



FAIRYPROOF

Pink BNB Token

AUDIT REPORT

Version 1.0.0

Serial No. 2023030400012017

Presented by Fairyproof

March 4, 2023

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the Pink BNB Token Issuance project.

Audit Start Time:

March 3, 2023

Audit End Time:

March 4, 2023

Audited Source File's Address:

<https://bscscan.com/address/0xf5BDe7Eb378661F04C841B22bA057326B0370153#code>

The goal of this audit is to review Pink BNB's solidity implementation for its Token Issuance function, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the Pink BNB team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code Review, Including:

- Project Diagnosis

Understanding the size, scope and functionality of your project's source code based on the specifications, sources, and instructions provided to Fairyproof.

- Manual Code Review

Reading your source code line-by-line to identify potential vulnerabilities.

- Specification Comparison

Determining whether your project's code successfully and efficiently accomplishes or executes its functions according to the specifications, sources, and instructions provided to Fairyproof.

2. Testing and Automated Analysis, Including:

- Test Coverage Analysis

Determining whether the test cases cover your code and how much of your code is exercised or executed when test cases are run.

- Symbolic Execution

Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.

3. Best Practices Review

Reviewing the source code to improve maintainability, security, and control based on the latest established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following source(s) of truth about how the token issuance function should work:

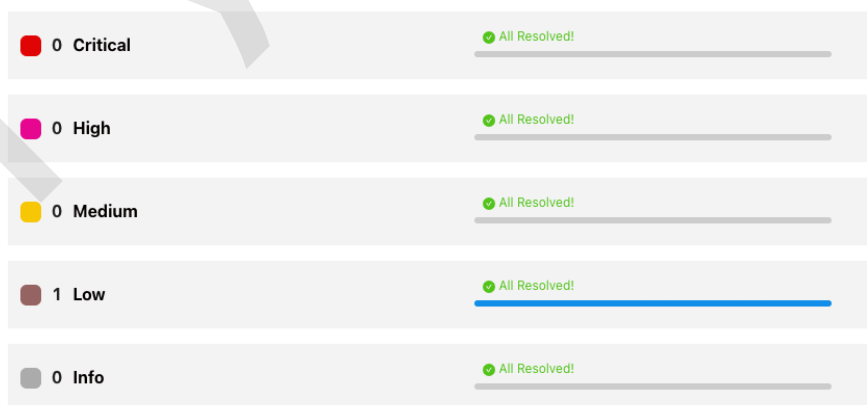
Website: <https://pnb.world/>

Source Code: <https://bscscan.com/address/0xf5BDe7Eb378661F04C841B22bA057326B0370153#code>

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Pink BNB team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2023030400012017	Fairyproof Security Team	Mar 3, 2023 - Mar 4, 2023	Passed



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, one issue of low-severity was uncovered. The Pink BNB team has fixed this issue.

02. About Fairyproof

[Fairyproof](#) is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Introduction to Pink BNB

It's designed to collect large amounts of data about the cryptocurrency industry, label training data and analyze the data for correlations and patterns, and use these patterns to make predictions about future states.

This system consists of 3 major layers, user interface(UI), crawler, and Artificial intelligence(AI). Each layer has different roles to give users the best and fastest experience.

The above description is quoted from relevant documents of Pink BNB.

04. Major functions of audited code

The audited code mainly implements a token issuance function. Here are the details:

- Blockchain: BNB Chain
- Token Standard: ERC20
- Token Address: 0xf5BDe7Eb378661F04C841B22bA057326B0370153
- Token Name: Pink BNB
- Token Symbol: PNB
- Decimals: 18
- Current Supply: 100,000,000,000,000
- Max Supply: 100,000,000,000,000

Note:

1% tax will be charged based on the amount transferred when tokens are transferred.
0.2% tax on the transferred amount is sent to the zero address and 0.8% tax on the transferred amount is sent to the team's address.

05. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Access Control
- Admin Rights
- Arithmetic Precision
- Code Improvement
- Contract Upgrade/Migration
- Delete Trap
- Design Vulnerability
- DoS Attack
- EOA Call Trap
- Fake Deposit
- Function Visibility
- Gas Consumption
- Implementation Vulnerability
- Inappropriate Callback Function
- Injection Attack
- Integer Overflow/Underflow
- IsContract Trap
- Miner's Advantage
- Misc
- Price Manipulation
- Proxy selector clashing
- Pseudo Random Number
- Re-entrancy Attack
- Replay Attack
- Rollback Attack
- Shadow Variable

- Slot Conflict
- Token Issuance
- Tx.origin Authentication
- Uninitialized Storage Pointer

06. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Informational is not an issue or risk but a suggestion for code improvement.

07. Major areas that need attention

Based on the provided source code the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Function Implementation

We checked whether or not the functions were correctly implemented.

We found one issue, for more details please refer to [FP-1] in "09. Issue description".

- Access Control

We checked each of the functions that could modify a state, especially those functions that could only be accessed by owner or administrator

We didn't find issues or risks in these functions or areas at the time of writing.

- Token Issuance & Transfer

We examined token issuance and transfers for situations that could harm the interests of holders.

We didn't find issues or risks in these functions or areas at the time of writing.

- State Update

We checked some key state variables which should only be set at initialization.

We didn't find issues or risks in these functions or areas at the time of writing.

- Asset Security

We checked whether or not all the functions that transfer assets were safely handled.


We didn't find issues or risks in these functions or areas at the time of writing.

- Miscellaneous

We checked the code for optimization and robustness.

We didn't find issues or risks in these functions or areas at the time of writing.

08. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	Owner Can Recover After Abandonment under Certain Conditions	Admin Rights	Low	 Fixed

09. Issue descriptions

[FP-1] Owner Can Recover After Abandonment under Certain Conditions

Admin Rights

Low

✓ Fixed

Issue/Risk: Admin Rights

Description:

The function `abandonOwnership` is used to abandon the ownership, but the function does not reset the value of `_NEW_OWNER`. If `_NEW_OWNER` has a valid value(address), the address can actually recover the ownership by calling the `claimOwnership` function even after the ownership is abandoned. Considering that the owner is only used to set the team's address to receive the tax of transferring tokens, it has no effect on regular users, the risk can be avoided through careful operations, so the risk is marked as low-severity.

Recommendation:

Since the contract has been deployed, it is recommended to call the `transferOwnership` function to reset `_NEW_OWNER` to zero before calling the `abandonOwnership` function.

Update/Status:

The Pink BNB team has revoked the ownership and the issue will never happen.

10. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- N/A

11. Appendices

11.1 Unit Test

1. PinkBNB.t.js

```

1  const { expect, assert } = require("chai");
2  const { ethers } = require("hardhat");
3
4
5  describe("Unit test of PinkBNT token deployed on BNB Chain", function () {
6      let owner,user1,user2,users;
7      let instance;
8      const init_supply = ethers.utils.parseEther("100000000000000");
9      const ZERO_ADDRESS = ethers.constants.AddressZero;
10     const BURN_TAX_RATE = 20;
11     const TRADE_TAX_RATE = 80;
12
13     async function deployTokensAndInit() {
14         [owner, user1, user2, ...users] = await ethers.getSigners();
15         const CustomERC20 = await ethers.getContractFactory("CustomERC20");
16         instance = await CustomERC20.deploy();
17         let args = [
18             owner.address,init_supply,"Pink BNB",
19             "PNB",18,BURN_TAX_RATE,TRADE_TAX_RATE,users[0].address
20         ]
21         await instance.init(...args)
22     }
23     beforeEach(async () =>{
24         await deployTokensAndInit();
25     });
26
27
28     describe("Initial state unit test", () => {
29         it("Call init twice should be failed", async () => {
30             let args = [
31                 owner.address,init_supply,"Pink BNB",
32                 "PNB",18,BURN_TAX_RATE,TRADE_TAX_RATE,users[0].address
33             ]
34             await
35             expect(instance.init(...args)).to.be.revertedWith("DODO_INITIALIZED");
36             await
37             expect(instance.initOwner(user1.address)).to.be.revertedWith("DODO_INITIALIZED");
38         });
39     });

```

```

37     it("Initial state should equal with the params of constructor"), async ()
=> {
38         expect(await instance._OWNER_()).to.be.equal(owner.address);
39         expect(await instance._NEW_OWNER_()).to.be.equal(ZERO_ADDRESS);
40         expect(await instance.name()).to.be.equal("Pink BNB");
41         expect(await instance.symbol()).to.be.equal("PNB");
42         expect(await instance.decimals()).to.be.equal(18);
43         expect(await instance.tradeBurnRatio()).to.be.equal(BURN_TAX_RATE);
44         expect(await instance.tradeFeeRatio()).to.be.equal(TRADE_TAX_RATE);
45         expect(await instance.team()).to.be.equal(users[0].address);
46         expect(await
instance.balanceOf(owner.address)).to.be.equal(init_supply);
47         expect(await instance.totalSupply()).to.be.equal(init_supply);
48     });
49 });
50
51 describe("Owner and OnlyOwner unit test", () => {
52     it("Only owner can transfer ownership", async () => {
53         await
expect(instance.connect(user1).transferOwnership(user1.address)).to.be.revertedWith("NOT_OWNER");
54         await expect(instance.transferOwnership(user1.address)).to.be.emit(
55             instance, "OwnershipTransferPrepared"
56         ).withArgs(owner.address, user1.address);
57     });
58
59     it("TransferOwnership can reset _NEW_OWNER_", async () => {
60         await instance.transferOwnership(user1.address);
61         expect(await instance._NEW_OWNER_()).to.be.equal(user1.address);
62         await instance.transferOwnership(ZERO_ADDRESS);
63         expect(await instance._NEW_OWNER_()).to.be.equal(ZERO_ADDRESS);
64     });
65
66     it("Only new owner can claim ownership", async () => {
67         await instance.transferOwnership(user1.address);
68         expect(await instance._NEW_OWNER_()).to.be.equal(user1.address);
69         await
expect(instance.connect(user2).claimOwnership()).to.be.revertedWith("INVALID_CLAIM");
70         await expect(instance.connect(user1).claimOwnership()).to.be.emit(
71             instance, "OwnershipTransferred"
72         ).withArgs(owner.address, user1.address);
73         expect(await instance._OWNER_()).to.be.equal(user1.address);
74         expect(await instance._NEW_OWNER_()).to.be.equal(ZERO_ADDRESS);
75     });
76
77     it("AbandonOwnership should set owner to zero address", async () => {

```

```

78         await
expect(instance.abandonOwnership(user2.address)).to.be.revertedWith("NOT_ZERO_ADDRESS");
79         await expect(instance.abandonOwnership(ZERO_ADDRESS)).to.be.emit(
80             instance, "OwnershipTransferred"
81         ).withArgs(owner.address, ZERO_ADDRESS);
82         expect(await instance._OWNER()).to.be.equal(ZERO_ADDRESS);
83     });
84
85     it("AbandonOwnership should reset the value of _NEW_OWNER_", async () =>
{
86         // transfer ownership
87         await instance.transferOwnership(user1.address);
88         // AbandonOwnership
89         await instance.abandonOwnership(ZERO_ADDRESS);
90         expect(await instance._OWNER()).to.be.equal(ZERO_ADDRESS);
91         expect(await instance._NEW_OWNER()).to.be.equal(user1.address);
92         await instance.connect(user1).claimOwnership();
93         expect(await instance._OWNER()).to.be.equal(ZERO_ADDRESS);
94     });
95 });
96
97 describe("changeTeamAccount unit test", () => {
98     it("Should change state and emit event", async () => {
99         await expect(instance.changeTeamAccount(users[1].address)).to.be.emit(
100             instance, "ChangeTeam"
101         ).withArgs(users[0].address, users[1].address);
102         expect(await instance.team()).to.be.equal(users[1].address);
103     });
104 });
105
106 describe("Transfer test", () => {
107     it("Transfer to zero address should be failed", async () => {
108         await
expect(instance.transfer(ZERO_ADDRESS, 100)).to.be.revertedWith("ERC20: transfer to
the zero address");
109     });
110
111     it("Transfer zero value should always be successful", async () => {
112         expect(await instance.balanceOf(user1.address)).to.be.equal(0);
113         await instance.connect(user1).transfer(user2.address, 0);
114     });
115
116     it("Transfer token beyond balance should be failed", async () => {
117         await
expect(instance.connect(user1).transfer(user2.address, 10)).to.be.revertedWith(
118             "ERC20: transfer amount exceeds balance"
119         );
120     });

```

```

121
122     it("Transfer token should be taxed", async () => {
123         let amount = 1000000;
124         let burn_fee = 1000000 * BURN_TAX_RATE / 10000;
125         let trade_fee = 1000000 * TRADE_TAX_RATE / 10000;
126         await expect(instance.transfer(user1.address, amount)).to.be.emit(
127             instance, "Transfer"
128         ).withArgs(owner.address, user1.address, amount - burn_fee - trade_fee);
129         expect(await
instance.balanceOf(owner.address)).to.be.equal(init_supply.sub(amount));
130         expect(await instance.balanceOf(user1.address)).to.be.equal(amount -
burn_fee - trade_fee);
131         expect(await instance.balanceOf(ZERO_ADDRESS)).to.be.equal(burn_fee);
132         expect(await
instance.balanceOf(users[0].address)).to.be.equal(trade_fee);
133     });
134
135     it("Transfer to self should only be taxed", async () => {
136         await instance.transfer(user1.address, 100000);
137         let balance_before = 100000 * 99 / 100;
138         expect(await
instance.balanceOf(user1.address)).to.be.equal(balance_before);
139         let amount = 1000;
140         await instance.connect(user1).transfer(user1.address, amount);
141         let balance_after = await instance.balanceOf(user1.address);
142         expect(balance_after).to.be.equal(balance_before - amount / 100);
143     });
144 })
145
146 describe("Approve and transferFrom unit test", () => {
147     it("Approve should change allowance and change state", async () => {
148         expect(await
instance.allowance(owner.address, user1.address)).to.be.equal(0);
149
150         await expect(instance.approve(user1.address, 1000)).to.be.emit(
151             instance, "Approval"
152         ).withArgs(owner.address, user1.address, 1000);
153
154         expect(await
instance.allowance(owner.address, user1.address)).to.be.equal(1000);
155     });
156
157     it("Transfer from should change allowance", async () => {
158         await
expect(instance.connect(user1).transferFrom(owner.address, user1.address, 1000)).to.
be.revertedWith(
159             "ALLOWANCE_NOT_ENOUGH"
160         );
161         await instance.approve(user1.address, 10000);

```

```

162         await
expect(instance.connect(user1).transferFrom(owner.address, user1.address, 1000)).to.
be.emit(
163         instance, "Transfer"
164     ).withArgs(owner.address, user1.address, 1000 * 99 / 100);
165     expect(await
instance.allowance(owner.address, user1.address)).to.be.equal(10000 - 1000);
166     });
167 });
168 });
169

```

2. UnitTestOutput

```

1  Unit test of PinkBNT token deployed on BNB Chain
2  Initial state unit test
3      ✓ Call init twice should be failed (62ms)
4      ✓ Initial state should be equal with the params of constructor (65ms)
5  Owner and OnlyOwner unit test
6      ✓ Only owner can transfer ownership
7      ✓ TransferOwnership can reset _NEW_OWNER_
8      ✓ Only new owner can claim ownership (52ms)
9      ✓ AbandonOwnership should set owner to zero address
10     1) AbandonOwnership should reset the value of _NEW_OWNER_
11  changeTeamAccount unit test
12     ✓ Should change state and emit event
13  Transfer test
14     ✓ Transfer to zero address should be failed
15     ✓ Transfer zero value should always be successful
16     ✓ Transfer token beyond balance should be failed
17     ✓ Transfer token should be taxed
18     ✓ Transfer to self should only be taxed
19  Approve and transferFrom unit test
20     ✓ Approve should change allowance and change state
21     ✓ Transfer from should change allowance (42ms)
22
23
24  14 passing (2s)
25  1 failing
26
27  1) Unit test of PinkBNT token deployed on BNB Chain
28     Owner and OnlyOwner unit test
29     AbandonOwnership should reset the value of _NEW_OWNER_:
30
31     AssertionError: expected '0x70997970C51812dc3A010C7d01b50e0d17d...' to equal
'0x00000000000000000000000000000000...'
32     + expected - actual

```

```

33
34     -0x70997970C51812dc3A010C7d01b50e0d17dc79C8
35     +0x0000000000000000000000000000000000000000000000000000000000000000
36
37     at Context.<anonymous> (test/PinkBNB.t.js:93:52)
38     at processTicksAndRejections (internal/process/task_queues.js:95:5)
39     at runNextTicks (internal/process/task_queues.js:64:3)
40     at listOnTimeout (internal/timers.js:524:9)
41     at processTimers (internal/timers.js:498:7)
42

```

11.2 External Functions Check Points

1. PinkBNB.md

File: contracts/PinkBNB.sol

(Empty elements in the table represent things that are not required or relevant)

contract: CustomERC20 is InitializableOwnable

Index	Function	Visibility	Permission Check	Re-entrancy Check	Injection Check	Unit Test	Notes
1	init(address,uint256,string,string,uint8,uint256,uint256,address)	public				Passed	notInitialized
2	transfer(address,uint256)	public				Passed	
3	balanceOf(address)	public				Passed	View
4	transferFrom(address,address,uint256)	public				Passed	
5	approve(address,uint256)	public				Passed	
6	allowance(address,address)	public				Passed	View
7	changeTeamAccount(address)	external	onlyOwner			Passed	
8	abandonOwnership(address)	external	onlyOwner			Failed	
9	initOwner(address)	public				Passed	notInitialized
10	transferOwnership(address)	public	onlyOwner			Passed	
11	claimOwnership()	public	onlyNewOwner			Passed	



-  <https://medium.com/@FairyproofT>
-  <https://twitter.com/FairyproofT>
-  <https://www.linkedin.com/company/fairyproof-tech>
-  https://t.me/Fairyproof_tech
-  [Reddit: https://www.reddit.com/user/FairyproofTech](https://www.reddit.com/user/FairyproofTech)

