Cyberscope

Audit Report
Puli Payroll

April 2022

Type	ERC20
Network	RINKEBY
Address	0x4feB43230508384350e2b50914355E29Bc8c1098
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	Payroll
Compiler Version	v0.8.1+commit.df193b15
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x4feB43230508384350e2b50914355E29Bc8c1098

Source Files

Filename	**SHA256**
contract.sol	2bc5bd3e58c24cd938888b21d38badf82217ee203487435048c3bcf66e99937b

Audit Updates

Initial Audit	11th April 2022
Corrected	

Contract Analysis

The contract implements a vesting-related feature where users are holding a node and get rewards proportionally to the node ratio. In Payroll terminology, the nodes are called departments, the vesting accounts are the head of the department, and the node ratio are the shares.

- A new department can be added only if all the current heads sign.
- One department head can only be owned by one address. So a single address can not hold more than one department.
- Anyone can distribute funds to the contract. Each department will be fueled with funds proportionally to the shares.
- The funds can either be the native currency or any other token.
- The head of each department can withdraw the proportional funds of the owned department any time.
- A department head can move the ownership to another address, only if the uniqueness rule is fulfilled.
- Each department role has the authority to increase the shares of the owned department only if it is signed by all the department heads.
- A department head can remove a department only if it is signed by all the department heads except one.
- The remove process will delete the department and add the share equally to the rest departments.

Total Shares Concurrency

Criticality	critical
Location	contract.sol#L716,560

Description

The contract is using the total shares as a divisor to distribute the shares for each department. Thus the totalShare property should always mirror the sum of the share of all the departments. There are two segments where this synchronisation is not happening:

1. The `updateShares()` method increases the shares of a specific department, but the totalShares is not updated.

2. The `removeDepartment()` method moves the shares of the department that is going to be removed to the rest departments. The calculation of the shared distribution is a result of division. That means that if the result is decimal in some deviations then the shared amount will not be precise. Hence, the sum of the share of all the departments will be different in comparison with the totalShares.

```
for (uint8 i = 0; i < departmentNames.length; i++) {
    Department memory department = departments[departmentNames[i]];
    uint256 addedAmount = (_amount * department.shares) / totalShares;
    balances[departmentNames[i]][_token] += addedAmount;
}
```

Recommendation

The contract should correctly update the totalShares variable when the departments.share property is updated.

Minimum Fund Distribution

Criticality	minor
Location	contract.sol#L752

Description

The contract deposit of native currency or pure token distributes the amount proportionally to the shares of each department. The calculation of the shared amount is a result of division. That means that if the provided amount is less than a threshold, there will be departments that will receive zero amount.

```
function _distributeTokens(address _token, uint256 _amount) internal {
    for (uint8 i = 0; i < departmentNames.length; i++) {
        Department memory department = departments[departmentNames[i]];
        uint256 addedAmount = (_amount * department.shares) / totalShares;
        balances[departmentNames[i]][_token] += addedAmount;
    }
}
```

Recommendation

The contract could embed a check for now allowing the transaction to proceed if there is at least one department that receives zero funds.

Redundant Checks

Criticality	minor
Location	contract.sol#L696

Description

The contract contains some tautology checks. That means that if one check is fulifies then the other check will be fullifies by definition.

```
require(
    departments[_departmentName].head != address(0),
    "Department does not exists"
);
require(
    departments[_departmentName].head == _msgSender(),
    "Sender is not head of the department"
);
```

Recommendation

The contract could eliminate the tautology checks.

Contract Diagnostics

🔴 Critical 🟠 Medium ● Minor

Severity	Code	Description
●	L01	Public Function could be Declared External
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality
●	L12	Using Variables before Declaration

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L525,529,560,630,634,690,716

Description

Public functions that are never called by the contract should be declared external to save gas.

```
updateShares
transferOwnership
releasedOf
removeDepartment
addDepartment
getAllDepartmentNames
```

Recommendation

Use the external attribute for functions never called from the contract

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L530,531,532,533,535,561,562,563,565,614,620,630,634,690,717,718,719,720,721,477

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
maxDeparments
_signatures
_data
_reason
_changeInShares
_departmentName
_tokenAddresses
_shares
_head
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L344,395,428,441,184,200,159

Description

Functions that are not used in the contract, and make the code's size bigger.

```
toString
toHexString
toTypedDataHash
toEthSignedMessageHash
recover
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contract.sol#L760

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(bool,string)(signatureRegister[signatures[i]] == false,Signature already
registered)
```

Recommendation

Remove the equality to the boolean constant.

L12 - Using Variables before Declaration

Criticality	minor
Location	contract.sol#L276

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or the variable has been declared in a different scope.

```
r
```

Recommendation

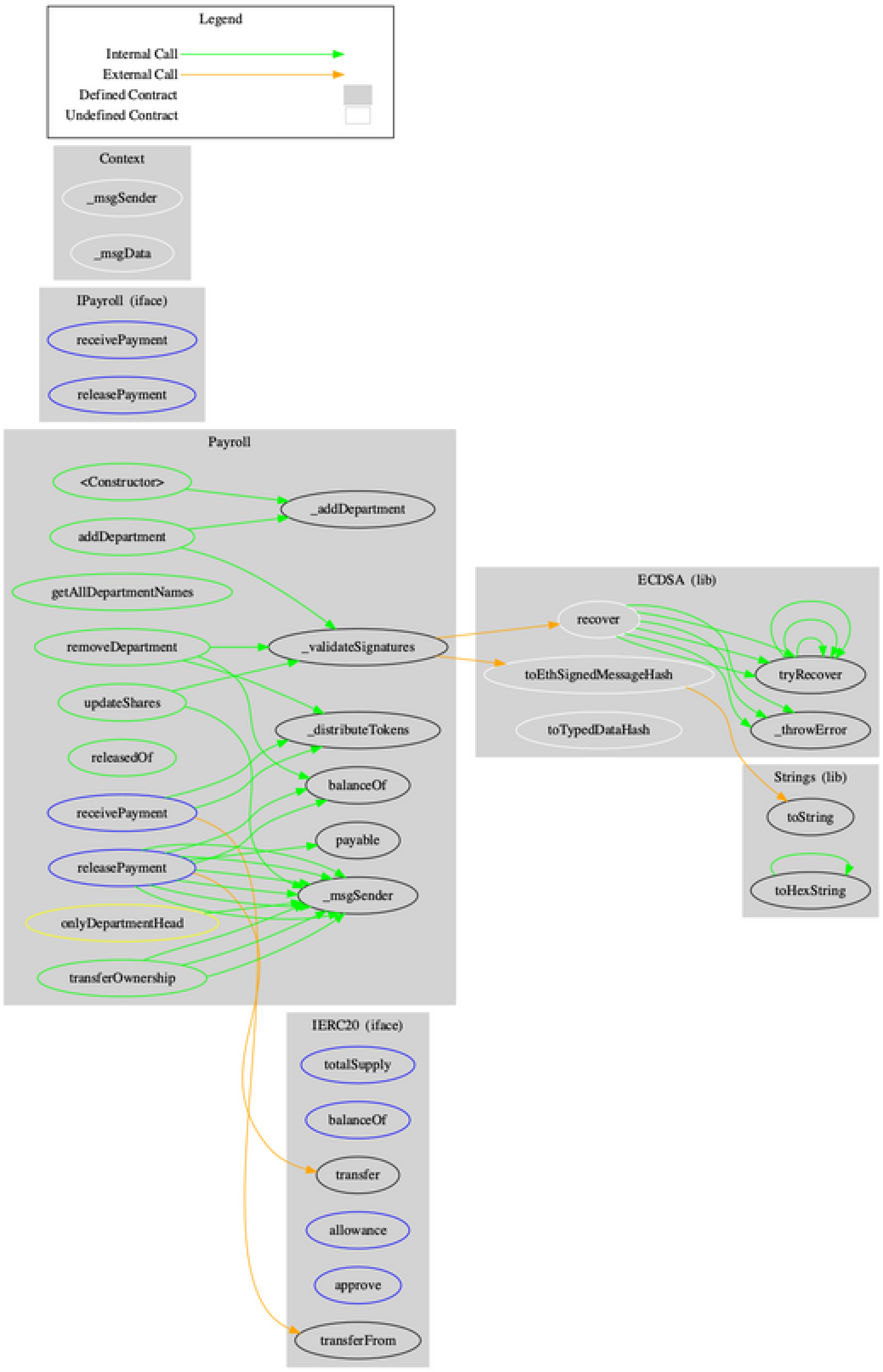
The variables should be declared before any usage of them.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IPayroll	Interface			
	receivePayment	External	Payable	-
	receivePayment	External	✓	-
	releasePayment	External	Payable	-
	releasePayment	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
ECDSA	Library			
	_throwError	Private		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		

	tryRecover	Internal		
	recover	Internal		
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toTypedDataHash	Internal		
Payroll	Implementation	IPayroll, Context		
	<Constructor>	Public	✓	-
	getAllDepartmentNames	Public		-
	addDepartment	Public	✓	onlyDepartmentHead
	removeDepartment	Public	✓	onlyDepartmentHead
	balanceOf	Public		-
	balanceOf	Public		-
	releasedOf	Public		-
	releasedOf	Public		-
	receivePayment	External	Payable	-
	receivePayment	External	✓	-
	releasePayment	External	Payable	onlyDepartmentHead
	releasePayment	External	✓	onlyDepartmentHead
	transferOwnership	Public	✓	onlyDepartmentHead
	updateShares	Public	✓	onlyDepartmentHead
	_distributeTokens	Internal	✓	
	_validateSignatures	Internal	✓	
	_addDepartment	Internal	✓	

Contract Flow



Summary

Puli Payroll is a staking variation contract. The users own nodes and get rewards according to the node’s power. In this audit we investigate the business logic, we mention some potential vulnerabilities and we suggest potential improvements.

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.

Cyberscope 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 Cyberscope 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 Cyberscope 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 Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io