



Building the Futuristic **Blockchain Ecosystem**

SECURITY AUDIT REPORT

TrumpStaking

TOKEN OVERVIEW

Risk Findings

Severity	Found
● High	0
● Medium	0
● Low	3
● 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

02	Token Overview	_____
03	Table of Contents	_____
04	Overview	_____
05	Contract Details	_____
06	Audit Methodology	_____
07	Vulnerabilities Checklist	_____
08	Risk Classification	_____
09	Inheritance Trees	_____
10	Static analysis	_____
11	Testnet Version	_____
12	Manual Review	_____
17	About Expelee	_____
18	Disclaimer	_____

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
Audit Date	15 June 2024

CONTRACT DETAILS

Token Address: 0xdfcf7265fbf92308Dea54adB2CAe565EeF76033c

Name: -

Symbol: -

Decimals: -

Network: EtherScan

Token Type: ERC-20

Owner: -

Deployer: 0x6f340882225Ad435904F50ea8C3a8a5b66d96dEB

Token Supply: -

Checksum: Ac6659e84744e0102ab19c1dle78a253

Testnet:

<https://testnet.bscscan.com/address/0x2c371bf6057f4357166c9fde9c8bc3fdea14b8e7#code>

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 access 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.

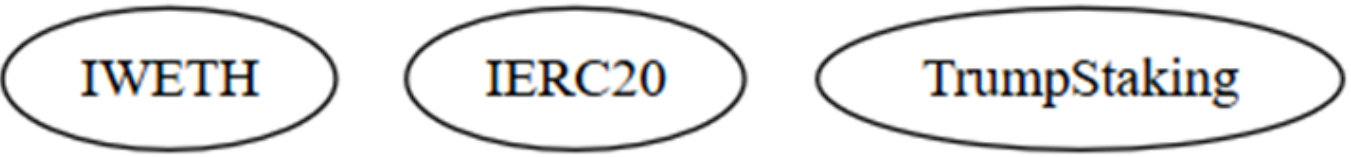
Low Risk

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

Informational

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

INHERITANCE TREE



IWETH

IERC20

TrumpStaking

STATIC ANALYSIS

```

INFO:Detectors:
TrumpStaking.withdraw(uint256,uint256,address) (TrumpStaking.sol#117-125) ignores return value by poolInfo.token.transfer(user_,amount_) (TrumpStaking.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
TrumpStaking.userPointsInPool(address,uint256) (TrumpStaking.sol#139-146) performs a multiplication on the result of a division:
- userStakedAmount / poolInfo.minAmount * poolInfo.unitPoints (TrumpStaking.sol#144)
TrumpStaking.userPoints(address) (TrumpStaking.sol#148-164) performs a multiplication on the result of a division:
- points += userStakedAmount / poolInfo.minAmount * poolInfo.unitPoints (TrumpStaking.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in TrumpStaking.deposit(uint256,uint256) (TrumpStaking.sol#81-110):
  External calls:
  - transferFrom(poolInfo.token,msg.sender,address(this),amount_) (TrumpStaking.sol#86)
    - require(bool,string)(token.transferFrom(sender,recipient,amount),Transfer from failed) (TrumpStaking.sol#114)
  - IETH(address(WETH)).deposit(value: msg.value)() (TrumpStaking.sol#90)
  - transferFrom(poolInfo.token,msg.sender,address(this),amount_) (TrumpStaking.sol#94)
    - require(bool,string)(token.transferFrom(sender,recipient,amount),Transfer from failed) (TrumpStaking.sol#114)
  External calls sending eth:
  - IETH(address(WETH)).deposit(value: msg.value)() (TrumpStaking.sol#90)
  State variables written after the call(s):
  - _depositUsersAmount ++ (TrumpStaking.sol#107)
  - _userDeposited[msg.sender] = 1 (TrumpStaking.sol#104)
  - _userStakedAmount[pid_][msg.sender] += amount_ (TrumpStaking.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in TrumpStaking.deposit(uint256,uint256) (TrumpStaking.sol#81-110):
  External calls:
  - transferFrom(poolInfo.token,msg.sender,address(this),amount_) (TrumpStaking.sol#86)
    - require(bool,string)(token.transferFrom(sender,recipient,amount),Transfer from failed) (TrumpStaking.sol#114)
  - IETH(address(WETH)).deposit(value: msg.value)() (TrumpStaking.sol#90)
  - transferFrom(poolInfo.token,msg.sender,address(this),amount_) (TrumpStaking.sol#94)
    - require(bool,string)(token.transferFrom(sender,recipient,amount),Transfer from failed) (TrumpStaking.sol#114)
  External calls sending eth:
  - IETH(address(WETH)).deposit(value: msg.value)() (TrumpStaking.sol#90)
  Event emitted after the call(s):
  - Deposit(msg.sender,pid_,amount_) (TrumpStaking.sol#101)
Reentrancy in TrumpStaking.emergencyWithdraw(uint256) (TrumpStaking.sol#132-137):
  External calls:
  - iWithdraw(pid_,amount,msg.sender) (TrumpStaking.sol#135)
    - poolInfo.token.transfer(user_,amount_) (TrumpStaking.sol#124)
  Event emitted after the call(s):
  - EmergencyWithdraw(msg.sender,pid_,amount) (TrumpStaking.sol#136)
Reentrancy in TrumpStaking.withdraw(uint256,uint256) (TrumpStaking.sol#127-130):
  External calls:
  - iWithdraw(pid_,amount,msg.sender) (TrumpStaking.sol#128)
    - poolInfo.token.transfer(user_,amount_) (TrumpStaking.sol#124)
  Event emitted after the call(s):
  - Withdraw(msg.sender,pid_,amount_) (TrumpStaking.sol#129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

INFO:Detectors:
Pragma version^0.8.0 (TrumpStaking.sol#7) allows old versions
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable TrumpStaking.DEPLOYER (TrumpStaking.sol#33) is not in mixedCase
Variable TrumpStaking._poolInfo (TrumpStaking.sol#35) is not in mixedCase
Variable TrumpStaking._depositUsersAmount (TrumpStaking.sol#36) is not in mixedCase
Variable TrumpStaking._userDeposited (TrumpStaking.sol#39) is not in mixedCase
Variable TrumpStaking._userStakedAmount (TrumpStaking.sol#40) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Loop condition pid < _poolInfo.length (TrumpStaking.sol#151) should use cached array length instead of referencing 'length' member of the storage array.
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Slither:TrumpStaking.sol analyzed (3 contracts with 93 detectors), 15 result(s) found

```

TESTNET VERSION

1- Add (**passed**):

<https://testnet.bscscan.com/tx/0xeaee637dc4a63f02c4d5496d0fa4b8844296664420e7e1308fa94aadd3ce6558>

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			

LOW RISK FINDING

Centralization – Missing Events

Severity: **Low**

Status: **Open**

Function: **add**

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function add(address token_, uint256 minAmount_, uint256 unitPoints_)
onlyDeployer() public {
    require(minAmount_ > 0, "minAmount_ must be greater than 0");
    require(!tokenExists[token_], "Token already added");

    PoolInfo memory pool = PoolInfo({
        token: IERC20(token_),
        minAmount: minAmount_,
        unitPoints: unitPoints_
    });

    _poolInfo.push(pool);
    tokenExists[token_] = true;
}
```

Suggestion:

Emit an event for critical changes.

LOW RISK FINDING

Centralization – Missing non-reentrant

Severity: **Low**

Status: **Open**

Function: **deposit/Withdraw/emergencyWithdraw**

Overview:

The nonReentrant modifier is added to ensure that the function can only be called once and cannot be re-entered by an attacker to drain the contract's funds.

```
function emergencyWithdraw(uint256 pid_) external payable {
    uint256 amount = _userStakedAmount[pid_][msg.sender];

    iWithdraw(pid_, amount, msg.sender);
    emit EmergencyWithdraw(msg.sender, pid_, amount);
}

function withdraw(uint256 pid_, uint256 amount_) external payable {
    iWithdraw(pid_, amount_, msg.sender);
    emit Withdraw(msg.sender, pid_, amount_);
}

function deposit(uint256 pid_, uint256 amount_) external payable {
    PoolInfo memory poolInfo = _poolInfo[pid_];

    if (WETH == address(poolInfo.token)) {
        if (amount_ > 0) {
            transferFrom(poolInfo.token, msg.sender, address(this), amount_);
        }

        if (msg.value > 0) {
            IWETH(address(WETH)).deposit{value: msg.value}();
            amount_ += msg.value;
        }
    } else if (amount_ > 0) {
        transferFrom(poolInfo.token, msg.sender, address(this), amount_);
    }
}
```

LOW RISK FINDING

```
require(amount_ >= poolInfo.minAmount);

_userStakedAmount[pid_][msg.sender] += amount_;

emit Deposit(msg.sender, pid_, amount_);

if (_userDeposited[msg.sender] == 0) {
    _userDeposited[msg.sender] = 1;

    unchecked {
        _depositUsersAmount++;
    }
}
```

Suggestion:

Using the ReentrancyGuard contract and the nonReentrant modifier, one can add an extra layer of security to one's Solidity smart contracts and protect them from reentrancy attacks.

LOW RISK FINDING

Optimization

Severity: **Low**

Subject: Missing requires error message.

Status: Open

Function: IWithdraw

Overview:

```
function iWithdraw(uint256 pid_, uint256 amount_, address user_) internal {  
    require(amount_ > 0);  
    require(amount_ <= _userStakedAmount[pid_][user_]);  
  
    PoolInfo storage poolInfo = _poolInfo[pid_];  
    _userStakedAmount[pid_][user_] -= amount_;  
  
    poolInfo.token.transfer(user_, amount_);  
}
```


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](https://twitter.com/expeleeofficial)

 [expelee](https://medium.com/expelee)

 [Expelee](https://t.me/Expelee)

 [expelee](https://in.linkedin.com/company/expelee)

 [expelee_official](https://www.instagram.com/expelee_official)

 [expelee-co](https://github.com/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.

The logo for 'expelee' is displayed in a large, stylized font. The letters 'expe' are white, and 'lee' is orange. The background of the page features faint, overlapping geometric shapes in shades of purple and blue.

Building the Futuristic **Blockchain Ecosystem**