auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

YourTrump

DATED : 27 Nov 24'

AUDIT SUMMARY

Project name – YourTrump

Date: 27 Nov, 2024

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	1	1	2
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- 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.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://sepolia.etherscan.io/address/0x1bc8b8432e72738f51f898e847b425d108c77fae#code

Token Information

Token Address: -

Name: YourTrump

Symbol: -

Decimals: -

Network: -

Token Type: ERC-20

Owner: -

Deployer: --

Token Supply: -

Checksum: Ee052c616934aeb47e6039f76b20d213

Testnet:
https://sepolia.etherscan.io/address/0x1bc8b8432e72738f51f898e847b425d108c77fae#code

TOKEN OVERVIEW

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

Other Privileges:

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.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

POINTS TO NOTE

- The owner can update user round data.
- The owner can set distribution wallets.
- The owner can set the token price.
- The owner can start and end round.
- The owner can withdraw ETH/TOKENS/USDT
- The owner can set Token/USDT/PriceFeed.

STATIC ANALYSIS

```
INFO:Detectors:
TokenPresale.calculateBonus(uint256) (TokenPresale.sol#308-322) performs a multiplication on the result of a division:
        - dynamicBonus = (finalRoundBonus * (tokenForSale() - totalTokensSold)) / tokenForSale() (TokenPresale.sol#316-317)
        - (tokens * dynamicBonus) / 100 (TokenPresale.sol#318)
TokenPresale.calculate(uint256,uint256) (TokenPresale.sol#324-342) performs a multiplication on the result of a division:
        - dynamicBonus = (finalRoundBonus * (tokenForSale() - totalTokensSold)) / tokenForSale() (TokenPresale.sol#336-337)
        - (tokens * dynamicBonus) / 100 (TokenPresale.sol#338)
TokenPresale.usdtToToken(uint256) (TokenPresale.sol#344-348) performs a multiplication on the result of a division:
        - totalTokens = (_amount * tokenPrice) / (100000000000000000) (TokenPresale.sol#345)
        - tokens = (totalTokens * (10 ** token.decimals())) / (100000000000000000) (TokenPresale.sol#346)
TokenPresale.ethToToken(uint256) (TokenPresale.sol#350-355) performs a multiplication on the result of a division:
        - ethToUsd = (_amount * getLatestPriceETH()) / (100000000000000000) (TokenPresale.sol#351)
        - numberOfTokens = (ethToUsd * tokenPrice) / 1000000000000000000 (TokenPresale.sol#352)
TokenPresale.ethToToken(uint256) (TokenPresale.sol#350-355) performs a multiplication on the result of a division:
        - numberOfTokens = (ethToUsd * tokenPrice) / 1000000000000000000 (TokenPresale.sol#352)
        - tokens = (numberOfTokens * (10 ** token.decimals())) / 1e8 (TokenPresale.sol#353)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

```
INFO:Detectors:
TokenPresale.setDistributionWallets(address[]) (TokenPresale.sol#399-412) should emit an event for:
        - numberOfWalletsToDistribute = wallets.length (TokenPresale.sol#411)
TokenPresale.setTokenPrice(uint256) (TokenPresale.sol#414-416) should emit an event for:
        - tokenPrice = _newPrice (TokenPresale.sol#415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

```
INFO:Detectors:
TokenPresale.claimTokens(TokenPresale.Rounds) (TokenPresale.sol#287-305) compares to a boolean constant:
        -require(bool,string)(roundStatus[_round] == true,Round not ended yet) (TokenPresale.sol#288)
TokenPresale.endRound(TokenPresale.Rounds) (TokenPresale.sol#419-422) compares to a boolean constant:
        -require(bool,string)(roundStatus[_round] == false,Round already ended) (TokenPresale.sol#420)
TokenPresale.startRound(TokenPresale.Rounds) (TokenPresale.sol#425-428) compares to a boolean constant:
        -require(bool,string)(roundStatus[_round] == true,Round is still active) (TokenPresale.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._contextSuffixLength() (TokenPresale.sol#24-26) is never used and should be removed
Context._msgData() (TokenPresale.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Version constraint ^0.8.20 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess.
It is used by:
        - ^0.8.20 (TokenPresale.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```


```
INFO:Detectors:
TokenPresale.claimTokens(TokenPresale.Rounds) (TokenPresale.sol#287-305) compares to a boolean constant:
        -require(bool,string)(roundStatus[_round] == true,Round not ended yet) (TokenPresale.sol#288)
TokenPresale.endRound(TokenPresale.Rounds) (TokenPresale.sol#419-422) compares to a boolean constant:
        -require(bool,string)(roundStatus[_round] == false,Round already ended) (TokenPresale.sol#420)
TokenPresale.startRound(TokenPresale.Rounds) (TokenPresale.sol#425-428) compares to a boolean constant:
        -require(bool,string)(roundStatus[_round] == true,Round is still active) (TokenPresale.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._contextSuffixLength() (TokenPresale.sol#24-26) is never used and should be removed
Context._msgData() (TokenPresale.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Version constraint ^0.8.20 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess.
It is used by:
        - ^0.8.20 (TokenPresale.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

STATIC ANALYSIS

```
INFO:Detectors:
Parameter TokenPresale.claimTokens(TokenPresale.Rounds)._round (TokenPresale.sol#287) is not in mixedCase
Parameter TokenPresale.usdtToToken(uint256)._amount (TokenPresale.sol#344) is not in mixedCase
Parameter TokenPresale.ethToToken(uint256)._amount (TokenPresale.sol#350) is not in mixedCase
Parameter TokenPresale.setTokenPrice(uint256)._newPrice (TokenPresale.sol#414) is not in mixedCase
Parameter TokenPresale.endRound(TokenPresale.Rounds)._round (TokenPresale.sol#419) is not in mixedCase
Parameter TokenPresale.startRound(TokenPresale.Rounds)._round (TokenPresale.sol#425) is not in mixedCase
Parameter TokenPresale.withdrawTokens(address,uint256)._tokenAddress (TokenPresale.sol#446) is not in mixedCase
Parameter TokenPresale.withdrawTokens(address,uint256)._amount (TokenPresale.sol#446) is not in mixedCase
Parameter TokenPresale.setToken(address)._token (TokenPresale.sol#459) is not in mixedCase
Parameter TokenPresale.setUSDT(address)._usdt (TokenPresale.sol#465) is not in mixedCase
Parameter TokenPresale.setPriceFeed(address)._priceFeed (TokenPresale.sol#471) is not in mixedCase
Variable TokenPresale.USDT (TokenPresale.sol#161) is not in mixedCase
Variable TokenPresale.ETHRaised (TokenPresale.sol#178) is not in mixedCase
Variable TokenPresale.USDTRaised (TokenPresale.sol#179) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
TokenPresale.finalRoundBonus (TokenPresale.sol#187) should be constant
TokenPresale.round1Bonus (TokenPresale.sol#184) should be constant
TokenPresale.round2Bonus (TokenPresale.sol#185) should be constant
TokenPresale.round3Bonus (TokenPresale.sol#186) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:TokenPresale.sol analyzed (5 contracts with 94 detectors), 47 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Set Distribution Wallets (passed):

https://sepolia.etherscan.io/tx/0x20cf6a87b747b3c8f5c9f8b7898db991f154d078f0d49e064262367ba7286db8

CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
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
Critical	0
High-Risk	0
Medium-Risk	1
Low-Risk	1
Gas Optimization / Suggestions	2

MANUAL TESTING

Centralization – Missing Require Check.

Severity: Medium

Function: setDistributionWallets

Status: Open

Overview:
The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function setDistributionWallets(address[] memory wallets)     //@audit
//Missing require check
    external
    onlyOwner
{
    require(wallets.length > 0, "Must provide at least one wallet");

    delete distributionWallets;

    for (uint256 i = 0; i < wallets.length; i++) {
distributionWallets.push(wallets[i]);
    }

    numberOfWalletsToDistribute = wallets.length;
}
```

Suggestion: It is recommended that the address should not be able to set as a contract address.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Subject: Missing Events

Status: Open

Overview:

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

```
function setDistributionWallets(address[] memory wallets)     //@audit
//Missing require check
     external
     onlyOwner
{
     require(wallets.length > 0, "Must provide at least one wallet");

     delete distributionWallets;

     for (uint256 i = 0; i < wallets.length; i++) {
distributionWallets.push(wallets[i]);
     }

     numberOfWalletsToDistribute = wallets.length;
}
 function setTokenPrice(uint256 _newPrice) external onlyOwner {
     tokenPrice = _newPrice;
}

// End a presale round
function endRound(Rounds _round) external onlyOwner {
     require(roundStatus[_round] == false, "Round already ended");
     roundStatus[_round] = true;
}
```

MANUAL TESTING

```
// Start a presale round
function startRound(Rounds _round) external onlyOwner {
      require(roundStatus[_round] == true, "Round is still active");
      roundStatus[_round] = false;
}

 function setToken(address _token) external onlyOwner {
      require(_token != address(0), "Invalid address");
      token = IToken(_token);
}

// Function to set the USDT address
function setUSDT(address _usdt) external onlyOwner {
      require(_usdt != address(0), "Invalid address");
      USDT = IToken(_usdt);
}

// Function to set the price feed address
function setPriceFeed(address _priceFeed) external onlyOwner {
      require(_priceFeed != address(0), "Invalid address");
      priceFeedeth = AggregatorV3Interface(_priceFeed);
}
}

 function withdrawETH() external onlyOwner {
payable(owner()).transfer(address(this).balance);
}

// Withdraw USDT by owner
function withdrawUSDT() external onlyOwner {
      uint256 contractBalance = USDT.balanceOf(address(this));
      require(contractBalance > 0, "No USDT balance to withdraw");
      require(
          USDT.transfer(owner(), contractBalance),
          "USDT withdrawal failed"
      );
}
```

MANUAL TESTING

```
// Emergency token recovery function
function withdrawTokens(address _tokenAddress, uint256 _amount)
    external
    onlyOwner
{
    uint256 contractBalance =
IToken(_tokenAddress).balanceOf(address(this));
    require(contractBalance >= _amount, "Insufficient token balance");
    require(
IToken(_tokenAddress).transfer(owner(), _amount),
        "Token transfer failed"
    );
}
```

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma.

Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

pragma solidity ^0.8.20;

Suggestion:
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:
Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.

```
function _msgData() internal view virtual returns (bytes calldata) {
return msg.data;
  }

function _contextSuffixLength() internal view virtual returns (uint256) {
return 0;
  }
```

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.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace