Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

December, 2021

Audit Details

Audited project

Pension Plan

Deployer address

0x53c538Ae77E9DFeD2337C94a26451C9a6eE4B435

Client contacts:

Pension Plan team

Blockchain

Ethereum

Project website:

https://pensionplan.finance/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (TechRate) owe no duty of care towards you or any other person, nor does TechRate make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and TechRate hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, TechRate hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against TechRate, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

TechRate was commissioned by Pension Plan to perform an audit of smart contracts:
https://etherscan.io/address/0xf14b9adf84812ba463799357f4dc716b4384010b#code

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts Details

Token contract details for 11.12.2021

Contract name	Pension Plan
Contract address	0xf14b9ADF84812bA463799357f4dc716b4384010B
Total supply	1,000,000,000,000
Token ticker	PP
Decimals	8
Token holders	228
Transactions count	771
Top 100 holders dominance	99.67%
Total fee	12
Uniswap V2 pair	0x5E62De3FAB648c2B83AF866EDFFb23799BF22Cd0
Contract deployer address	0x53c538Ae77E9DFeD2337C94a26451C9a6eE4B435
Contract's current owner address	0x53c538Ae77E9DFeD2337C94a26451C9a6eE4B435

Pension Plan Token Distribution

The top 100 holders collectively own 99.67% (996,667,575,916.40 Tokens) of Pension Plan

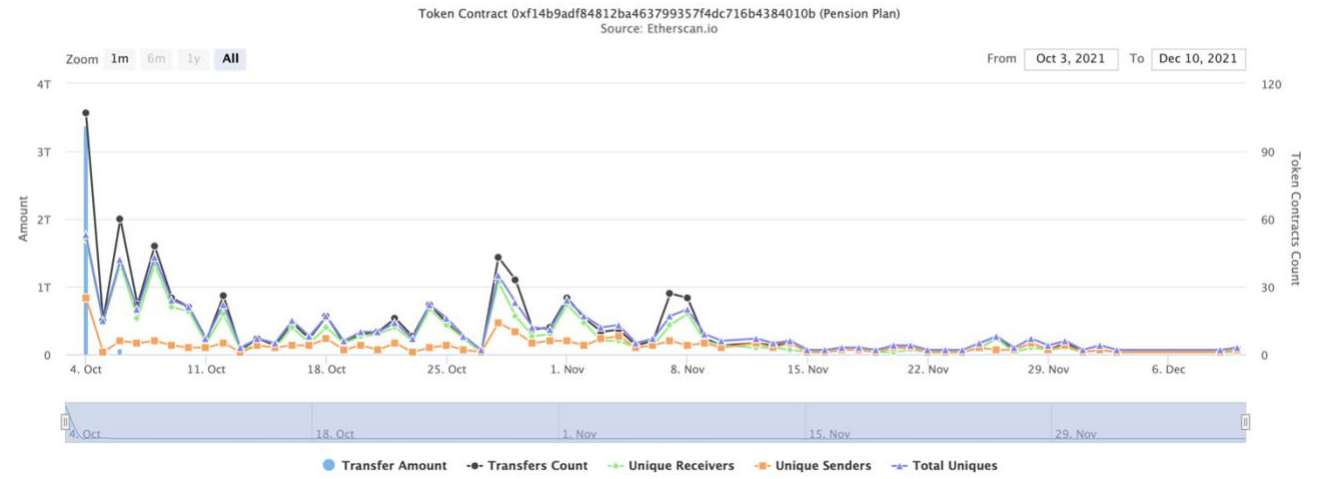
Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 228

(A total of 996,667,575,916.40 tokens held by the top 100 accounts from the total supply of 1,000,000,000,000.00 token)

Pension Plan Interaction details

Time Series: Token Contract Overview

Mon 4, Oct 2021 - Fri 10, Dec 2021



Pension Plan Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	Null Address: 0x000...dEaD	513,367,618,338.26097826	51.3368%
2	Pension Plan: Deployer	50,000,000,000	5.0000%
3	0xc46350900b078af7ab8042bd6f81198a0efd1f58	37,072,502,458.50282216	3.7073%
4	0xe424f3a919fe55d28dede398eb0674922a9f8154	33,609,864,270.8416214	3.3610%
5	0x9d4321ff203bf403f832697643f256818b686df6	31,947,627,341.0370226	3.1948%
6	0x0e9ea21ce2647d29089535c903fb5c41a7a92c58	29,575,916,360.00151627	2.9576%
7	0x31b0f4e38a12406274e9da31cc3a71bf512c1e70	27,770,000,000.19949017	2.7770%
8	0x14732a6c04f012c538f224e7eb50fb8233951916	20,650,638,136.89187278	2.0651%
9	0xbe91107fc7caabe8263796576c3549e3187aca4f	20,339,764,904.32791582	2.0340%
10	0xb81014724c1dc1beece29a9e66244115326a3598	18,993,349,027.65061645	1.8993%

Contract functions details

+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ [Int] IERC20Metadata (IERC20)
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ [Lib] Math
  - [Int] max
  - [Int] min
  - [Int] average
  - [Int] ceilDiv

+ [Lib] Arrays
  - [Int] findUpperBound

+ [Lib] Counters
  - [Int] current
  - [Int] increment #
  - [Int] decrement #
  - [Int] reset #

+ Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Prv] _setOwner #

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Int] functionStaticCall
  - [Int] functionStaticCall
  - [Int] functionDelegateCall #

  - [Int] functionDelegateCall #
  - [Int] verifyCallResult

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #
  - [Ext] removeLiquidityETHWithPermit #
  - [Ext] swapExactTokensForTokens #
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens ($)
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #

- [Ext] swapETHForExactTokens ($)
- [Ext] quote
- [Ext] getAmountOut
- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ PensionPlan (Context, IERC20, IERC20Metadata, Ownable)
- [Pub] <Constructor> #
- [Pub] name
- [Pub] symbol
- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Int] _approve #
- [Pub] eligibleSupply
- [Pub] eligibleSupplyAt
- [Pub] isExcludedFromReward
- [Pub] excludeFromReward #
  - modifiers: onlyOwner
- [Pub] includeInReward #
  - modifiers: onlyOwner
- [Pub] isBanned
- [Ext] ban #
  - modifiers: onlyOwner
- [Ext] unban #
  - modifiers: onlyOwner
- [Prv] _processPayouts #
- [Prv] _handleSwapAndPayout #
- [Int] _transfer #
- [Int] _beforeTokenTransfer #
- [Prv] swapTokensForEth #
  - modifiers: lockTheSwap
- [Prv] swapETHForTokensAndBurn #
  - modifiers: lockTheSwap
- [Ext] setMinimumTokensBeforeSwap #
  - modifiers: onlyOwner
- [Ext] setMarketingAddress #
  - modifiers: onlyOwner
- [Ext] setDevelopmentAddress #
  - modifiers: onlyOwner
- [Ext] setFoundationAddress #

  - modifiers: onlyOwner
- [Ext] setHachikoInuBuybackAddress #
  - modifiers: onlyOwner
- [Ext] setMinimumETHBeforePayout #
  - modifiers: onlyOwner
- [Ext] setPayoutsToProcess #
  - modifiers: onlyOwner
- [Ext] manuallyProcessPayouts #
  - modifiers: onlyOwner
- [Int] _snapshot #
- [Int] _getCurrentSnapshotId
- [Pub] balanceOfAt
- [Pub] totalSupplyAt
- [Prv] _valueAt
- [Prv] _updateAccountSnapshot #
- [Prv] _updateTotalSupplySnapshot #
- [Prv] _updateSnapshot #
- [Prv] _lastSnapshotId
- [Ext] <Fallback> ($)

($) = payable function
= non-constant function

Issues Checking Status

	Issue description	Checking status
1.	Compiler errors.	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.	Low issues
10.	Methods execution permissions.	Passed
11.	Economy model of the contract.	Passed
12.	The impact of the exchange rate on the logic.	Passed
13.	Private user data leaks.	Passed
14.	Malicious Event log.	Passed
15.	Scoping and Declarations.	Passed
16.	Uninitialized storage pointers.	Passed
17.	Arithmetic accuracy.	Passed
18.	Design Logic.	Passed
19.	Cross-function race conditions.	Passed
20.	Safe Open Zeppelin contracts implementation and usage.	Passed
21.	Fallback function security.	Passed

Security Issues

High Severity Issues

No high severity issues found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

1. Out of gas

Issue:

- The function `includeInReward()` uses the loop to find and remove addresses from the `_excluded` list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function includeInReward(address account) public onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The functions `eligibleSupply()` and `eligibleSupplyAt()` also uses the loop for evaluating total supply. It also could be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function eligibleSupply() public view returns (uint256) {
    uint256 supply =  _totalSupply;
    for (uint256 i = 0; i < _excluded.length; i++) {
        unchecked {
            supply = supply - _balances[_excluded[i]];
        }
    }
    return supply;
}

function eligibleSupplyAt(uint256 snapshotId) public view returns (uint256) {
    uint256 supply =  _totalSupply;
    for (uint256 i = 0; i < _excluded.length; i++) {
        unchecked {
            supply = supply - balanceOfAt(_excluded[i], snapshotId);
        }
    }
    return supply;
}
```

- The functions `unban()` uses the loop to find and remove addresses from `_banned` list. It also could be aborted with `OUT_OF_GAS` exception if there will be a long banned addresses list.

```
function unban(address account) external onlyOwner() {
    require(_isBanned[account], "Account is already unbanned");
    for (uint256 i = 0; i < _banned.length; i++) {
        if (_banned[i] == account) {
            _banned[i] = _banned[_banned.length - 1];
            _isBanned[account] = false;
            _banned.pop();
            break;
        }
    }
    if (_isExcluded[account]) {
        includeInReward(account);
    }
}
```

Recommendation:
Check that the addresses array length is not too big.

Notes

- No Transfer event emitted on taking a fee. And the amount written in the event is not correct.

Owner privileges (In the period when the owner is not renounced)

- Owner can include in and exclude from reward.

```
function excludeFromReward(address account) public onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) public onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- Owner can include in and exclude from banned list (no transfer for these addresses).

```
function ban(address account) external onlyOwner() {
    require(!_isBanned[account], "Account is already banned");
    _isBanned[account] = true;
    _banned.push(account);
    if (!_isExcluded[account]) {
        excludeFromReward(account);
    }
}

function unban(address account) external onlyOwner() {
    require(_isBanned[account], "Account is already unbanned");
    for (uint256 i = 0; i < _banned.length; i++) {
        if (_banned[i] == account) {
            _banned[i] = _banned[_banned.length - 1];
            _isBanned[account] = false;
            _banned.pop();
            break;
        }
    }
    if (_isExcluded[account]) {
        includeInReward(account);
    }
}
```

- Owner can change minimum tokens before swap value.

```
function setMinimumTokensBeforeSwap(uint256 _minimumTokensBeforeSwap) external onlyOwner() {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap;
}
```

- Owner can change minimum ETH before payout value.

```
function setMinimumETHBeforePayout(uint256 _minimumETHBeforePayout) external onlyOwner() {
    minimumETHBeforePayout = _minimumETHBeforePayout;
}
```

- **Owner can change marketing, development, foundation and HachikoInu buyback addresses.**

```
function setMarketingAddress(address _marketingAddress) external onlyOwner() {
    marketingAddress = payable(_marketingAddress);
}

function setDevelopmentAddress(address _developmentAddress) external onlyOwner() {
    developmentAddress = payable(_developmentAddress);
}

function setFoundationAddress(address _foundationAddress) external onlyOwner() {
    foundationAddress = payable(_foundationAddress);
}

function setHachikoInuBuybackAddress(address _hachikoInuBuybackAddress) external onlyOwner() {
    hachikoInuBuybackAddress = payable(_hachikoInuBuybackAddress);
}
```

- **Owner can change payoutToProcess value.**

```
function setPayoutsToProcess(uint256 _payoutsToProcess) external onlyOwner() {
    payoutsToProcess = _payoutsToProcess;
}
```

- **Owner can manually process payouts.**

```
function manuallyProcessPayouts() external onlyOwner() returns(bool, uint256) {
    if (processingPayouts) {
        _processPayouts();
    }
    else {
        uint256 balance = address(this).balance;
        marketingAddress.transfer(balance / 6);
        developmentAddress.transfer(balance / 12);
        foundationAddress.transfer(balance / 12);
        hachikoInuBuybackAddress.transfer( balance / 24);
        swapETHForTokensAndBurn(balance / 24);
        processingPayouts = true;
        _payoutAmount = address(this).balance;
        _snapshotId = _snapshot();
        emit PayoutStarted(_payoutAmount);
    }
    return (processingPayouts, _lastProcessedAddressIndex);
}
```

Conclusion

Smart contracts contain low severity issues and owner privileges! Audited only token of the project. Liquidity pair contract's security is not checked due to out of scope.

Liquidity locking details provided by the team:
https://app.unicrypt.network/amm/uni-v2/pair/0x5E62De3FAB648c2B83AF866EDFFb23799BF22Cd0

TechRate note:

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.