Cyberscope

Audit Report

PayMe Crowdsale

November 2022

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

Table of Contents

Contract Review

Contract Name	payMETokenCrowdsale
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/0xa246B4B25BD840ac5C71378bFe2E344BEcCD4810
Domain	https://payme.games

Audit Updates

Initial Audit	17th October 2022 https://github.com/cyberscope-io/audits/blob/main/payme/v1/paymeTokenCrowdsale.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
@openzeppelin/contracts/access/AccessControl.sol	5af1771388b4fe634e0a566716e32c6d00a5372875099127b274d4cf8a94e9d2
@openzeppelin/contracts/access/IAccessControl.sol	d03c1257f2094da6c86efa7aa09c1c07ebd33dd31046480c5097bc2542140e45
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/security/Pausable.sol	2072248d2f79e661c149fd6a6593a8a3f038466557c9b75e50e0b001bcb5cf97
@openzeppelin/contracts/security/ReentrancyGuar	aa73590d5265031c5bb64b5c0e7f84c44cf5f8539e6d8606b763adac784e8b2e

d.sol	
@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol	3e7aa0e0f69eec8f097ad664d525e7b3f0a3fda8dcdd97de5433ddb131db86ef
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	fa36a21bd954262006d806b988e4495562e7b50420775e2aa0deecb596fd1902
@openzeppelin/contracts/utils/Address.sol	1e0922f6c0bf6b1b8b4d480dcabb691b1359195a297bde6dc5172e79f3a1f826
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/introspection/ERC165.sol	8806a632d7b656cadb8133ff8f2acae4405b3a64d8709d93b0fa6a216a8a6154
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca0dfe5ec14916369a2935d888ff257a
@openzeppelin/contracts/utils/Strings.sol	34127ad0054df5963b0fd694c1b313d17e9114a2f426b85526d6d976210298ab

contracts/crowdsale/Crowdsale.sol	75d18d26e92cbf556cfb34d575d75d035a3a181b070cd6f7fc6bf8f5b5acd332
contracts/crowdsale/distribution/FinalizableCrowdsale.sol	86b0fedc1e18aacfdfa2a1edf12c9d9d3bf32cc5868dfa50f9abd564770d5d9f
contracts/crowdsale/validation/CappedCrowdsale.sol	55f1dbe7de91970f5d3df901a284a31070ff2300f4ede6b51e35d7c2c09ebb47
contracts/crowdsale/validation/PausableCrowdsale.sol	ac8c188fe707b59659dd8a47f1b0633cc8494836570ebd3ac362d36de92b7c99
contracts/crowdsale/validation/TimedCrowdsale.sol	9bfaadf36357ac8bb9605a0181e0e93168de8bf4e99556138dd36caa3d77a9c0
contracts/crowdsale/validation/WhitelistCrowdsale.sol	921a62b6373ff93cb353600afc92587f4eed3b90b042e1f9ee800761990e8b76
contracts/ico/payMETokenCrowdsale.sol	408fb462bb49968fae7b010dc6100b994e96c4e6c4c34840f917bfcaecf99b64
contracts/ico/payMETokenVesting.sol	d8fd864e3c39f49ce36ca539c33169535e045fbfbd09e0dc0999af014e2fde77

Introductions

The PaymeTokenCrowdsale contract implements a crowd sale mechanism.

The users deposit a specific type of token in order to vest the crowdsaled token. The deposited and the crowdsaled tokens will be defined once the Crowdsale contract is deployed. The vesting schedule starts on the finalization step of the crowdsale.

Roles

The owner is responsible for finalizing the crowd sale after the crowd sale ends.

Users have the ability to participate in the crowdsale by depositing a specific type of token.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CMA	Crowdsale Maximum Amount	Unresolved
●	STC	Succeeded Transfer Check	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved

CMA - Crowdsale Maximum Amount

Criticality	minor / informative
Location	contract.sol#L197
Status	Unresolved

Description

During the finalization step, the contract transfers the vesting amount to the corresponding address. The vested amount is calculated based on two variations. The total raised amount and some predefined proportions of the token's total supply. If the configuration is abused by the contract owner, then the vested amount might be greater than the total supply. As a result, the finalization will not be able to proceed. This could happen if the raised tokens are more than the `total supply - total shared`.

```
uint256 totalWei =  weiRaised();
uint256 tokenRate = rate();

uint256 ptShare = totalSupply.mul(projectTeamPercentage).div(100);
uint256 tdShare = totalSupply.mul(techincalDevelopersPercentage).div(100);
uint256 bdShare = totalSupply.mul(businessDevelopmentPercentage).div(100);
uint256 totalShare = ptShare.add(tdShare).add(bdShare);
uint256 totalSales = totalWei.mul(tokenRate);

uint total = totalShare.add(totalSales);
if(total >  totalSupply){
    revert TotalExceedTotalSupply(total);
}


paymeToken.safeTransfer(vestingAddress, totalShare.add(totalSales));
```

Recommendation

The contract owners should be extra careful when they are configuring the crowdsale options. Additionally, the contract could implement a mechanism that guarantees that the sum of totalShare and totalSales will always be sufficient.
A possible solution could be to guarantee that the sum of totalShare and totalSales will always be sufficient before a user buys tokens in BUSD.

STC - Succeeded Transfer Check

Criticality	minor / informative
Location	contract.sol/Crowdsale.sol#L209
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
function _forwardFunds() internal {
        _wallet.transfer(msg.value);
}
```

Recommendation

The contract should check if the result of the transfer methods is successful.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contracts/crowdsale/Crowdsale.sol#L41 contracts/crowdsale/validation/WhitelistCrowdsale.sol#L15
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
_weiRaised
INVESTOR_WHITELISTED
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/ico/payMETokenCrowdsale.sol#L52,23,50,51 contracts/crowdsale/validation/WhitelistCrowdsale.sol#L15
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.

```
businessDevelopmentPercentage
payMETokenCrowdsale
INVESTOR_WHITELISTED
projectTeamPercentage
techincalDevelopersPercentage
```

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 / informative
Location	contracts/crowdsale/validation/TimedCrowdsale.sol#L84,97 contracts/crowdsale/validation/WhitelistCrowdsale.sol#L23 contracts/crowdsale/Crowdsale.sol#L171,181,209 contracts/crowdsale/validation/CappedCrowdsale.sol#L46
Status	Unresolved

Description

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

```
_preValidatePurchase
_deliverTokens
_processPurchase
_forwardFunds
_extendTime
```

Recommendation

Remove unused functions.

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		
AccessControl	Implementation	Context, IAccessControl, ERC165		
	supportsInterface	Public		-
	hasRole	Public		-
	_checkRole	Internal		
	_checkRole	Internal		
	getRoleAdmin	Public		-
	grantRole	Public	✓	onlyRole
	revokeRole	Public	✓	onlyRole
	renounceRole	Public	✓	-

	_setupRole	Internal	✓	
	_setRoleAdmin	Internal	✓	
	_grantRole	Internal	✓	
	_revokeRole	Internal	✓	
IAccessControl	Interface			
	hasRole	External		-
	getRoleAdmin	External		-
	grantRole	External	✓	-
	revokeRole	External	✓	-
	renounceRole	External	✓	-
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Pausable	Implementation	Context		
	<Constructor>	Public	✓	-
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
IERC20Permit	Interface			
	permit	External	✓	-

	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	safePermit	Internal	✓	
	_callOptionalReturn	Private	✓	
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
Context	Implementation			
	_msgSender	Internal		

	_msgData	Internal		
ERC165	Implementation	IERC165		
	supportsInterface	Public		-
IERC165	Interface			
	supportsInterface	External		-
SafeMath	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		
Strings	Library			
	toString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
Crowdsale	Implementation	Context, ReentrancyGuard, AccessControl		
	<Constructor>	Public	✓	-
	<Fallback>	External	Payable	-
	<Receive Ether>	External	Payable	-

	token	Public		-
	wallet	Public		-
	rate	Public		-
	weiRaised	Public		-
	buyTokens	Public	Payable	nonReentrant
	_preValidatePurchase	Internal		
	_postValidatePurchase	Internal		
	_deliverTokens	Internal	✓	
	_processPurchase	Internal	✓	
	_updatePurchasingState	Internal	✓	
	_getTokenAmount	Internal		
	_forwardFunds	Internal	✓	
FinalizableCrowdsale	Implementation	TimedCrowdsale		
	<Constructor>	Public	✓	-
	finalized	Public		-
	finalize	Public	✓	-
	_finalization	Internal	✓	
CappedCrowdsale	Implementation	Crowdsale		
	<Constructor>	Public	✓	-
	cap	Public		-
	capReached	Public		-
	_preValidatePurchase	Internal		
PausableCrowdsale	Implementation	Crowdsale, Pausable, Ownable		
	_preValidatePurchase	Internal		whenNotPaused
	pause	Public	✓	onlyOwner whenNotPaused
	unpause	Public	✓	onlyOwner whenPaused

TimedCrowdsale	Implementation	Crowdsale		
	<Constructor>	Public	✓	-
	openingTime	Public		-
	closingTime	Public		-
	isOpen	Public		-
	hasClosed	Public		-
	_preValidatePurchase	Internal		onlyWhileOpen
	_extendTime	Internal	✓	
WhitelistCrowdsale	Implementation	AccessControl, Crowdsale		
	_preValidatePurchase	Internal		
	addWhitelisted	Public	✓	onlyRole
payMETokenCrowdsale	Implementation	Ownable, CappedCrowdsale, TimedCrowdsale, WhitelistCrowdsale, FinalizableCrowdsale, PausableCrowdsale		
	<Constructor>	Public	✓	Crowdsale CappedCrowdsale TimedCrowdsale
	buyTokensInBUSD	Public	Payable	nonReentrant
	buyTokens	Public	Payable	nonReentrant
	_forwardFunds	Internal	✓	
	_preValidatePurchase	Internal		
	createInvestor	Internal	✓	
	_processPurchase	Internal	✓	
	_updatePurchasingState	Internal	✓	
	_finalization	Internal	✓	

	createInvestors	Public	✓	-
	finalize	Public	✓	onlyOwner
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 about 2 months.

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

Summary

The PaymeTokenCrowdsale contract is responsible for exchanging BUSD for native tokens. In order to vest them. This audit investigates security issues and mentions business logic concerns and potential improvements.

We state that owner privileges are necessary and required for proper protocol operations. Thus, we emphasize the contract owner be extra careful with the credentials.

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