QuillAudits

Audit Report
March, 2022

For

PlayVRS

Contents

Scope of the Audit

The scope of this audit was to analyze and document the PlayVRS Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	2	0	0
Closed	0	0	5	3

Introduction

During the period of **March 07, 2022 to March 21, 2022** - QuillAudits Team performed a security audit for PlayVRS smart contracts.

The code for the audit was taken from following the official link:
https://github.com/PlayVRS/SmartContracts

V	Date	Commit ID
1	March 7	57217e54a814318759f241f919aa150d92c081ea
2	March 21	c9eb690d573ac42d17e82eef4b1e507b78464b7c

Issues Found – Code Review / Manual Testing

A. Contract - PlayverseToken.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational Issues

No issues were found.

B. Contract - PresaleContract.sol

High severity issues

No issues were found.

Medium severity issues

B.1 Centralization Risks

Description

The role Governance has the authority to update the critical settings:

- withdrawToken()
- withdraw()
- addStableCoins()

Remediation

We advise the client to handle the governance account carefully to avoid any potential hack. We also recommend the client to consider the following solutions:

1. With reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement;

Status: Acknowledged

B.2 Insufficient check for setTokenDecimal()

Description

setTokenDecimal() does not check if the stable coin exists, whereby the stable coin might get the default value set by the contract if it’s not set correctly via setTokenDecimal().

We recommend adding a check in setTokenDecimal() to ensure the stable coin exists and added.

Status: Acknowledged

The Auditee stated that setTokenDecimal() is not only used for stable coins, but also the coin they are selling. It's by design a common configuration method for relative coins.

Low severity issues

B.3 Lack of event emissions

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- addStableCoins()
- removeStableCoin()
- setTokenDecimal()
- setWhitelists()
- setWhitelist()

Remediation

We recommend emitting an event to log the update of the above variables for those above-mentioned functions.

Status: Fixed in version 01

B.4 Tautology or Contradiction Issue

Line	Code/Function
192	(from >= 0) && (from <= to) && (to < whitelistUserSet.length()),

Description

from is an address type variable, so from >= 0 will be always true. This comparison is a tautology that will waste gas during the execution.

Remediation

Fix the incorrect comparison by changing the value type or the comparison.

Status: Fixed in version 01

B.5 Lack of Zero address validation in constructor()

Description

To favor explicitness, consider adding a check for all functions that are taking address parameters in the entire codebase. Although most of the functions throughout the codebase properly validate function inputs, there are some instances of functions that do not check for zero address

- constructor() - does not check for zero address
- setWhitelist() and setWhitelists() function

Remediation

Consider implementing require statements where appropriate to validate all user-controlled input, including governance functions, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status: Fixed in version 01

Informational Issues

B.6 Typos

Description

Throughout PlayVRS's codebase, there are a few typos in the code and in the comments. We list them here:

- Withdrawed should be Withdrawn

Status: Fixed in version 01

B.7 Public function that could be declared external

Description

The following public functions that are never called by the contract should be declared external to save gas:

- buyPresale()
- withdrawToken()
- withdraw()

Remediation
Use the external attribute for functions never called from the contract.

Status: Fixed in version 01

Contract - VestingContract.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

C.1 Lack of Zero address validation in constructor()

Description
To favor explicitness, consider adding a check for all functions that are taking address parameters in the entire codebase. Although most of the functions throughout the codebase properly validate function inputs, there are some instances of functions that do not. One example is:
- constructor() - does not check for zero address
- transferManagement() function
- addBeneficiary() function

Remediation
Consider implementing require statements where appropriate to validate all user-controlled input, including governance functions, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas

Status: Fixed in version 01

C.2 Lack of event emissions

Description

The missing event makes it difficult to track off-chain liquidity fee changes. An event should be emitted for significant transactions calling the following functions:

- addBeneficiary()

Remediation

We recommend emitting an event to log the update of the above variables for those above-mentioned functions.

Status: Fixed in version 01

Informational Issues

C.3 Public function that could be declared external

Description

The following public functions that are never called by the contract should be declared external to save gas:

- transferManagement()
- addBeneficiary()
- withdraw()
- release()
- changeBeneficiary()

Remediation

Use the external attribute for functions never called from the contract.

Status: Fixed in version 01

Functional Testing

PlayverseToken

Testing getters

- Should get name of the contract
- Should get symbol of the contract
- Should get decimals of the contract
- Should get total supply of the contract

Testing Setters

- Should get balance of the tokenHolder
- Should return if user has no balance (45ms)
- Owner should transfer to account 1 (44ms)
- Should call approval to account 2 (45ms)
- Testing decreaseAllowance() and increaseAllowance() (46ms)
- Testing TransferFrom() (60ms)

PresaleContract

Testing getters

- addStableCoins() should be called only by owner (80ms)
- addStableCoins() should return true
- addStableCoins() adds the same stable coins
- removeStableCoin() should be called only by owner
- removeStableCoin() should be remove a coin (38ms)
- addStableCoins() adds multiple stable coins
- setTokenDecimal() should be called only by owner
- [Acknowledged] setTokenDecimal() should change USDC decimal
- setWhitelists() should be called only by owner
- setWhitelists() should reverted if addrs.length != amounts.length
- setWhitelists() should return true if all passed
- setWhitelist() should be called only by owner
- setWhitelist() should be called only by owner (41ms)

Testing buyPresale

- buyPresale() should be called by only whitelisted users (1763ms)
- buyPresale() should be reverted if not enough token balance (726ms)
- buyPresale() should be reverted if coin is not supported (72ms)
- buyPresale() should be reverted if Insufficient Stable Coin allowance(75ms)
- buyPresale() should return true if all passed (1550ms)
- one user buy multiple times (131ms)
- User should get Exceed the purchase limit when overbought (74ms)
- buyPresale() with another stable coin should return true if all passed (843ms)

Testing withdrawToken()

- withdrawToken() should return correct USDC amount (680ms)

Testing withdraw()

- withdraw() should return correct amount of all token

VestingContract.sol

Testing getters

- Get manager address
- Get token address

Testing setters

- transferManagement() should be called only by manager
- transferManagement() should return true when all passed
- addBeneficiary() should add account2 from account 1 (74ms)
- addBeneficiary() should be reverted if Beneficiary already exists
- changeBeneficiary() should be reverted if beneficiaryAmount[_originalBeneficiary] == 0
- changeBeneficiary() should be reverted if beneficiaryAmount[_newBeneficiary] != 0
- changeBeneficiary() should be reverted if msg.sender != _originalBeneficiary || msg.sender != manager
- changeBeneficiary() should called by the manager
- vestingAmountSchedule() should return '0% of tokens are unlocked' when timestamp < startSecond + 0
- vestingAmountSchedule() should return '30% tokens are unlocked' when startSecond + 0 <= timestamp < startSecond + 1000

- vestingAmountSchedule() should return '70% tokens are unlocked' when startSecond + 1000 <= timestamp < startSecond + 2000
- vestingAmountSchedule() should return '100% tokens are unlocked' when startSecond + 2000` <= timestamp
- release() should be called only by beneficiaries
- release() should be reverted when scheduledRelease < released[msg.sender]
- release() should release 30% of the token (43ms)
- release() should release 70% of the token (51ms)
- release() should release the rest of the token (49ms)
- release() should be reverted if user's balance = 0

Automated Tests

PlayverseToken

Slither

```
INFO:Detectors:
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#280-295) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploy
ing with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PlayverseToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Solhint Linter

```
PresaleContract.sol
    2:1  error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement                          compiler-version
   59:5  warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
   87:9  warning  Error message for require is too long                                                           reason-string
  118:9  warning  Error message for require is too long                                                           reason-string
  131:9  warning  Error message for require is too long                                                           reason-string

✖ 5 problems (1 error, 4 warnings)
```

PresaleContract

Slither

```
INFO:Detectors:
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#280-295) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.
6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7
.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploy
ing with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (PlayverseToken.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
        - ERC20.symbol() (@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
        - ERC20.decimals() (@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
        - ERC20.totalSupply() (@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin/contracts/access/Ownable.sol#62-65)
buyPresale(address,uint256) should be declared external:
        - PresaleContract.buyPresale(address,uint256) (PresaleContract.sol#108-139)
withdrawToken(address,address,uint256) should be declared external:
        - PresaleContract.withdrawToken(address,address,uint256) (PresaleContract.sol#142-149)
withdraw(address) should be declared external:
        - PresaleContract.withdraw(address) (PresaleContract.sol#152-168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionDelegateCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Context._msgData() (@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#158-160) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#297-299) is never used and should be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#196-198) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#335-337) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#175-177) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#314-316) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#182-184) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#321-323) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#168-170) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#307-309) is never used and should be removed
EnumerableSet.values(EnumerableSet.Bytes32Set) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#208-210) is never used and should be removed
EnumerableSet.values(EnumerableSet.UintSet) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#347-356) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.0 (PresaleContract.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
PresaleContract.addStableCoins(address[]) (PresaleContract.sol#65-70) ignores return value by stableCoinSet.add(coin) (PresaleContract.sol#68)
PresaleContract.removeStableCoin(address) (PresaleContract.sol#73-75) ignores return value by stableCoinSet.remove(stableCoin) (PresaleContract.sol#74)
PresaleContract.setWhitelists(address[],uint256[]) (PresaleContract.sol#83-95) ignores return value by whitelistUserSet.add(addrs[i]) (PresaleContract.sol#93)
PresaleContract.setWhitelist(address,uint256) (PresaleContract.sol#98-101) ignores return value by whitelistUserSet.add(addr) (PresaleContract.sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
PresaleContract.constructor(address,uint256)._tokenAddress (PresaleContract.sol#59) lacks a zero-check on :
                - tokenAddress = _tokenAddress (PresaleContract.sol#60)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
PresaleContract.withdraw(address) (PresaleContract.sol#152-168) has external calls inside a loop: balance = IERC20(coin).balanceOf(address(this)) (PresaleCont
ract.sol#163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#213-216)
EnumerableSet.values(EnumerableSet.AddressSet) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#274-283) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#278-280)
EnumerableSet.values(EnumerableSet.UintSet) (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#347-356) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#351-353)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.1']
        - ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/structs/EnumerableSet.sol#4)
        - ^0.8.0 (PresaleContract.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

Solhint Linter

```
PresaleContract.sol
    2:1  error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement                          compiler-version
   59:5  warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
   87:9  warning  Error message for require is too long                                                            reason-string
  118:9  warning  Error message for require is too long                                                            reason-string
  131:9  warning  Error message for require is too long                                                            reason-string
```

VestingContract.sol

Slither

```
INFO:Detectors:
VestingContract.constructor(address,address,uint256,uint256[],uint256[])._manager (VestingContract.sol#61) lacks a zero-check on :
                - manager = _manager (VestingContract.sol#67)
VestingContract.constructor(address,address,uint256,uint256[],uint256[])._tokenAddress (VestingContract.sol#62) lacks a zero-check on :
                - tokenAddress = _tokenAddress (VestingContract.sol#68)
VestingContract.transferManagement(address)._newManager (VestingContract.sol#79) lacks a zero-check on :
                - manager = _newManager (VestingContract.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
VestingContract.vestingProportionSchedule(uint256) (VestingContract.sol#157-168) uses timestamp for comparisons
        Dangerous comparisons:
        - timestamp < startSecond + stageSecond[i] (VestingContract.sol#163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.1']
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (VestingContract.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionDelegateCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.1 (@openzeppelin/contracts/utils/Address.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (VestingContract.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.1 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter VestingContract.transferManagement(address)._newManager (VestingContract.sol#79) is not in mixedCase
Parameter VestingContract.addBeneficiary(address,uint256)._beneficiary (VestingContract.sol#96) is not in mixedCase
Parameter VestingContract.addBeneficiary(address,uint256)._amount (VestingContract.sol#96) is not in mixedCase
Parameter VestingContract.changeBeneficiary(address,address)._originalBeneficiary (VestingContract.sol#175) is not in mixedCase
Parameter VestingContract.changeBeneficiary(address,address)._newBeneficiary (VestingContract.sol#176) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
transferManagement(address) should be declared external:
        - VestingContract.transferManagement(address) (VestingContract.sol#79-85)
addBeneficiary(address,uint256) should be declared external:
        - VestingContract.addBeneficiary(address,uint256) (VestingContract.sol#96-105)
release() should be declared external:
        - VestingContract.release() (VestingContract.sol#110-138)
changeBeneficiary(address,address) should be declared external:
        - VestingContract.changeBeneficiary(address,address) (VestingContract.sol#174-192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Solhint Linter

```
VestingContract.sol
   2:1   error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement                          compiler-version
  30:5   warning  Explicitly mark visibility of state                                                             state-visibility
  60:5   warning  Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  func-visibility
  71:9   warning  Provide an error message for require                                                            reason-string
  72:9   warning  Provide an error message for require                                                            reason-string
 118:13  warning  Avoid to make time-based decisions in your business logic                                       not-rely-on-time
 181:9   warning  Error message for require is too long                                                          reason-string
```

Results

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the PlayVRS platform. We performed our audit according to the procedure described above.

The audit showed several medium, low, and informational severity issues. In the end, the majority of the issues were fixed and acknowledged by the Auditee.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the PlayVRS platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the PlayVRS Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
March, 2022

For

PlayVRS

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com