AuditAce

FROM INCEPTION TO SUCCESS

Pirate Plunder

Smart Contract Audit Report

ABOUT AUDITACE

Audit Ace is built, to combat financial fraud in the cryptocurrency industry, a growing security firm that provides audits, Smart contract creation, and end-to-end solutions to all crypto-related queries.

Website - https://auditace.tech/

Telegram - https://t.me/Audit_Ace

Twitter - https://twitter.com/auditace_

Github - https://github.com/Audit-Ace

Overview

AUDITACE team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks.

Audit Result: Passed with Low Risk

Audit Date: December 10, 2022

KYC:Not done till date of Audit

Audit Team: TEAM AUDITACE

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.

Used Tools

Manual Review - Forked Pancakeswap V2 on Local Blockchain

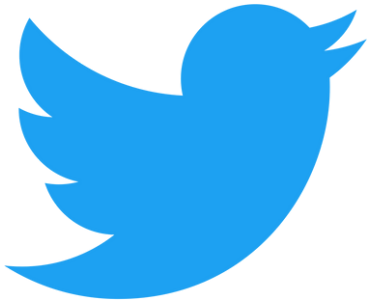
Tests:

- Deployment
- Adding Liquidity
- Buying and selling for everyone, right after adding liquidity
- Auto Liquidity
- Marketing & Dev wallet receiving BNB

Pirate Plunder

Social Media Overview

https://t.me/piratesonbsc

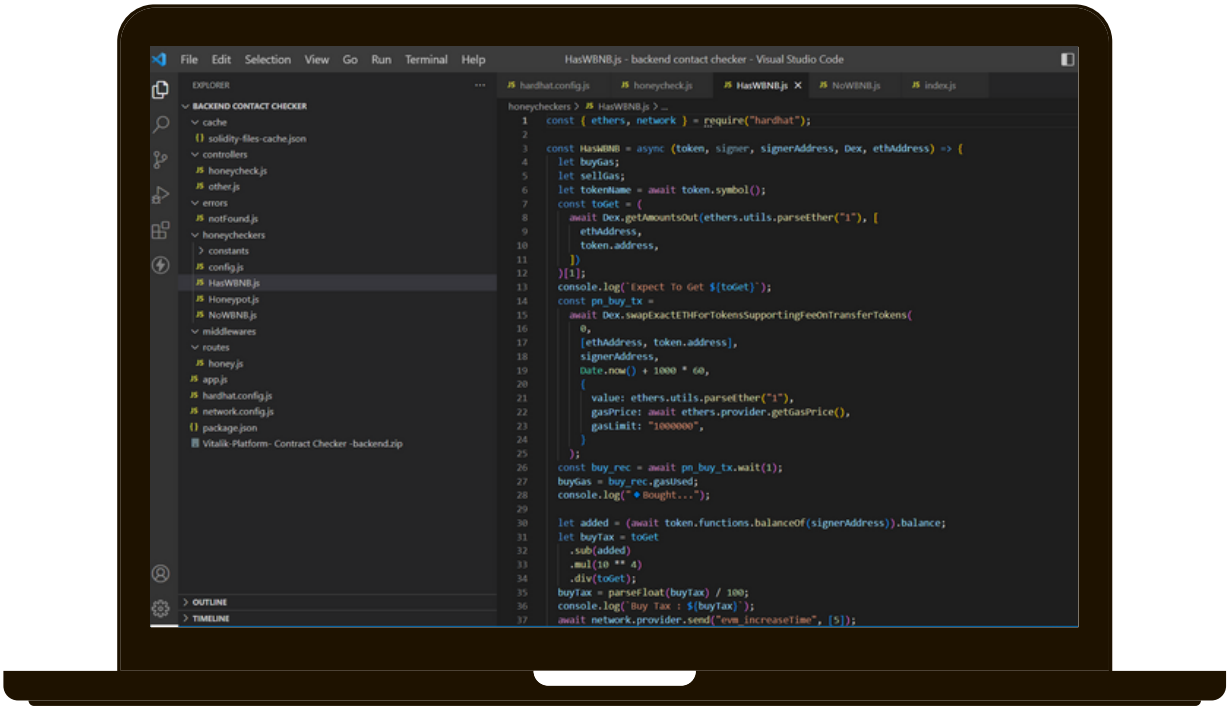


https://twitter.com/piratesonbsc

Token Summary

Parameter	Result
Address	0x24175F4B4A3f248f1127528b641401E865FaC110
Token Type	BEP 20
Contract Checksum	189f40034be7a199f1fa9891668ee3ab6049f82d38c68be70f596eab2e1857b7
Decimals	18
Supply	777,777,777,777
Platform	Binance Smart Chain
Compiler	v0.8.9+commit.e5eed63a
Token Name	Pirates Plunder
Symbol	PIRATE
License Type	Unilicense
Language	Solidity

CONTRACT FUNCTION SUMMARY



Can edit Tax?

DETECTED

Can take back Ownership?

NOT DETECTED

Is Blacklisted?

NOT DETECTED

Is Whitelisted?

NOT DETECTED

Is Mintable?

NOT DETECTED

Disable Trade?

NOT DETECTED

Is Trading with CooldownTime?

NOT DETECTED

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.

Issues Checking Status

No	Issue Description	Checking Status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Design Logic.	Passed
12	Cross-function race conditions.	Passed
13	Safe Zeppelin module.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Fallback function security.	Passed
17	Arithmetic accuracy.	Passed

SWC ATTACK TEST

SWC ID	Description	Test Result
SWC-100	Function Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed

SWC ID	Description	Test Result
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Grieving	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions with Multiple Variable Length Arguments	Passed
SWC-134	Unencrypted Private Data On-Chain	Passed

Inheritance Tree

Summary

- Owner is not able to set buy+sell taxes over 20%
- Owner is able to set max buy/sell/transfer amount but not less than 0.1% of total supply
- Owner is able to set a max wallet amount but not less than 0.5% of total supply
- Owner is able to set transfer/buy delays (1 buy or trade per block)
- Owner is not able to pause trades
- Owner is not able to mint new tokens
- Owner is not able to blacklist an arbitrary address

Classification of Risks

Severity	Description
◆ 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
◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	5
◆ Gas Optimization / Suggestions	2

MANUAL AUDIT

Low Risk Findings

Logical - calling enableTrading resets launchedAt variable to current block number

Suggestions : make sure you are not able to call enableTrading again after trading is enabled

Logical – bad anti-bot implementation

```
 if (block.number <= (launchedAt) &&
        to != uniswapV2Pair &&
        to !=
address(0x10ED43C718714eb63d5aA57B78B54704E256024E)
        ) {
        _blacklist[to] = false;
}
```

this block of code never will be executed, since block.number can never be less than launchedAt (launchedAt = launch block), assuming that block.number = launchedAt, setting _blacklist[to] to false is redundant (since blacklist[to] is set to false by default)

MANUAL AUDIT

Centralization -Owner is able to set a max sell/buy/transfer amount but not less than 0.1% of total supply

```
function updateMaxTxnAmount(uint256 newNum) external
onlyOwner {
require(newNum >= (totalSupply() * 1 / 1000)/1e18, "Cannot set
maxTransactionAmount lower than 0.1%");
maxTransactionAmount = newNum * (10**18);
}
```

Centralization -Owner is able to set a max wallet amount, this max amount can not be less than 0.5% of total supply

```
function updateMaxWalletAmount(uint256 newNum) external
onlyOwner {
require(newNum >= (totalSupply() * 5 / 1000)/1e18, "Cannot set
maxWallet lower than 0.5%");
maxWallet = newNum * (10**18);
}
```

MANUAL AUDIT

Centralization -Owner is able to change buy and sell fees, buy + sell fees can not exceed 20%

```
function updateFees(uint256 _buyMarketingFee, uint256
_buyLiquidityFee, uint256 _buyDevFee, uint256 _sellMarketingFee,
uint256 _sellLiquidityFee, uint256 _sellDevFee) external
onlyOwner {
buyMarketingFee = _buyMarketingFee;
buyLiquidityFee = _buyLiquidityFee;
buyDevFee = _buyDevFee;

sellMarketingFee = _sellMarketingFee;
sellLiquidityFee = _sellLiquidityFee;
sellDevFee = _sellDevFee;

totalBuySellFees = buyMarketingFee + buyLiquidityFee +
buyDevFee + sellMarketingFee + sellLiquidityFee + sellDevFee;
require(totalBuySellFees <= 20, "Must keep fees at 20% or less");
}
```

MANUAL AUDIT

Gas Optimizations

- Redundant function and code: manageBot function is never used in the contract and is also set to false, also, anti-bot implementation is not practical (refer to first logical issue) and can be deleted from the contract

- holdersFirstBuyTimestamp variable is redundant and never used in the contract

Suggestions

- since compiler version is more than 0.8.0, then we can ignore using safeMath to increase gas efficiency