ETHOS
AUDIT

Pixel Hub

Audit Report

Jul. 20, 2022

Contents

Executive Summary

Audit Details

Project Name	Pixel hub
Codebase	https://bscscan.com/address/0x7a4A9D34A825b28259c6B229111E216367c81a99#code
Initial Audit Date	Jun. 21, 2022
Revision Dates	-
Methodology	Manual, Automated

Methodology

This audit's objectives are to evaluate:

- Security-related issues
- Code quality
- Relevant documentation
- Adherence to specifications
- Adherence to best practices

This audit examines the possibility of issues existing along the following vectors (but not limited to):

- Single & Cross-Function Reentrancy
- Front Running (Transaction Order Dependence)
- Timestamp dependence
- Integer Overflow and Underflow
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Number rounding errors
- DoS with (Unexpected) Revert
- DoS with Block Gas Limit
- Insufficient gas griefing
- Forcibly sending native currency
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification Contradiction
- Functionality duplication
- Malicious token minting

The code review conducted for this audit follows the following structure:

1. Review of specifications, documentation to assess smart contract functionality
2. Manual, line-by-line review of code
3. Code's adherence to functionality as presented by documentation
4. Automated tool-driven review of smart contract functionality
5. Assess adherence to best practices
6. Provide actionable recommendations

Results Summary

The Pixel Hub token project has been audited by Ethos and has been given a **PASSING** grade.

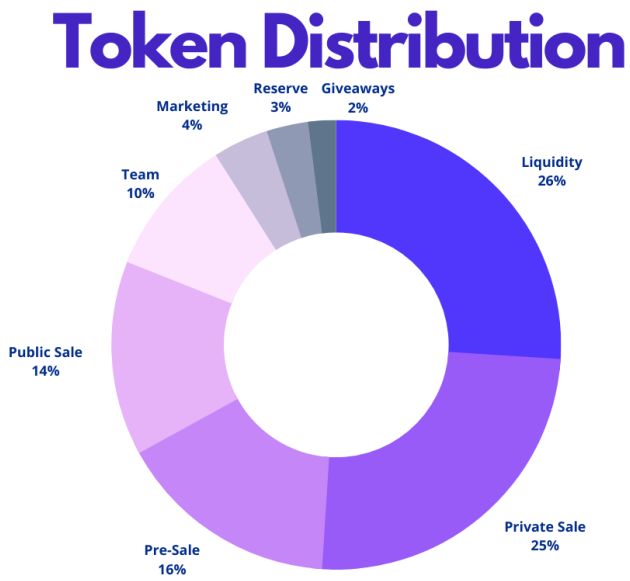
The PHT token is a reflection token built on the BSC with the following details:

Total Supply: 1,000,000,000
Buy/Sell Tax: 9% (to be updated by owner post launch)
Max tx: NA
Max wallet: NA
Tx Cooldown: 45 seconds

The audit found several low risk and informational issues that don't require any changes due to their low impact on the overall security of the smart contract and wallets.

The contract also does not contain any backdoors or malicious code. There are functions that allow the owner to update the tax fees per transaction which requires community transparency to mitigate any risk to token investors.

PHT tokenomics and whitepaper can be found here: https://whitepaper.pixelhub.finance/pixel-hub-token-pht/tokenomics



Issues Reported

Severity	Unresolved	Acknowledged	Resolved
High	0	0	0
Medium	0	0	0
Low	0	3	0
Informational	0	9	0

Issues Summary

ID	Title	Severity	Status
PHT-0	Re-entrancy	Low	Acknowledged
PHT-1	Unhandled return value	Low	Acknowledged
PHT-2	Missing zero-address validation	Low	Acknowledged
PHT-3	Missing events	Info	Acknowledged
PHT-4	Block timestamp comparison	Info	Acknowledged
PHT-5	Dead code	Info	Acknowledged
PHT-6	Function initializing state	Info	Acknowledged
PHT-7	Incorrect Solidity version	Info	Acknowledged
PHT-8	Low-level calls	Info	Acknowledged
PHT-9	Unused state variables	Info	Acknowledged
PHT-10	State variables that could be constant	Info	Acknowledged
PHT-11	Public functions that could be external	Info	Acknowledged

Detailed Findings

Code Documentation

The code contains minimal commenting.

Adherence to Specifications

The PHT smart contract adheres to the smart contract functionality described by the project documentation and is in line with its intended usage.

Adherence to Best Practices

The PHT smart contract adheres to the best practices associated with a standard EVM compatible Solidity smart contract.

PHT-0 – Re-entrancy

Severity: Low **Status**: Acknowledged

Description: A state variable is changed after a contract uses call.value. The attacker uses a fallback function—which is automatically executed after value is transferred from the targeted contract—to execute the vulnerable function again, before the state variable is changed.

Risk: A contract that holds a map of account balances allows users to call a withdraw function. However, withdraw calls send which transfers control to the calling contract, but doesn't decrease their balance until after send has finished executing. The attacker can then repeatedly withdraw money that they do not have.

Recommendation: Update all bookkeeping state variables i.e. update of ***_balances*** array before transferring execution of the ***swapBack()*** call. Alternatively, use the reentrancyGuard modifier.

Occurrences: PHT.sol#507

Team comments: The issue has been acknowledged by the team, but due to the low severity and the contract having been deployed, it is not expected to be updated.

PHT-1 – Unhandled return value

Severity: Low **Status**: Reported

Description: The return value of the function call ***addLiquidityETH*** is not stored in any local or state variable.

Risk: The computation has no affect and if any call of this function fails, it will revert, however the remaining steps will still execute creating a mismatch of state.

Recommendation: Store and check the return values of all function calls with return values and execute remaining steps afterwards.

Occurrences: PHT.sol#586-630

PHT-2 – Missing zero-address validation

Severity: Low **Status**: Reported

Description: Functions perform address assignments without checking for zero-address first, in instances where a zero-address assignment is not desired.

Risk: Assigning a zero-address to a crucial address variable where it is not desired may be unwanted functionality and should be checked for.

Recommendation: Check that the address is not zero.

Occurrences: PHT.sol#116; PHT.sol#675; PHT.sol#676;

PHT-3 – Missing events

Severity: Info **Status**: Reported

Description: Missing events for critical arithmetic parameters.

Risk: If execution of functions which update state variables do not emit events, they cannot be tracked by dApps which may rely on success/failure of such calls.

Recommendation: Emit an event for critical parameter changes.

Occurrences: PHT.setTxLimit; PHT.setbuyFees; PHT.setsellFees; PHT.setSwapBackSettings; PHT.setTargetLiquidity;

PHT-4 – Block timestamp comparison

Severity: Informational **Status**: Reported

Description: ***_transferFrom*** function uses a require statement that relies on a block timestamp comparison.

Risk: Miners can manipulate block.timestamp value to exploit the require statement and contract.

Recommendation: Avoid using block.timestamp for comparison logic.

Occurances: PHT.sol#498

PHT-5 – Dead code

Severity: Informational **Status**: Reported

Description: PHT.takesellFee is an internal function but never used and can be removed.

Risk: Gas optimization

Recommendation: Remove the unused function.

Occurrences: PHT.sol#554-561

PHT-6 – Function initializing state

Severity: Informational **Status**: Reported

Description: Variables are set pre-construction with a non-constant function or state variable.

Risk: Special care must be taken when initializing state variables from an immediate function call so as not to incorrectly assume the state is initialized.

Recommendation: Remove any initialization of state variables via non-constant state variables or function calls. If variables must be set upon contract deployment, locate initialization in the constructor instead.

Occurrences: PHT._maxTxAmount; PHT._maxWalletToken; PHT.swapThreshold;

PHT-7 – Incorrect Solidity version

Severity: Informational **Status**: Reported

Description: PHT.sol uses ^0.7.4 pragma, which allows old versions. Solc frequently releases new compiler versions.

Risk: Using an old version prevents access to new Solidity security checks.

Recommendation: Use the most recent pragma versions where possible.

Occurrences: PHT.sol#5

PHT-8 – Low-level calls

Severity: Informational **Status**: Reported

Description: The use of low-level calls is error-prone: (tmpSuccess) = address(marketingFeeReceiver).call{gas: 30000,value: amountBNBMarketing}()

Risk: Low-level calls do not check for code existence or call success.

Recommendation: Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

Occurrences: PHT.sol#614

PHT-9 – Unused state variables

Severity: Informational **Status**: Reported

Description: Some state variables are unused and can be removed.

Risk: Gas optimization

Recommendation: Remove unused state variables.

Occurrences: PHT.Reward; PHT.dev;

PHT-10 – State variables that could be constant

Severity: Informational **Status**: Reported

Description: Constant state variables should be declared constant to save gas.

Risk: Gas optimization

Recommendation: Declare state variables as constant.

Occurrences: PHT.DEAD; PHT.Reward; PHT.WBNB; PHT.ZERO; PHT._totalSupply; PHT.dev; PHT.feeDenominator; PHT.launchedAt;

PHT-11 – Public functions that could be external

Severity: Informational **Status**: Reported

Description: Public functions that are never called by the contract should be declared external to save gas.

Risk: Gas optimization

Recommendation: Use the external attribute for functions never called from the contract.

Occurrences: PHT.cooldownEnabled;