Cyberscope

Audit Report

PayMe Vesting

November 2022

Github	https://github.com/payMeQuiz/payMe-Project
Commit	3314623dd1f47d2ee69aa33b32972d081845c272
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	payMETokenVesting
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	0 runs
Github	https://github.com/payMeQuiz/payMe-Project
Commit	3314623dd1f47d2ee69aa33b32972d081845c272
Explorer	https://testnet.bscscan.com/token/0x88779C9f4F7972b0926861E904B71C6D8227AC30
Domain	https://payme.games

Audit Updates

Initial Audit	17th October 2022 https://github.com/cyberscope-io/audits/blob/main/payme/v1/paymeTokenVesting.pdf
Corrected	9th November 2022

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	da66c17044345dc892d85bd7ddc9745d25df0b3dacfba8f84eb87c60d6e40fe3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	cd823c76cbf5f5b6ef1bda565d58be66c843c37707cd93eb8fb5425deebd6756
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	b6adbe9bc075b15cfb4b90f1ae020da4c78e3feada056a4c75b875350285c915
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol	b97515a88e75c313eacf0a27c9439ef371d86d4c2730d3b13076640942f813df
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	4e09a7479aa3e7c313f8fc141c4c8fc04e0abfeb8754615ef7d78ec94c298b07
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	b7410d275fc7d26e36b0851541d6ff290593ba72d64b5c906978124b123915c1

@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	35fb271561f3dc72e91b3a42c6e40c2bb2e788cd8ca58014ac43f6198b8d32ca
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	43127075ebfd67044ac7cbee0734c30911e435f58a42d8cf20a86d9fe963ae80
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	4039686a509394aed475619c4e0b3a2df1df34fe59e90b9add8669de371eb731
contracts/ico/payMETokenVesting.sol	d8fd864e3c39f49ce36ca539c33169535e045fbfbd09e0dc0999af014e2fde77

Introductions

The PaymeTokenVesting contract implements a vesting contract as an upgradable proxy. The contract is responsible for creating and configuring vesting schedules for a beneficiary.
Each beneficiary can have multiple vesting schedules. In addition, the contract monitors the vesting schedules by keeping track of the beneficiaries and how many times its beneficiary has vested.

Roles

The contract has an owner role and a beneficiary role. The beneficiary is any user that vests on the contract. The owner has the authority to withdraw a specific amount from the contract if possible. Additionally, the owner and any user that is beneficiary have the authority:

1. Revoke all the vested amount if the vesting period is elapsed or the proportional amount in relation to the vested period.
2. Release tokens for TGE If the TGE opening time has elapsed.
3. Release a specific amount of vested tokens if it is possible.

Contract Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Medium	VTAI	Vesting Total Amount Inconsistency	Unresolved
Minor / Informative	VTI	Vesting Token Issues	Unresolved
Minor / Informative	MC	Missing Check	Unresolved
Minor / Informative	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor / Informative	L05	Unused State Variable	Unresolved
Minor / Informative	L11	Unnecessary Boolean equality	Unresolved

VTAI - Vesting Total Amount Inconsistency

Criticality	medium
Location	contract.sol#L273
Status	Unresolved

Description

The contract is using two variables to monitor vesting total amounts.

- The variable `vestingSchedulesTotalAmount` to aggregate the total vested amount of all vesting schedules. The variable is aggregating the total vested amount of the contract, but it is not taking into account the TGE amount.

```
vestingSchedulesTotalAmount = vestingSchedulesTotalAmount.add(iAmount);
```

- The variable `vestingSchedule.amountTotal` to aggregate the total vested amount of each vesting schedule, but it is not taking into account the TGE amount.

```
vestingSchedules[vestingScheduleId] = VestingSchedule(
        true,
        iBeneficiary,
        cliff,
        iStart,
        iDuration,
        iSlicePeriodSeconds,
        iRevocable,
        iAmount,
```

When the TGE amount is claimed by the investors, it is creating an inconsistency between the actual total amount and the calculated vested ("vestingSchedulesTotalAmount") amount.

```
function releaseTokenForTGE(bytes32 vestingScheduleId)
    public
    nonReentrant
    {
        //..
        uint256 currentTime = getCurrentTime();
```

```
        require(currentTime >= TGEOpeningTime, "TGE: time not reached!");
        require(TGETokenParticipates[vestingScheduleId] == 0, "TGE: Token
Already claimed");
        uint256 TGEReleaseAmount =
vestingSchedule.amountTotal.mul(TGEPercent).div(100);
        vestingSchedule.released =
vestingSchedule.released.add(TGEReleaseAmount);
        vestingSchedulesTotalAmount =
vestingSchedulesTotalAmount.sub(TGEReleaseAmount);
         //..
    }
```

Recommendation

The contract could sum up the extra TGE amount and the iAmount in the total vesting amount variables. To be more specific the total vesting amount is the aggregation of the invested amount and the TGE percent in relation to the invested amount.

VTI - Vesting Token Issues

Criticality	minor / informative
Location	contract.sol#L273,425
Status	Unresolved

Description

The vesting functionality may produce some vulnerabilities during the claiming period.

TGE vs Normal Claim

A user is able to claim the whole vested amount and then claim the TGE amount as well. The opposite cannot happen. If a user claims the TGE, then he will not be able to claim the vested amount.

Early TGE claim

If a user claims the TGE amount at the begging of the vesting period, then the '_computeReleasableAmount' method will revert. This will happen because the vestingSchedule.released will be greater than the releasable amount.

```
function releaseTokenForTGE(bytes32 vestingScheduleId) public nonReentrant
    {
        //..
       vestingSchedule.released = vestingSchedule.released.add(TGEReleaseAmount);
        //..
    }

function _computeReleasableAmount(VestingSchedule memory vestingSchedule) internal view
returns(uint256){
          if ((currentTime < vestingSchedule.cliff) || vestingSchedule.revoked) {
            return 0;
        } else if (currentTime >= vestingSchedule.start.add(vestingSchedule.duration)) {
            //time has elapsed -> release all

            return vestingSchedule.amountTotal.sub(vestingSchedule.released);
        } else {
```

```
        //compute daily vesting amount
        //vested amount = amount * ( current time - start time )/ duration
        uint256 timeFromStart = currentTime.sub(vestingSchedule.start);
        uint256 vestedAmount =
vestingSchedule.amountTotal.mul(timeFromStart).div(vestingSchedule.duration);
        if(currentTime >= tgeOpeningTime){
          uint256 tgeAmount = vestingSchedule.amountTotal.mul(tgePercent).div(100);
          vestedAmount.add(tgeAmount);
        }
        vestedAmount = vestedAmount.sub(vestingSchedule.released);
        return vestedAmount;
      }
    }
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

The contract should aggregate all the releasable amounts in the variable vestingSchedule.amountTotal.

MC - Missing Check

Criticality	minor / informative
Location	contract.sol#L119
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues. To be more specific, the variable `TGEPercent` is not properly sanitized.

```
function initialize(IERC20Upgradeable iToken,uint256 iTGEPercent,uint256 iTGEOpeningTime)
public initializer {
        require(address(iToken) != address(0));
        require(iTGEPercent > 0, "TGE Amount must be greater than 0");
        require(iTGEOpeningTime > 0, "TGE Openning time must be greater than 0");

        __Ownable_init_unchained();
        __ReentrancyGuard_init_unchained();

        _token = iToken;

        tgeOpeningTime = iTGEOpeningTime;
        tgePercent = iTGEPercent;
    }
```

Recommendation

The contract should properly check the variables according to the required specifications.

- TGEPercent should be greater than zero and lower than 100 percent.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/ico/payMETokenVesting.sol#L55,17
Status	Unresolved

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.

```
TGETokenParticipates
payMETokenVesting
```

Recommendation

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

L05 - Unused State Variable

Criticality	minor / informative
Location	@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#L74
Status	Unresolved

Description

There are segments that contain unused state variables.

```
__gap
```

Recommendation

Remove unused state variables.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contracts/ico/payMETokenVesting.sol#L273
Status	Unresolved

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)(vestingSchedule.releaseAtTGE == true,ReleaseTokenAtTGE:
only investors can claim token at TGE)
```

Recommendation

Remove the equality to the boolean constant.

Contract Functions

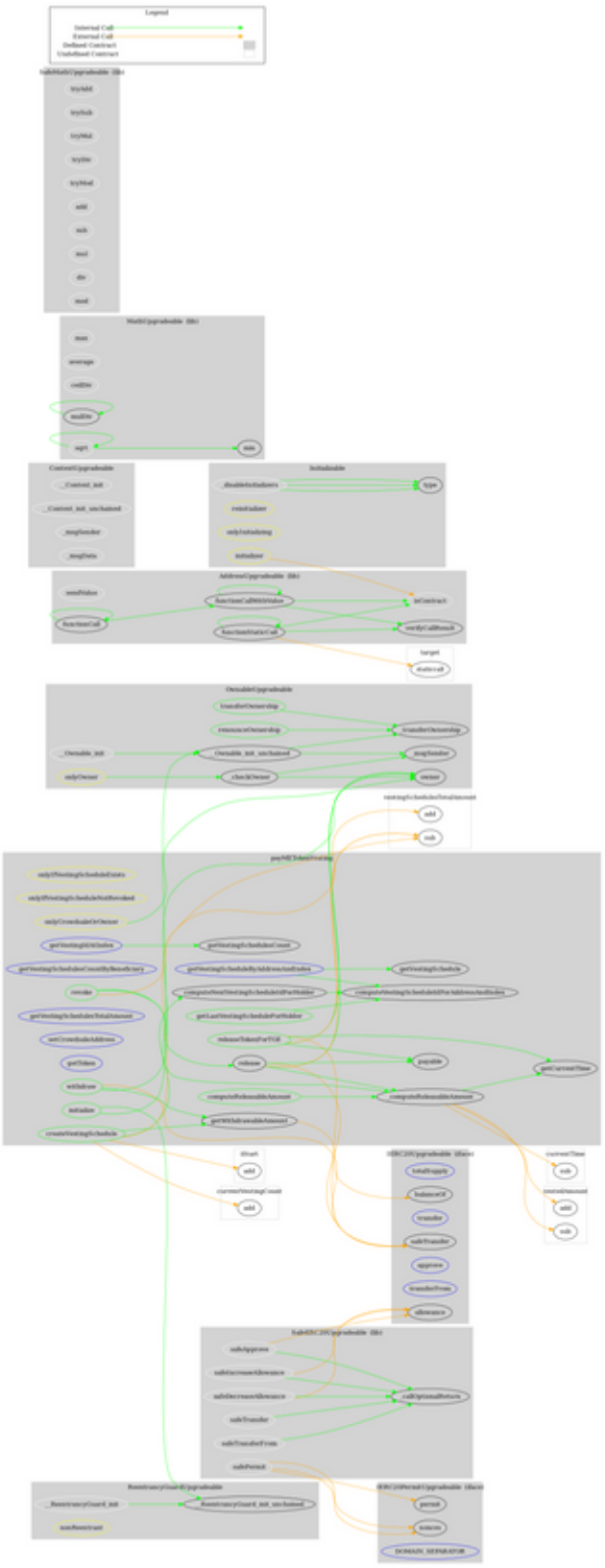
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
OwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Ownable_init	Internal	✓	onlyInitializing
	__Ownable_init_unchained	Internal	✓	onlyInitializing
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Initializable	Implementation			
	_disableInitializers	Internal	✓	
ReentrancyGuardUpgradeable	Implementation	Initializable		
	__ReentrancyGuard_init	Internal	✓	onlyInitializing
	__ReentrancyGuard_init_unchained	Internal	✓	onlyInitializing
IERC20PermitUpgradeable	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20Upgradeable	Interface			
	totalSupply	External		-
	balanceOf	External		-

	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeERC20Upgradeable	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	safePermit	Internal	✓	
	_callOptionalReturn	Private	✓	
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResult	Internal		
ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal	✓	onlyInitializing
	__Context_init_unchained	Internal	✓	onlyInitializing
	_msgSender	Internal		
	_msgData	Internal		
MathUpgradeable	Library			

	max	Internal		
	min	Internal		
	average	Internal		
	ceilDiv	Internal		
	mulDiv	Internal		
	mulDiv	Internal		
	sqrt	Internal		
	sqrt	Internal		
SafeMathUpgradeable	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
payMETokenVesting	Implementation	OwnableUpgradeable, ReentrancyGuardUpgradeable		
	initialize	Public	✓	initializer
	getVestingSchedulesCountByBeneficiary	External		-
	getVestingIdAtIndex	External		-
	getVestingScheduleByAddressAndIndex	External		-
	getVestingSchedulesTotalAmount	External		-
	setCrowdsaleAddress	External	✓	-

	getToken	External		-
	createVestingSchedule	Public	✓	onlyCrowdsaleOrOwner
	revoke	Public	✓	onlyOwner onlyIfVestingScheduleNotRevoked
	withdraw	Public	✓	nonReentrant onlyOwner
	releaseTokenForTGE	Public	✓	nonReentrant
	release	Public	✓	nonReentrant onlyIfVestingScheduleNotRevoked
	getVestingSchedulesCount	Public		-
	computeReleasableAmount	Public		onlyIfVestingScheduleNotRevoked
	getVestingSchedule	Public		-
	getWithdrawableAmount	Public		-
	computeNextVestingScheduleIdForHolder	Public		-
	getLastVestingScheduleForHolder	Public		-
	computeVestingScheduleIdForAddressAndIndex	Public		-
	_computeReleasableAmount	Internal		
	getCurrentTime	Internal		

Contract Flow



Domain Info

Domain Name	payme.games
Registry Domain ID	29f4ee9286e043058b41ccc27375747f-DONUTS
Creation Date	2021-01-06T13:00:37Z
Updated Date	2022-08-05T11:31:27Z
Registry Expiry Date	2023-01-06T13:00:37Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

The domain was created almost 2 years before the creation of the audit. It will expire in 2 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

The PaymeTokenVesting contract is responsible for generating vesting schedules. This audit investigates security issues and mentions business logic concerns and 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 provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io