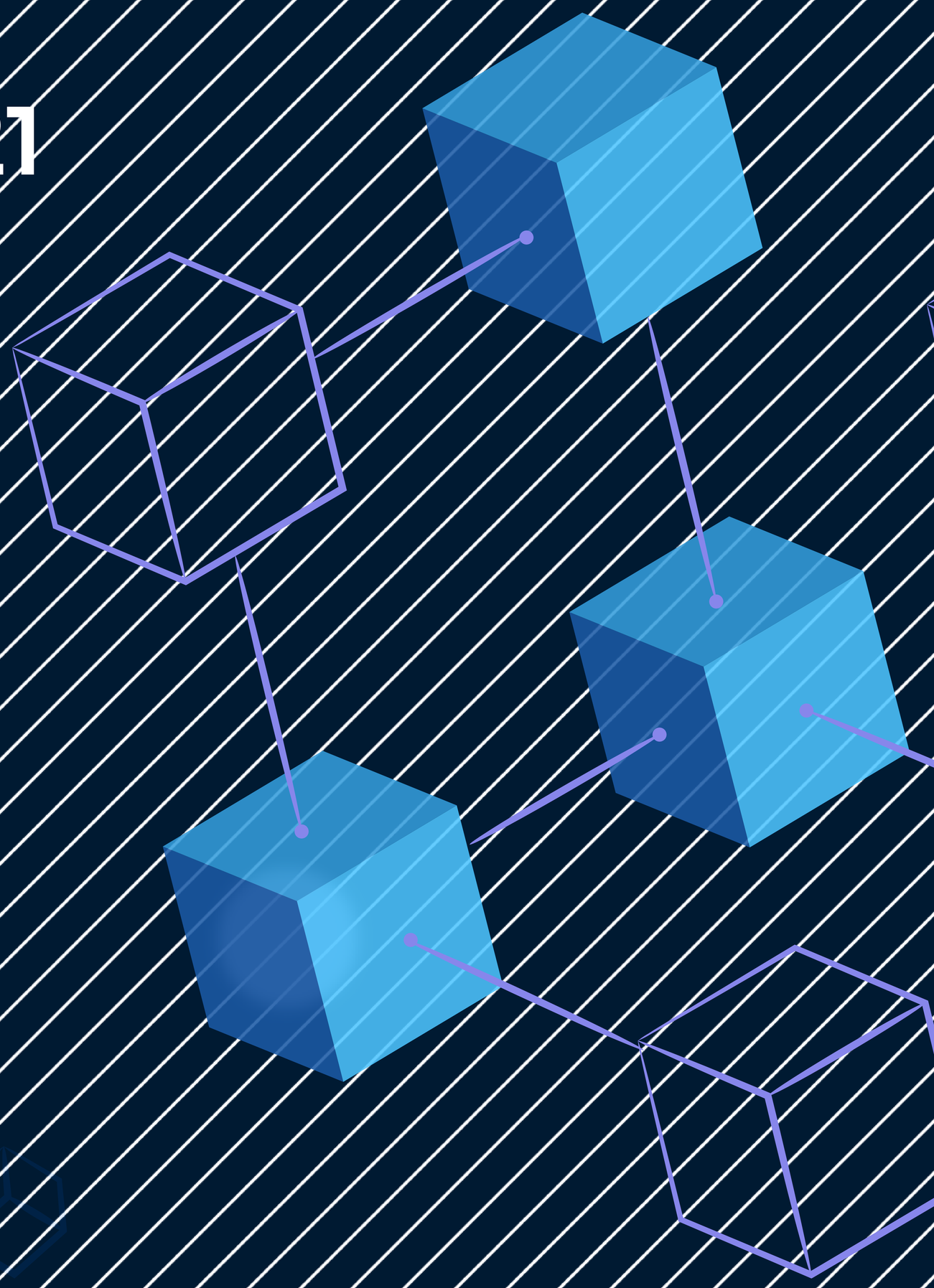
QuillAudits

Audit Report
November, 2021



For

99STARZ

Contents

Contents

Scope of the Audit

The scope of this audit was to analyze and document the 99Starz smart contracts 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	0	2	0
Closed	0	2	5	0

Introduction

During the period of **November 15, 2021, to November 26, 2021** - QuillAudits Team performed a security audit for **99Starz** smart contracts.

The code for the audit was taken from the following official repo of **99Starz**: https://gitlab.com/qundeel347/stz

Note	Date	Commit hash
Version 1	November	b29fd097cb4017e0fc8a24194229f528a5d84cd6
Version 2	November	6849127dbbad14cedef40626b0aacce9d252f84b

Issues Found

A. Contract – 99starz

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

A.1 Approve Race

```
Line 323:
function approve(address spender, uint256 amount) public virtual
        override returns (bool) {
            _approve(_msgSender(), spender, amount);
            return true;
        }
```

Description
The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Remediation
Use increaseAllowance and decreaseAllowance functions to modify the approval amount instead of using the approve function to modify it.

Solved
The 99Starz team has fixed this issue using two solutions :
- The team restricts users to use approve only when the amount is zero or less than zero.
- The changed the approve function, In case the current allowance and amount is the same we will simply return true. In case the amount is high we will use an increased allowance and in case the amount is low we will use a decreased allowance.
Repo : https://gitlab.com/qundeel347/stz/-/blob/99starzz-tests/contracts/99starz.sol

A.2 Renounce Ownership

```
Line 602:
contract ERC20_99starz is ERC20Burnable, Ownable {
```

Description
Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Remediation
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Fixed
The 99Starz team has fixed the issue by removing the renounceOwnership function.

A.3 Floating Pragma

```
Line 4:
pragma solidity ^0.8.0;
```

Description

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Fixed

The 99Starz team has fixed the issue by locking the pragma version.

B. Contract – BaseVesting

High severity issues

No issues were found.

Medium severity issues

B.1 Missing Value Verification

```
Line 955:
    constructor(address token_, uint256 _cliff, uint256 _start,
        uint256 _duration, uint256 _slicePeriodSeconds) {
        require(token_ != address(0x0));
        _token = IERC20(token_);
        cliff_ = _cliff;
        start = _start;
        duration = _duration;
        slicePeriodSeconds = _slicePeriodSeconds;
    }
```

Description

Certain functions lack a safety check in the values, the values of the arguments should include some safety checks test, otherwise, the contract's functionality may get hurt.

Remediation

Add require conditions which verifies that all the values are different than 0 and the start argument is higher than block.timestamp.

Fixed

The 99Starz team has fixed the issue by verifying the values provided from the arguments, and accepted the start variable to be initialized with a value that is less than the current timestamp.

Low severity issues

B.2 Renounce Ownership

```
Line 888:
contract PrivateVesting is Ownable, ReentrancyGuard, Whitelist {
```

Description
Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Remediation
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Fixed
The 99Starz team has fixed the issue by removing the renounceOwnership function.

B.3 Usage Of block.timestamp

```
Line 1249:
    function getCurrentTime() internal view virtual returns (uint256) {
        return block.timestamp;
}
```

Description

Block.timestamp is used in the contract. The variable block is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right, all what is guaranteed is that it is higher than the timestamp of the previous block.

Remediation

You can use an Oracle to get the exact time or verify if a delay of 900 seconds won't destroy the logic of the staking contract.

Acknowledged

The 99Starz team has acknowledged the risk, saying that vesting will be daily based so some time delay in seconds will not affect the logic of the smart contract.

C. Contract – privateVesting

High severity issues

No issues were found.

Medium severity issues

C.1 Missing Value Verification

```
Line 947:
    constructor(address token_, uint256 _cliff, uint256 _start,
        uint256 _duration, uint256 _slicePeriodSeconds) {
        require(token_ != address(0x0));
        _token = IERC20(token_);
        cliff_ = _cliff;
        start = _start;
        duration = _duration;
        slicePeriodSeconds = _slicePeriodSeconds;
    }
```

Description
Certain functions lack a safety check in the values; the values of the arguments should include some safety checks, otherwise, the contract's functionality may get hurt.

Remediation
Add require conditions which verifies that all the values are different than 0 and the start argument is higher than block.timestamp.

Fixed
The 99Starz team has fixed the issue by verifying the values provided from the arguments, and accepted the start variable to be initialized with a value that is less than the current timestamp.

Low severity issues

C.2 Renounce Ownership

```
Line 887:
contract TokenVesting is Ownable, ReentrancyGuard, Whitelist {
```

Description

Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Remediation

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Fixed

The 99Starz team has fixed the issue by removing the renounceOwnership function.

C.3 Usage Of block.timestamp

```
Line 1235:
    function getCurrentTime() internal view virtual returns (uint256) {
        return block.timestamp;
}
```

Description

Block.timestamp is used in the contract. The variable block is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right, all that is guaranteed is that it is higher than the timestamp of the previous block.

Remediation

You can use an Oracle to get the exact time or verify if a delay of 900 seconds won't destroy the logic of the staking contract.

Acknowledged

The 99Starz team has acknowledged the risk, saying that vesting will be daily based so some time delay in seconds will not affect the logic of the smart contract.

Test Cases for Functional Testing

Some of the test cases which were part of the functional testing are listed below:

99starz.sol

- Should create the ERC20 Token and mint the initial Supply **PASS**

BaseVesting.sol

- Should create a vesting Schedule but only for Allowed persons **PASS**
- Should revoke a vesting schedule **PASS**
- Should Withdraw a Specific Amount **PASS**

Automated Tests

Slither

```
ERC20_99starz.constructor(string,string,uint256,address).name (99starz.sol#662) shadows:
        - ERC20.name() (99starz.sol#263-265) (function)
        - IERC20Metadata.name() (99starz.sol#102) (function)
ERC20_99starz.constructor(string,string,uint256,address).symbol (99starz.sol#663) shadows:
        - ERC20.symbol() (99starz.sol#271-273) (function)
        - IERC20Metadata.symbol() (99starz.sol#107) (function)
ERC20_99starz.constructor(string,string,uint256,address).owner (99starz.sol#665) shadows:
        - Ownable.owner() (99starz.sol#165-167) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (99starz.sol#130-132) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (99starz.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.5 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Contract ERC20_99starz (99starz.sol#654-669) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (99starz.sol#184-186)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (99starz.sol#192-198)
name() should be declared external:
        - ERC20.name() (99starz.sol#263-265)
symbol() should be declared external:
        - ERC20.symbol() (99starz.sol#271-273)
decimals() should be declared external:
        - ERC20.decimals() (99starz.sol#288-290)
totalSupply() should be declared external:
        - ERC20.totalSupply() (99starz.sol#295-297)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (99starz.sol#302-310)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (99starz.sol#320-328)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (99starz.sol#350-359)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (99starz.sol#374-391)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (99starz.sol#405-416)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (99starz.sol#432-447)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (99starz.sol#614-616)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (99starz.sol#629-639)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
Reentrancy in PrivateVesting.revoke(bytes32) (BaseVesting.sol#1155-1183):
        External calls:
        - release(vestingScheduleId,vestedAmount) (BaseVesting.sol#1174)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (BaseVesting.sol#912-915)
                - (success,returndata) = target.call{value: value}(data) (BaseVesting.sol#601-603)
                - _token.safeTransfer(beneficiaryPayable,amount) (BaseVesting.sol#1227)
        External calls sending eth:
        - release(vestingScheduleId,vestedAmount) (BaseVesting.sol#1174)
                - (success,returndata) = target.call{value: value}(data) (BaseVesting.sol#601-603)
        State variables written after the call(s):
        - vestingSchedule.revoked = true (BaseVesting.sol#1182)
        - vestingSchedulesTotalAmount = vestingSchedulesTotalAmount.sub(unreleased) (BaseVesting.sol#1179-1181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

PrivateVesting._computeReleasableAmount(PrivateVesting.VestingSchedule) (BaseVesting.sol#1335-1359) performs a multiplication on the result of a division:
        -vestedSlicePeriods = timeFromStart.div(secondsPerSlice) (BaseVesting.sol#1350)
        -vestedSeconds = vestedSlicePeriods.mul(secondsPerSlice) (BaseVesting.sol#1351)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

PrivateVesting.onlyIfVestingScheduleExists(bytes32) (BaseVesting.sol#1005-1008) uses a dangerous strict equality:
        - require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (BaseVesting.sol#1006)
PrivateVesting.onlyIfVestingScheduleNotRevoked(bytes32) (BaseVesting.sol#1013-1017) uses a dangerous strict equality:
        - require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (BaseVesting.sol#1014)
PrivateVesting.onlyIfVestingScheduleNotRevoked(bytes32) (BaseVesting.sol#1013-1017) uses a dangerous strict equality:
        - require(bool)(vestingSchedules[vestingScheduleId].revoked == false) (BaseVesting.sol#1015)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

PrivateVesting.createVestingSchedule(address,bool,uint256) (BaseVesting.sol#1112-1149) should emit an event for:
        - vestingSchedulesTotalAmount = vestingSchedulesTotalAmount.add(_amount) (BaseVesting.sol#1144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

PrivateVesting.revoke(bytes32) (BaseVesting.sol#1155-1183) uses timestamp for comparisons
        Dangerous comparisons:
        - vestedAmount > 0 (BaseVesting.sol#1173)
PrivateVesting.release(bytes32,uint256) (BaseVesting.sol#1202-1228) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(vestedAmount >= amount,TokenVesting: cannot release tokens, not enough vested tokens) (BaseVesting.sol#1217-1220)
PrivateVesting._computeReleasableAmount(PrivateVesting.VestingSchedule) (BaseVesting.sol#1335-1359) uses timestamp for comparisons
        Dangerous comparisons:
        - (currentTime < start.add(cliff_)) || vestingSchedule.revoked == true (BaseVesting.sol#1342)
        - currentTime >= start.add(duration) (BaseVesting.sol#1345)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (BaseVesting.sol#478-488) uses assembly
        - INLINE ASM (BaseVesting.sol#484-486)
Address.verifyCallResult(bool,bytes,string) (BaseVesting.sol#684-704) uses assembly
        - INLINE ASM (BaseVesting.sol#696-699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Whitelist.addAddress(address) (BaseVesting.sol#936-940) compares to a boolean constant:
        -require(bool,string)(whitelistedMap[_address] != true,Account already exist) (BaseVesting.sol#937)
Whitelist.removeAddress(address) (BaseVesting.sol#942-946) compares to a boolean constant:
        -require(bool)(whitelistedMap[_address] != false) (BaseVesting.sol#943)
PrivateVesting.createVestingSchedule(address,bool,uint256) (BaseVesting.sol#1112-1149) compares to a boolean constant:
        -require(bool,string)(isWhitelisted(_beneficiary) != false,Whitelist: Beneficiary must be pre-regesitred) (BaseVesting.sol#1127-1130)
PrivateVesting.revoke(bytes32) (BaseVesting.sol#1155-1183) compares to a boolean constant:
        -require(bool,string)(vestingSchedule.revocable == true,TokenVesting: vesting is not revocable) (BaseVesting.sol#1162-1165)
PrivateVesting._computeReleasableAmount(PrivateVesting.VestingSchedule) (BaseVesting.sol#1335-1359) compares to a boolean constant:
        -(currentTime < start.add(cliff_)) || vestingSchedule.revoked == true (BaseVesting.sol#1342)
PrivateVesting.onlyIfVestingScheduleExists(bytes32) (BaseVesting.sol#1005-1008) compares to a boolean constant:
        -require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (BaseVesting.sol#1006)
PrivateVesting.onlyIfVestingScheduleNotRevoked(bytes32) (BaseVesting.sol#1013-1017) compares to a boolean constant:
        -require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (BaseVesting.sol#1014)
PrivateVesting.onlyIfVestingScheduleNotRevoked(bytes32) (BaseVesting.sol#1013-1017) compares to a boolean constant:
        -require(bool)(vestingSchedules[vestingScheduleId].revoked == false) (BaseVesting.sol#1015)
PrivateVesting.onlyAllowed() (BaseVesting.sol#1019-1023) compares to a boolean constant:
        -require(bool)(true == isWhitelisted(msg.sender) || isOwner == true) (BaseVesting.sol#1021)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.5', '^0.8.0', '^0.8.5']
        - ^0.8.0 (BaseVesting.sol#2)
        - ^0.8.0 (BaseVesting.sol#249)
        - ^0.8.0 (BaseVesting.sol#292)
        - ^0.8.0 (BaseVesting.sol#316)
        - ^0.8.5 (BaseVesting.sol#392)
        - ^0.8.0 (BaseVesting.sol#455)
        - ^0.8.0 (BaseVesting.sol#709)
        - ^0.8.0 (BaseVesting.sol#800)
        - 0.8.5 (BaseVesting.sol#956)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (BaseVesting.sol#537-542) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (BaseVesting.sol#569-581) is never used and should be removed
Address.functionDelegateCall(address,bytes) (BaseVesting.sol#649-659) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (BaseVesting.sol#667-676) is never used and should be removed
Address.functionStaticCall(address,bytes) (BaseVesting.sol#613-624) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (BaseVesting.sol#632-641) is never used and should be removed
Address.sendValue(address,uint256) (BaseVesting.sol#506-517) is never used and should be removed
Context._msgData() (BaseVesting.sol#309-311) is never used and should be removed
Math.average(uint256,uint256) (BaseVesting.sol#273-276) is never used and should be removed
Math.ceilDiv(uint256,uint256) (BaseVesting.sol#284-287) is never used and should be removed
Math.max(uint256,uint256) (BaseVesting.sol#258-260) is never used and should be removed
Math.min(uint256,uint256) (BaseVesting.sol#265-267) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (BaseVesting.sol#844-860) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (BaseVesting.sol#878-899) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (BaseVesting.sol#862-876) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (BaseVesting.sol#825-835) is never used and should be removed
SafeMath.div(uint256,uint256,string) (BaseVesting.sol#209-218) is never used and should be removed
```

```
SafeMath.div(uint256,uint256,string) (BaseVesting.sol#209-218) is never used and should be removed
SafeMath.mod(uint256,uint256) (BaseVesting.sol#169-171) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (BaseVesting.sol#235-244) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (BaseVesting.sol#186-195) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (BaseVesting.sol#20-30) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (BaseVesting.sol#74-83) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (BaseVesting.sol#90-99) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (BaseVesting.sol#53-67) is never used and should be removed
SafeMath.trySub(uint256,uint256) (BaseVesting.sol#37-46) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (BaseVesting.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (BaseVesting.sol#249) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (BaseVesting.sol#292) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (BaseVesting.sol#316) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.5 (BaseVesting.sol#392) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (BaseVesting.sol#455) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (BaseVesting.sol#709) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (BaseVesting.sol#800) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.5 (BaseVesting.sol#956) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.5 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (BaseVesting.sol#506-517):
        - (success) = recipient.call{value: amount}() (BaseVesting.sol#512)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (BaseVesting.sol#589-605):
        - (success,returndata) = target.call{value: value}(data) (BaseVesting.sol#601-603)
Low level call in Address.functionStaticCall(address,bytes,string) (BaseVesting.sol#632-641):
        - (success,returndata) = target.staticcall(data) (BaseVesting.sol#639)
Low level call in Address.functionDelegateCall(address,bytes,string) (BaseVesting.sol#667-676):
        - (success,returndata) = target.delegatecall(data) (BaseVesting.sol#674)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Whitelist.whitelisted(address)._address (BaseVesting.sol#932) is not in mixedCase
Parameter Whitelist.addAddress(address)._address (BaseVesting.sol#936) is not in mixedCase
Parameter Whitelist.removeAddress(address)._address (BaseVesting.sol#942) is not in mixedCase
Parameter Whitelist.isWhitelisted(address)._address (BaseVesting.sol#948) is not in mixedCase
Constant Whitelist.version (BaseVesting.sol#927) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter PrivateVesting.getVestingSchedulesCountByBeneficiary(address)._beneficiary (BaseVesting.sol#1053) is not in mixedCase
Parameter PrivateVesting.createVestingSchedule(address,bool,uint256)._beneficiary (BaseVesting.sol#1113) is not in mixedCase
Parameter PrivateVesting.createVestingSchedule(address,bool,uint256)._revocable (BaseVesting.sol#1114) is not in mixedCase
Parameter PrivateVesting.createVestingSchedule(address,bool,uint256)._amount (BaseVesting.sol#1115) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (BaseVesting.sol#367-369)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (BaseVesting.sol#375-381)
whitelisted(address) should be declared external:
        - Whitelist.whitelisted(address) (BaseVesting.sol#932-934)
```

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (BaseVesting.sol#367-369)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (BaseVesting.sol#375-381)
whitelisted(address) should be declared external:
        - Whitelist.whitelisted(address) (BaseVesting.sol#932-934)
addAddress(address) should be declared external:
        - Whitelist.addAddress(address) (BaseVesting.sol#936-940)
removeAddress(address) should be declared external:
        - Whitelist.removeAddress(address) (BaseVesting.sol#942-946)
createVestingSchedule(address,bool,uint256) should be declared external:
        - PrivateVesting.createVestingSchedule(address,bool,uint256) (BaseVesting.sol#1112-1149)
revoke(bytes32) should be declared external:
        - PrivateVesting.revoke(bytes32) (BaseVesting.sol#1155-1183)
withdraw(uint256) should be declared external:
        - PrivateVesting.withdraw(uint256) (BaseVesting.sol#1189-1195)
computeReleasableAmount(bytes32) should be declared external:
        - PrivateVesting.computeReleasableAmount(bytes32) (BaseVesting.sol#1242-1252)
getWithdrawableAmount() should be declared external:
        - PrivateVesting.getWithdrawableAmount() (BaseVesting.sol#1270-1272)
computeNextVestingScheduleIdForHolder(address) should be declared external:
        - PrivateVesting.computeNextVestingScheduleIdForHolder(address) (BaseVesting.sol#1277-1287)
getLastVestingScheduleForHolder(address) should be declared external:
        - PrivateVesting.getLastVestingScheduleForHolder(address) (BaseVesting.sol#1293-1305)
computeByteIdForAddress(address) should be declared external:
        - PrivateVesting.computeByteIdForAddress(address) (BaseVesting.sol#1322-1328)
getLockedAmount() should be declared external:
        - PrivateVesting.getLockedAmount() (BaseVesting.sol#1361-1363)
startTime() should be declared external:
        - PrivateVesting.startTime() (BaseVesting.sol#1369-1371)
endTime() should be declared external:
        - PrivateVesting.endTime() (BaseVesting.sol#1373-1375)
getCliff() should be declared external:
        - PrivateVesting.getCliff() (BaseVesting.sol#1377-1379)
getAmountWithdrawn() should be declared external:
        - PrivateVesting.getAmountWithdrawn() (BaseVesting.sol#1381-1383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Reentrancy in TokenVesting.revoke(bytes32) (privateVesting.sol#1148-1171):
        External calls:
        - release(vestingScheduleId,vestedAmount) (privateVesting.sol#1162)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (privateVesting.sol#912-915)
                - (success,returndata) = target.call{value: value}(data) (privateVesting.sol#601-603)
                - _token.safeTransfer(beneficiaryPayable,amount) (privateVesting.sol#1215)
        External calls sending eth:
        - release(vestingScheduleId,vestedAmount) (privateVesting.sol#1162)
                - (success,returndata) = target.call{value: value}(data) (privateVesting.sol#601-603)
        State variables written after the call(s):
        - vestingSchedule.revoked = true (privateVesting.sol#1170)
        - vestingSchedulesTotalAmount = vestingSchedulesTotalAmount.sub(unreleased) (privateVesting.sol#1167-1169)
```

```
TokenVesting._computeReleasableAmount(TokenVesting.VestingSchedule) (privateVesting.sol#1323-1347) performs a multiplication on the result of a division:
        -vestedSlicePeriods = timeFromStart.div(secondsPerSlice) (privateVesting.sol#1338)
        -vestedSeconds = vestedSlicePeriods.mul(secondsPerSlice) (privateVesting.sol#1339)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

TokenVesting.onlyIfVestingScheduleExists(bytes32) (privateVesting.sol#1005-1008) uses a dangerous strict equality:
        - require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (privateVesting.sol#1006)
TokenVesting.onlyIfVestingScheduleNotRevoked(bytes32) (privateVesting.sol#1013-1017) uses a dangerous strict equality:
        - require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (privateVesting.sol#1014)
TokenVesting.onlyIfVestingScheduleNotRevoked(bytes32) (privateVesting.sol#1013-1017) uses a dangerous strict equality:
        - require(bool)(vestingSchedules[vestingScheduleId].revoked == false) (privateVesting.sol#1015)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

TokenVesting.createVestingSchedule(address,bool,uint256) (privateVesting.sol#1106-1142) should emit an event for:
        - vestingSchedulesTotalAmount = vestingSchedulesTotalAmount.add(_amount) (privateVesting.sol#1137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

TokenVesting.revoke(bytes32) (privateVesting.sol#1148-1171) uses timestamp for comparisons
        Dangerous comparisons:
        - vestedAmount > 0 (privateVesting.sol#1161)
TokenVesting.release(bytes32,uint256) (privateVesting.sol#1190-1216) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(vestedAmount >= amount,TokenVesting: cannot release tokens, not enough vested tokens) (privateVesting.sol#1205-1208)
TokenVesting._computeReleasableAmount(TokenVesting.VestingSchedule) (privateVesting.sol#1323-1347) uses timestamp for comparisons
        Dangerous comparisons:
        - (currentTime < start.add(cliff_)) || vestingSchedule.revoked == true (privateVesting.sol#1330)
        - currentTime >= start.add(duration) (privateVesting.sol#1333)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (privateVesting.sol#478-488) uses assembly
        - INLINE ASM (privateVesting.sol#484-486)
Address.verifyCallResult(bool,bytes,string) (privateVesting.sol#684-704) uses assembly
        - INLINE ASM (privateVesting.sol#696-699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Whitelist.addAddress(address) (privateVesting.sol#936-940) compares to a boolean constant:
        -require(bool,string)(whitelistedMap[_address] != true,Account already exist) (privateVesting.sol#937)
Whitelist.removeAddress(address) (privateVesting.sol#942-946) compares to a boolean constant:
        -require(bool)(whitelistedMap[_address] != false) (privateVesting.sol#943)
TokenVesting.createVestingSchedule(address,bool,uint256) (privateVesting.sol#1106-1142) compares to a boolean constant:
        -require(bool,string)(isWhitelisted(_beneficiary) != false,Whitelist: Beneficiary must be pre-regesitred) (privateVesting.sol#1121-1124)
TokenVesting.revoke(bytes32) (privateVesting.sol#1148-1171) compares to a boolean constant:
        -require(bool,string)(vestingSchedule.revocable == true,TokenVesting: vesting is not revocable) (privateVesting.sol#1156-1159)
TokenVesting._computeReleasableAmount(TokenVesting.VestingSchedule) (privateVesting.sol#1323-1347) compares to a boolean constant:
        -(currentTime < start.add(cliff_)) || vestingSchedule.revoked == true (privateVesting.sol#1330)
TokenVesting.onlyIfVestingScheduleExists(bytes32) (privateVesting.sol#1005-1008) compares to a boolean constant:
        -require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (privateVesting.sol#1006)
TokenVesting.onlyIfVestingScheduleNotRevoked(bytes32) (privateVesting.sol#1013-1017) compares to a boolean constant:
        -require(bool)(vestingSchedules[vestingScheduleId].initialized == true) (privateVesting.sol#1014)
TokenVesting.onlyIfVestingScheduleNotRevoked(bytes32) (privateVesting.sol#1013-1017) compares to a boolean constant:
```

```
Different versions of Solidity is used:
        - Version used: ['0.8.5', '^0.8.0', '^0.8.5']
        - ^0.8.0 (privateVesting.sol#2)
        - ^0.8.0 (privateVesting.sol#249)
        - ^0.8.0 (privateVesting.sol#292)
        - ^0.8.0 (privateVesting.sol#316)
        - ^0.8.5 (privateVesting.sol#392)
        - ^0.8.0 (privateVesting.sol#455)
        - ^0.8.0 (privateVesting.sol#709)
        - ^0.8.0 (privateVesting.sol#800)
        - 0.8.5 (privateVesting.sol#956)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (privateVesting.sol#537-542) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (privateVesting.sol#569-581) is never used and should be removed
Address.functionDelegateCall(address,bytes) (privateVesting.sol#649-659) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (privateVesting.sol#667-676) is never used and should be removed
Address.functionStaticCall(address,bytes) (privateVesting.sol#613-624) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (privateVesting.sol#632-641) is never used and should be removed
Address.sendValue(address,uint256) (privateVesting.sol#506-517) is never used and should be removed
Context._msgData() (privateVesting.sol#309-311) is never used and should be removed
Math.average(uint256,uint256) (privateVesting.sol#273-276) is never used and should be removed
Math.ceilDiv(uint256,uint256) (privateVesting.sol#284-287) is never used and should be removed
Math.max(uint256,uint256) (privateVesting.sol#258-260) is never used and should be removed
Math.min(uint256,uint256) (privateVesting.sol#265-267) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (privateVesting.sol#844-860) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (privateVesting.sol#878-899) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (privateVesting.sol#862-876) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (privateVesting.sol#825-835) is never used and should be removed
SafeMath.div(uint256,uint256,string) (privateVesting.sol#209-218) is never used and should be removed
SafeMath.mod(uint256,uint256) (privateVesting.sol#169-171) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (privateVesting.sol#235-244) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (privateVesting.sol#186-195) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (privateVesting.sol#20-30) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (privateVesting.sol#74-83) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (privateVesting.sol#90-99) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (privateVesting.sol#53-67) is never used and should be removed
SafeMath.trySub(uint256,uint256) (privateVesting.sol#37-46) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (privateVesting.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (privateVesting.sol#249) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (privateVesting.sol#292) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (privateVesting.sol#316) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.5 (privateVesting.sol#392) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (privateVesting.sol#455) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (privateVesting.sol#709) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (privateVesting.sol#800) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.5 (privateVesting.sol#956) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.5 is not recommended for deployment
```

```
Parameter Whitelist.whitelisted(address)._address (privateVesting.sol#932) is not in mixedCase
Parameter Whitelist.addAddress(address)._address (privateVesting.sol#936) is not in mixedCase
Parameter Whitelist.removeAddress(address)._address (privateVesting.sol#942) is not in mixedCase
Parameter Whitelist.isWhitelisted(address)._address (privateVesting.sol#948) is not in mixedCase
Constant Whitelist.version (privateVesting.sol#927) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter TokenVesting.getVestingSchedulesCountByBeneficiary(address)._beneficiary (privateVesting.sol#1047) is not in mixedCase
Parameter TokenVesting.createVestingSchedule(address,bool,uint256)._beneficiary (privateVesting.sol#1107) is not in mixedCase
Parameter TokenVesting.createVestingSchedule(address,bool,uint256)._revocable (privateVesting.sol#1108) is not in mixedCase
Parameter TokenVesting.createVestingSchedule(address,bool,uint256)._amount (privateVesting.sol#1109) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (privateVesting.sol#367-369)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (privateVesting.sol#375-381)
whitelisted(address) should be declared external:
        - Whitelist.whitelisted(address) (privateVesting.sol#932-934)
addAddress(address) should be declared external:
        - Whitelist.addAddress(address) (privateVesting.sol#936-940)
removeAddress(address) should be declared external:
        - Whitelist.removeAddress(address) (privateVesting.sol#942-946)
createVestingSchedule(address,bool,uint256) should be declared external:
        - TokenVesting.createVestingSchedule(address,bool,uint256) (privateVesting.sol#1106-1142)
revoke(bytes32) should be declared external:
        - TokenVesting.revoke(bytes32) (privateVesting.sol#1148-1171)
withdraw(uint256) should be declared external:
        - TokenVesting.withdraw(uint256) (privateVesting.sol#1177-1183)
computeReleasableAmount(bytes32) should be declared external:
        - TokenVesting.computeReleasableAmount(bytes32) (privateVesting.sol#1230-1240)
getWithdrawableAmount() should be declared external:
        - TokenVesting.getWithdrawableAmount() (privateVesting.sol#1258-1260)
computeNextVestingScheduleIdForHolder(address) should be declared external:
        - TokenVesting.computeNextVestingScheduleIdForHolder(address) (privateVesting.sol#1265-1275)
getLastVestingScheduleForHolder(address) should be declared external:
        - TokenVesting.getLastVestingScheduleForHolder(address) (privateVesting.sol#1281-1293)
computeByteIdForAddress(address) should be declared external:
        - TokenVesting.computeByteIdForAddress(address) (privateVesting.sol#1310-1316)
getLockedAmount() should be declared external:
        - TokenVesting.getLockedAmount() (privateVesting.sol#1349-1351)
startTime() should be declared external:
        - TokenVesting.startTime() (privateVesting.sol#1357-1359)
endTime() should be declared external:
        - TokenVesting.endTime() (privateVesting.sol#1361-1363)
getCliff() should be declared external:
        - TokenVesting.getCliff() (privateVesting.sol#1365-1367)
getAmountWithdrawn() should be declared external:
        - TokenVesting.getAmountWithdrawn() (privateVesting.sol#1369-1371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (27 contracts with 75 detectors), 194 result(s) found
```

Results

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

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. Many issues were discovered during the audit;the majority of the issues are fixed by the team.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **99Starz** Contracts. 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 **99Starz** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
November, 2021

For

99STARZ

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com

QuillAudits