expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

BroToken

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	2
Low	1
Informational	0

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner needs to enable trading ?	Not Detected
Can Owner Disable Trades ?	Not Detected
Can Owner Mint ?	Not Detected
Can Owner Blacklist ?	Not Detected
Can Owner set Max Wallet amount ?	Not Detected
Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	**Passed with High Risk**
Audit Date	**07 December 2024**

CONTRACT DETAILS

Token Address: -

Name: BroToken

Symbol: -

Decimals: -

Network: -

Token Type: ERC-20

Owner: -

Deployer: --

Token Supply: -

Checksum: Ge056c616934aeb47e6038f76b20d003

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- Manual Review: The code has undergone a line-by-line review by the Ace team.
- 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.
- Slither: The code has undergone static analysis using Slither.

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Issues on this level are minor details and warning that can remain unfixed.

POINTS TO NOTE

- The owner can withdraw Avax.
- The owner can transfer/approve iERC20.
- The owner execute transfer and safe transer iERC721.

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

HIGH RISK FINDING

Centralization – Missing non-re-entrant check

Severity: High

Function: buyPresale

Status: Open

Overview:

1. The risk of reentrancy is high if the contract handles significant funds and includes functions like withdrawal, claim, or similar operations that modify the state after sending funds.

2. If the state is updated after an external call, an attacker could repeatedly exploit the vulnerability to drain the contract's funds.

```
function buyPresale(address buyer_) public payable {
    _buyPresale(msg.value, buyer_);
}
```

Suggestion:

Use the ReentrancyGuard contract from OpenZeppelin or implement a custom non-reentrant modifier.

MEDIUM RISK FINDING

Centralization – Missing access control
Severity: Medium
Function: airdropBuyers.
Status: Open

Overview:
Missing access control in smart contracts can lead to catastrophic outcomes, such as loss of funds, unauthorized control, or disruption of operations/tokens.

```
function airdropBuyers() external {
    require(lpSeeded, "LP must be seeded before airdrop can start");
    require(!airdropCompleted, "Airdrop has already been completed");
    require(block.timestamp >= AIRDROP_TIME, "It is not yet time to airdrop the presale buyer's tokens");
    _airdrop(100); //100 max transfers per tx
}
```

```
function airdropBuyers(uint256 maxTransfers_) external {
    require(lpSeeded, "LP must be seeded before airdrop can start");
    require(!airdropCompleted, "Airdrop has already been completed");
    require(block.timestamp >= AIRDROP_TIME, "It is not yet time to airdrop the presale buyer's tokens");
    _airdrop(maxTransfers_);
}
```

Suggestion:
Restrict sensitive functions to specific roles, such as the owner or admin.

MEDIUM RISK FINDING

Centralization – Owner can transfer any arbitrary token.

Severity: Medium

Function: iERC20TransferFrom, iERC20Transfer, iERC721TransferFrom, iERC721SafeTransferFrom, iERC721Transfer, iERC721SafeTransfer.

Status: Open

Overview:

The "Can Transfer Any Arbitrary Token" bug refers to a vulnerability where a function in a smart contract allows an unauthorized party to transfer any token held by the contract to an arbitrary address. This typically arises due to improper input validation or lack of access control.

```
function iERC20TransferFrom(address contract_, address to_, uint256 amount_)
external onlyOwner afterAirdrop {
    (bool success) = IERC20Token(contract_).transferFrom(address(this), to_,
amount_);
    require(success, 'token transfer from sender failed');
    emit TokenWithdraw(to_, amount_, contract_);
  }

 function iERC20Transfer(address contract_, address to_, uint256 amount_)
external onlyOwner afterAirdrop {
    (bool success) = IERC20Token(contract_).transfer(to_, amount_);

    require(success, 'token transfer from sender failed');
    emit TokenWithdraw(to_, amount_, contract_);
  }

 function iERC721TransferFrom(address contract_, address to_, uint256 tokenId_)
external onlyOwner afterAirdrop {
```

MEDIUM RISK FINDING

```
IERC721Token(contract_).transferFrom(address(this), to_, tokenId_);
        emit NFTWithdraw(to_, tokenId_, contract_);
    }

    function iERC721SafeTransferFrom(address contract_, address to_, uint256
tokenId_) external onlyOwner afterAirdrop {
        IERC721Token(contract_).safeTransferFrom(address(this), to_, tokenId_);
//Can only transfer from our own contract
        emit NFTWithdraw(to_, tokenId_, contract_);
    }

    function iERC721Transfer(address contract_, address to_, uint256 tokenId_)
external onlyOwner afterAirdrop {
IERC721Token(contract_).transfer( address(this), to_, tokenId_); //Can only transfer
from our own contract
        emit NFTWithdraw(to_, tokenId_, contract_);
    }

    function iERC721SafeTransfer(address contract_, address to_, uint256 tokenId_)
external onlyOwner afterAirdrop {
IERC721Token(contract_).safeTransfer( address(this), to_, tokenId_); //Can only
transfer from our own contract
        emit NFTWithdraw(to_, tokenId_, contract_);
    }

}
```

Suggestion:

1. Ensure the token address is one of the expected tokens that the contract is designed to handle.

2. Proper access control, validation of inputs, and whitelisting are critical measures to mitigate this risk.

LOW RISK FINDING

Centralization – Redundant code

Severity: Low

Subject: Fallback.

Status: Open

Overview:

There are unnecessary, duplicate, or unused portions of code in a smart contract. This issue, while not always directly exploitable, can lead to inefficiencies, increased gas costs, larger contract sizes, and greater complexity, making the contract harder to audit and maintain. Redundant code can also inadvertently introduce vulnerabilities or unintended behaviours.

```
 fallback() external payable { //This function is used to receive AVAX from users for the presale
      _buyPresale(msg.value, msg.sender);
  }
```

Suggestion:

Consolidate repeated logic into a single function and reuse it across the contract.

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

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 guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee 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 Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway 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 Expelee team disclaims any liability for the resulting losses.