Revoluzion Audit

Audit Report

Name	: Pong Heroes
Symbol	: PONG
Decimals	: 18
Address	: 0x651189C8c0ABBD79d51AF276AA241915CA782b21
Owner	: 0xD1A3AFbAc5DE5BE0091FC19404Ac258B84C6157D
Network	: Binance Smart Chain (Mainnet)
Type	: BEP20
Audited on	: 20 February 2023

Revoluzion Audit

Contents

Project Overview

Name	Pong Heroes
Symbol	PONG
Decimals	18
Total Supply	1,000,000,000
Tax	Buy 3% | Sell 3% — (Variable Tax: Max Buy 5% | Max Sell 5%)
Compiler Version	v0.8.18+commit.87f61d96
Optimization	Yes with 200 runs
License Type	MIT
Explorer Link	https://bscscan.com/address/0x651189C8c0ABBD79d51AF276AA241915CA782b21
Create Tx	0xf67da9a4b4a420e3e196256a6319b33a95b30e118d75f21ff12b7a02ce02b8b8
Creator	0xD1A3AFbAc5DE5BE0091FC19404Ac258B84C6157D
Featured Wallet	Marketing Wallet — 0xc36A79776364177A72F40ce31ed425A982ff1e43
Website	https://pongheroes.io

Project Description

According to their website

Pong Heroes is inspired by the legendary first game ever, reintroduced into web3. We are a hyper-casual, fast-paced, Win2Earn, Free2Play skill-driven competitive multiplayer PONG game, aiming for real sustainability in the web3 gaming space.

Release Date : TBA

Category : P2E Web3 Game

Online Presence

About Website

Registrar : https://www.eurodns.com

Domain Expiration : 2023-05-03

SSL Certificate : Issued by Let's Encrypt

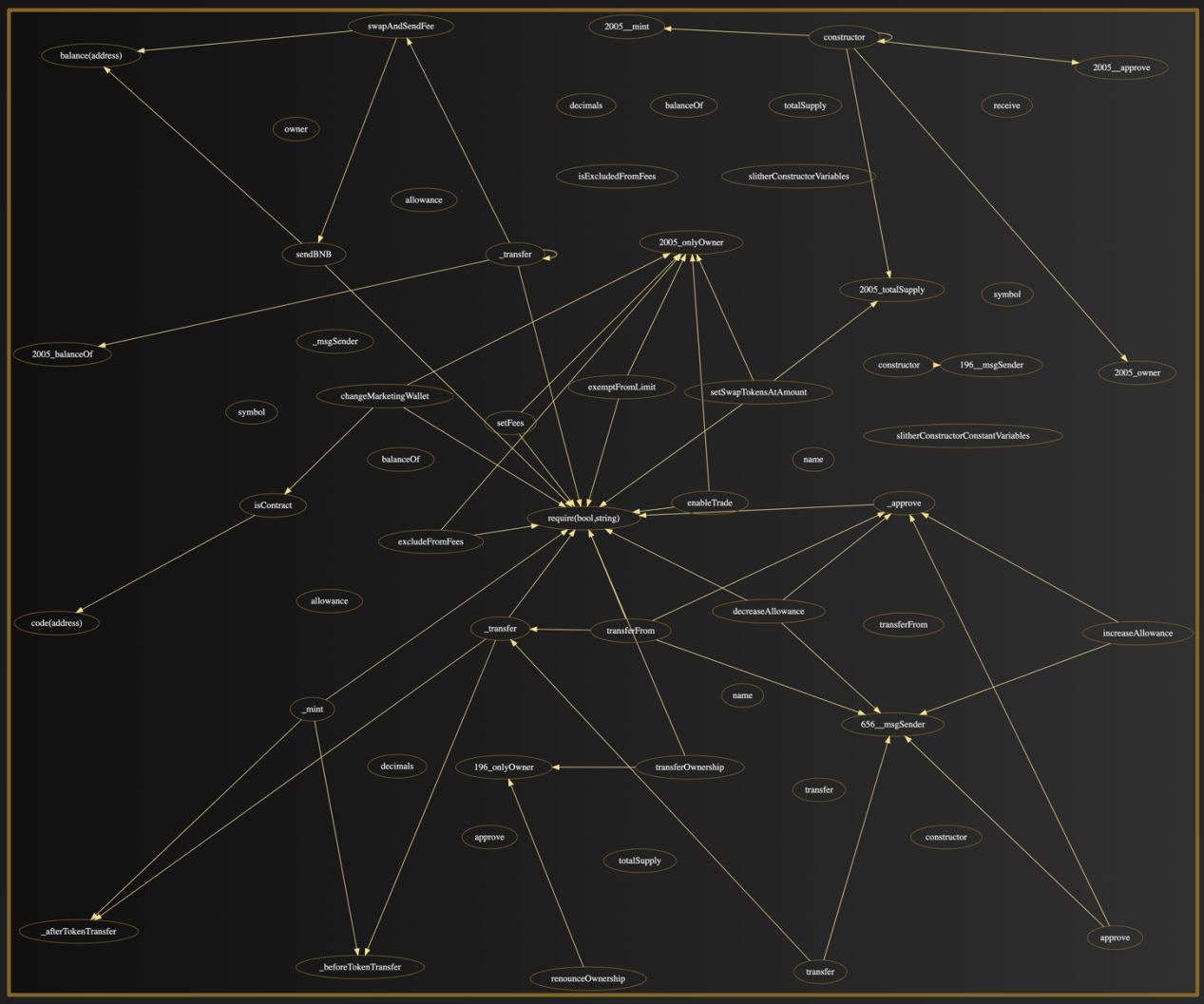
Official Links

Website	https://pongheroes.io
Twitter	https://twitter.com/pongheroes
Telegram	https://t.me/pongheroes

The Team

About	Based on their website, we found that there are 10 members in the team. However, there are no KYC procedure being conducted by Revoluzion on any of Abitoken' team members.
KYC Issuer	N/A
Member's KYC'd	N/A
KYC Date	N/A
Certificate Link	N/A
Task Completed	N/A

Contract Functions Interaction



Audit Overview

Threat Level

When conducting audit on smart contract(s), we first look for known vulnerabilities and issues within the code because any exploitation on such vulnerabilities and issues by malicious actors could potentially result in serious financial damage to the projects. All the issues and vulnerabilities will be categorized into the categories as provided below.

Critical

This category provides issues and vulnerabilities that are critical to the performance/functionality of the smart contract and should be fixed by project creator before moving to a live environment.

Medium

This category provides issues and vulnerabilities that are not that critical to the performance/functionality of the smart contract but is recommended to be fixed by project creator before moving to a live environment.

Minor

This category provides issues and vulnerabilities that are minor to the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment.

Informational

This category provides issues and vulnerability that have insignificant effect on the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment. However, fixing them can further improve the efficacy or security for features with a risk-free factor.

Notable Information

- Contract Owner cannot stop or pause transactions.
- Contract Owner cannot transfer tokens from specific address.
- Contract Owner cannot mint new tokens after deploying smart contract.
- Contract Owner cannot burn tokens from specific wallet.
- Both buy and sell fees are set to be a total of 3%.
- Both buy and sell fees can be change to a maximum total of 10%.
- Contract Owner cannot blacklist wallets from selling.
- There are no compiler warnings when compiling the smart contracts.
- Contract is using interface from safe Zeppelin modules.

Bugs and Optimizations Detection

This table is based on the result obtained from running the smart contract through Slither's Solidity static analysis.

What it detects	Impact	Confidence	Status
Storage abiencoderv2 array	High	High	Passed
transferFrom uses arbitrary from	High	High	Passed
Modifying storage array by value	High	High	Passed
The order of parameters in a shift instruction is incorrect.	High	High	Passed
Multiple constructor schemes	High	High	Passed
Contract's name reused	High	High	Passed
Detected unprotected variables	High	High	Passed
Public mappings with nested variables	High	High	Passed
Right-To-Left-Override control character is used	High	High	Passed
State variables shadowing	High	High	Passed
Functions allowing anyone to destruct the contract	High	High	Passed
Uninitialized state variables	High	High	Passed
Uninitialized storage variables	High	High	Passed

Unprotected upgradeable contract	High	High	Passed
transferFrom uses arbitrary from with permit	High	Medium	Passed
Functions that send Ether to arbitrary destinations	High	Medium	Moderated
Tainted array length assignment	High	Medium	Passed
Controlled delegatecall destination	High	Medium	Passed
Payable functions using delegatecall inside a loop	High	Medium	Passed
msg.value inside a loop	High	Medium	Passed
Reentrancy vulnerabilities (theft of ethers)	High	Medium	Moderated
Signed storage integer array compiler bug	High	Medium	Passed
Unchecked tokens transfer	High	Medium	Passed
Weak PRNG	High	Medium	Passed
Detects ERC20 tokens that have a function whose signature collides with EIP-2612's DOMAIN_SEPARATOR()	Medium	High	Passed
Detect dangerous enum conversion	Medium	High	Passed
Incorrect ERC20 interfaces	Medium	High	Passed
Incorrect ERC721 interfaces	Medium	High	Passed
Dangerous strict equalities	Medium	High	Passed

Contracts that lock ether	Medium	High	Passed
Deletion on mapping containing a structure	Medium	High	Passed
State variables shadowing from abstract contracts	Medium	High	Passed
Tautology or contradiction	Medium	High	Passed
Unused write	Medium	High	Passed
Misuse of Boolean constant	Medium	Medium	Passed
Constant functions using assembly code	Medium	Medium	Passed
Constant functions changing the state	Medium	Medium	Passed
Imprecise arithmetic operations order	Medium	Medium	Passed
Reentrancy vulnerabilities (no theft of ethers)	Medium	Medium	Passed
Reused base constructor	Medium	Medium	Passed
Dangerous usage of tx.origin	Medium	Medium	Passed
Unchecked low-level calls	Medium	Medium	Passed
Unchecked send	Medium	Medium	Passed
Uninitialized local variables	Medium	Medium	Passed
Unused return values	Medium	Medium	Passed
Modifiers that can return the default value	Low	High	Passed

Built-in symbol shadowing	Low	High	Passed
Local variables shadowing	Low	High	Passed
Uninitialized function pointer calls in constructors	Low	High	Passed
Local variables used prior their declaration	Low	High	Passed
Constructor called not implemented	Low	High	Passed
Multiple calls in a loop	Low	Medium	Passed
Missing Events Access Control	Low	Medium	Passed
Missing Events Arithmetic	Low	Medium	Passed
Dangerous unary expressions	Low	Medium	Passed
Missing Zero Address Validation	Low	Medium	Passed
Benign reentrancy vulnerabilities	Low	Medium	Passed
Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium	Moderated
Dangerous usage of block.timestamp	Low	Medium	Passed
Assembly usage	Informational	High	Passed
Assert state change	Informational	High	Passed
Comparison to boolean constant	Informational	High	Moderated
Deprecated Solidity Standards	Informational	High	Passed
Un-indexed ERC20 event parameters	Informational	High	Passed

Function initializing state variables	Informational	High	Passed
Low level calls	Informational	High	Moderated
Missing inheritance	Informational	High	Passed
Conformity to Solidity naming conventions	Informational	High	Moderated
If different pragma directives are used	Informational	High	Passed
Redundant statements	Informational	High	Passed
Incorrect Solidity version	Informational	High	Moderated
Unimplemented functions	Informational	High	Passed
Unused state variables	Informational	High	Passed
Costly operations in a loop	Informational	Medium	Passed
Functions that are not used	Informational	Medium	Passed
Reentrancy vulnerabilities through send and transfer	Informational	Medium	Passed
Variable names are too similar	Informational	Medium	Moderated
Conformance to numeric notation best practices	Informational	Medium	Moderated
State variables that could be declared constant	Optimization	High	Passed
Public function that could be declared external	Optimization	High	Passed

Contract Diagnostic

CODE	SEVERITY	DESCRIPTION
SWC-110	**Unknown**	Out of bounds array access.
BE	**Informational**	Boolean equal.
LLC	**Informational**	Low level calls.
NC	**Informational**	Naming convention.
SN	**Informational**	Similar name.
TMD	**Informational**	Too many digits.

SWC-110 — Out of bounds array access

SEVERITY	Unknown
LOCATION(S)	PongHeroes.sol#L804-806

```
803  function swapAndSendFee(uint256 tokenAmount) private {
804      address[] memory path = new address[](2);
805      path[0] = address(this);
806      path[1] = uniswapV2Router.WETH();
807
808      uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
809          tokenAmount,
810          0, // accept any amount of ETH
811          path,
812          address(this),
813          block.timestamp
814      );
815
816      uint256 newBalance = address(this).balance;
817      uint256 addressBalance = address(this).balance;
818
819      sendBNB(
820          payable(marketingWallet),
821          addressBalance
822      );
823
824      emit SwapAndSendFee(tokenAmount, newBalance);
825  }
```

DESCRIPTION	The index access expression can cause an exception in case of use of invalid array index value.
RECOMMENDATIONS	As long as project creator is careful with the index access expression to prevent an exception in case of use of invalid array index value, this should not produce any issue. No specific actions needed to be taken by project creator.
STATUS	N/A

BE — Boolean equal

SEVERITY	Informational — Minor
LOCATION(S)	PongHeroes.sol#L675

```
674  function enableTrade() external onlyOwner{
675      require(isTradeOpen == false, "Trade is already open!");
676      isTradeOpen = true;
677  }
```

DESCRIPTION	[PongHeroes.enableTrade()] (#L674-677) compares to a boolean constant at #L675
RECOMMENDATIONS	Project creator is recommended to use boolean constants directly instead of comparing to true or false. We would recommend to remove the equality to the boolean constant and directly use “!isTradeOpen” instead of “isTradeOpen == false”.
STATUS	**N/A**

LLC — Low level calls

SEVERITY	Informational — Medium
LOCATION(S)	PongHeroes.sol#L668

```
662  function sendBNB(address payable recipient, uint256 amount) internal {
663      require(
664          address(this).balance >= amount,
665          "Address: insufficient balance"
666      );
667
668      (bool success, ) = recipient.call{value: amount}("");
669      require(
670          success,
671          "Address: unable to send value, recipient may have reverted"
672      );
673  }
```

DESCRIPTION	[PongHeroes.sendBNB] (#L662-673) is using low level call at #L668.
RECOMMENDATIONS	Project creator should avoid using low-level calls. Make sure to check the call success or for code existence if the call is meant for a contract. The use of low-level calls is usually error-prone since they do not check for code existence or call success.
STATUS	**N/A**

NC — Naming convention

SEVERITY	Informational — Minor
LOCATION(S)	PongHeroes.sol#L338, 340, 365, 407, 708, 722

```
338   function DOMAIN_SEPARATOR() external view returns (bytes32);
```

```
340   function PERMIT_TYPEHASH() external pure returns (bytes32);
```

```
365   function MINIMUM_LIQUIDITY() external pure returns (uint256);
```

```
407   function WETH() external pure returns (address);
```

```
708  function setFees(uint256 _feeOnBuy, uint256 _feeOnSell) external onlyOwner {
709      require(
710          _feeOnBuy <= 10,
711          "Marketing fee on buy cannot be more than 10%"
712      );
713      require(
714          _feeOnSell <= 10,
715          "Marketing fee on sell cannot be more than 10%"
716      );
717      feeOnBuy = _feeOnBuy;
718      feeOnSell = _feeOnSell;
719      emit FeesUpdated(feeOnBuy, feeOnSell);
720  }
```

```
722  function changeMarketingWallet(address _marketingWallet)
723      external
724      onlyOwner
725  {
726      require(
727          _marketingWallet != marketingWallet,
728          " wallet is already that address"
729      );
730      require(
731          _marketingWallet != address(0),
732          " wallet cannot be the zero address"
733      );
734      require(
735          !isContract(_marketingWallet),
736          " wallet cannot be a contract"
737      );
738      marketingWallet = _marketingWallet;
739      _isExcludedFromFees[marketingWallet] = true;
740      emit MarketingWalletChanged(marketingWallet);
741  }
```

DESCRIPTION	[IUniswapV2Pair.DOMAIN_SEPARATOR()] (#L338) is not in mixedCase.

	[IUniswapV2Pair.PERMIT_TYPEHASH()] (#L340) is not in mixedCase. [IUniswapV2Pair.MINIMUM_LIQUIDITY()] (#L365) is not in mixedCase. [IUniswapV2Router01.WETH] (#L407) is not in mixedCase. [PongHeroes.setFees()._feeOnBuy] (#L708) is not in mixedCase. [PongHeroes.setFees()._feeOnSell] (#L708) is not in mixedCase. [PongHeroes.changeMarketingWallet()._marketingWallet] (#L722) is not in mixedCase.
RECOMMENDATIONS	Based on our analysis, the IUniswapV2Pair and IUniswapV2Router01 smart contract are direct forks from Uniswap. Although the name doesn't conform to the standard convention, it's still okay to leave it be to avoid from potentially breaking any external function. However, for PongHeroes smart contract, it is okay for project creator to update the name of the parameters in those functions so that they conform to the standard naming convention.
STATUS	**N/A**

SN — Similar name

SEVERITY	Informational — Minor
LOCATION(S)	PongHeroes.sol#L412, 413
DESCRIPTION	[IUniswapV2Router01.addLiquidity()] (#L409-424) has two parameters names that are too similar.

```
409  function addLiquidity(
410      address tokenA,
411      address tokenB,
412      uint256 amountADesired,
413      uint256 amountBDesired,
414      uint256 amountAMin,
415      uint256 amountBMin,
416      address to,
417      uint256 deadline
418  )
419      external
420      returns (
421          uint256 amountA,
422          uint256 amountB,
423          uint256 liquidity
424      );
```

RECOMMENDATIONS	Based on our analysis, the IUniswapV2Router01 smart contract is a direct fork from Uniswap and it is just an interface. Although their names are too similar, it's still okay to leave them be for the purpose of following the standard parameter declaration that is widely used as reference.
STATUS	N/A

TMD — Too many digits

SEVERITY	Informational — Medium
LOCATION(S)	PongHeroes.sol#L744-747

```
743  function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner {
744      require(
745          newAmount > totalSupply() / 100000,
746          "SwapTokensAtAmount must be greater than 0.001% of total supply"
747      );
748      swapTokensAtAmount = newAmount;
749      emit SwapTokensAtAmountChanged(newAmount);
750  }
```

DESCRIPTION	[PongHeroes.setSwapTokensAtAmount()] (#L743-750) uses literals with too many digits.
RECOMMENDATIONS	Literals that use too many digits are usually difficult to read and review, which makes them likely to be used incorrectly. Project creator can try to use ether suffix.
STATUS	N/A

Disclaimer

This report only shows findings based on our limited project analysis according to the good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall online presence and team transparency details of which are set out in this report. To get a full view of our analysis, **it is important for you to read the full report**. Under no circumstances did Revoluzion Audit receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. **Our team provides no guarantees against the sale of team tokens or the removal of liquidity by the project** audited in this document.

While **we have done our best to conduct thorough analysis to produce this report**, it is crucial to note that you should not rely solely on this report and use the content provided in this document as financial advice or a reason to buy any investment. The Our team disclaims any liability against us for the resulting losses based on the you decision made by relying on the content of this report. **You must conduct your own independent investigations before making any decisions** to protect yourselves from being scammed. We go into more detail on this in the disclaimer clause in the next page — please make sure to read it in full.

Full Disclaimer Clause

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you. This report is provided for information purposes only, on a non-reliance basis and does not constitute to any investment advice.

No one shall have any right to rely on the report or its contents, and Revoluzion Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (collectively known as Revoluzion) owe no duty of care towards you or any other person, nor do we make any warranty or representation to any person on the accuracy or completeness of the report.

The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Revoluzion hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report.

Except and only to the extent that it is prohibited by law, Revoluzion hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Revoluzion, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts, website, social media, and team.