee[to]
) {
   swapAndLiquify();
}
```

Suggestion

To mitigate this Logical issue, ensure that swapTokensAtAmount is more than 0 or stop swapAndLiquify if balance of the contract is equal to 0

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
  require(
    amount <= 1e9,
    "Cannot set swap threshold amount higher than 1% of tokens"
  );
  require(
    amount >= 1e6,
    "Cannot set swap threshold amount less than 0.1% of tokens"
  );
 //rest of the code
}

function swapAndLiquify() private lockTheSwap {

   uint256 contractBalance = balanceOf(address(this));

   if(contractBalance == 0) return;

   //rest of the code

}
```

MANUAL TESTING

Informational – Static fees

Status: Not Resolved

Overview:

Fees are immutable (5% for buy/sell/transfer), this means owner is not able to adjust fees based on different market conditions.
For example;
owner might need to increase sell tax to reduce selling pressure or decrease buy fees to incourage investors for purchasing the token.

Suggestion

Its suggested to create setter functions to be able to adjust fees in a safe range.

0 <= Buy Fees <= 10

0 <= Sell Fees <= 10

0 <= Transfer Fees <= 10

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