QuillAudits

Audit Report
September, 2021

For

locklet

Contents

Contents

Scope of the Audit

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

Checked Vulnerabilities

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

- 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**
Mythril, Slither, SmartCheck, Surya, Solhint.

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 four 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	1	2	0
Closed	1	1	5	1

Introduction

During the period of **September 13, 2021, to September 18, 2021** - QuillAudits Team performed a security audit for LockletTokenVault smart contract.

The code for the audit was taken from the following official repo of Locklet: https://github.com/locklet/locklet-evm-contracts/blob/main/contracts/LockletTokenVault.sol

V	Date	Commit hash	Note
1	September	7fd89cbcf96a10429b9cfeee295dd4c51c2af982	Version 1
2	September	438433345bd1757826f1f21ba08a09e4f3e2c24a	Version 2

Issues Found

A. Contract – LockletTokenVault

High severity issues

1. Invalid fee calculation

```
Line 225:
LockletToken lktToken = lockletToken();
        require(lktToken.balanceOf(msg.sender) >= _revocationFlatFeeLktAmount,
"LockletTokenVault: Not enough LKT to pay fees");
        require(lktToken.transferFrom(msg.sender, address(this),
_revocationFlatFeeLktAmount));
      uint256 burnAmount = _creationFlatFeeLktAmount.div(100).mul(45);
      uint256 stakersRedisAmount = _creationFlatFeeLktAmount.div(100).mul(45);
      uint256 foundationRedisAmount = _creationFlatFeeLktAmount.div(100)
.mul(10);
```

Description
In the contract, the user has to pay the revocation fee in order to revoke a lock that he created. However, in the revokeLock function, the user pays the revoke fee which is the value of the _revocationFlatFeeLktAmount variable but the amount that is distributed between the stakers and the Locklet foundation is using the value of the _creationFlatFeeLktAmount variable. The risk can affect either the initiator or the stakers and the Locklet foundation. When the revocation fee is higher than the creation fee a part of the initiator's tokens will go to the contract without getting transferred to any of the other parties. In the other case where the revocation fee will be lower than the creation fee, the initiator will pay less than the number of tokens that are going to be distributed between the stackers and the Locklet foundation.

Remediation
Change the value that is distributed in the revokeLock function from the _creationFlatFeeLktAmount variable to the _revocationFlatFeeLktAmount.

Solved
The Locklet team has fixed the issue in version 2 by changing the _creationFlatFeeLktAmount variable to _revocationFlatFeeLktAmount in the calculation of the revoke fee.

Medium severity issues

2. Race Condition

```
Line 117:
if (payFeesWithLkt) {
        LockletToken lktToken = lockletToken();
        require(lktToken.balanceOf(msg.sender) >= _creationFlatFeeLktAmount,
          "LockletTokenVault: Not enough LKT to pay fees");
        require(lktToken.transferFrom(msg.sender, address(this),
            _creationFlatFeeLktAmount));
```

```
Line 132:
else {
        uint256 creationPercentFeeAmount = totalAmount.div(10000).mul(
          _creationPercentFee);
        uint256 totalAmountWithFees = totalAmount.add(
          creationPercentFeeAmount);
        require(token.balanceOf(msg.sender) >= totalAmountWithFees,
          "LockletTokenVault: Token insufficient balance");
        require(token.transferFrom(msg.sender, address(this),
            totalAmountWithFees));
```

```
Line 225:
LockletToken lktToken = lockletToken();
      require(lktToken.balanceOf(msg.sender) >= _revocationFlatFeeLktAmount,
        "LockletTokenVault: Not enough LKT to pay fees");
      require(lktToken.transferFrom(msg.sender, address(this),
          _revocationFlatFeeLktAmount));
```

Description

In the contract the user can call the function getCreationFlatFeeLktAmount to check the _creationFlatFeeLktAmount then he might decide to create a lock but at the same time, the governor might have called setCreationFlatFeeLktAmount to modify the creation fee. In the scenario where the governor's transaction gets mined first, the user could possibly create a lock with a higher or lower than the one that was returned by getCreationFlatFeeLktAmount.
The same logic applies with the _revocationFlatFeeLktAmount and _creationPercentFee variables.

Remediation

Add the creation fee and the creation fee percent as arguments to the addLock function and add a require that verifies that the fees provided in the arguments are the same as the one that is stored in the smart contract. The same logic should be applied to the revokeLock function.

Solved
The Locklet team has fixed the issue in version 2 by pausing the lock creation and the revocation from the front end until the governance actions are done.

3. Governor Can Deny Any Lock Revocation:

```
Line 226:
require(lktToken.balanceOf(msg.sender) >= _revocationFlatFeeLktAmount,
          "LockletTokenVault: Not enough LKT to pay fees");
      require(lktToken.transferFrom(msg.sender, address(this),
              _revocationFlatFeeLktAmount));
```

```
Line 437:
    function setRevocationFlatFeeLktAmount(uint256 amount) external
onlyGovernor {
            _revocationFlatFeeLktAmount = amount;
        }
```

Description
In order for the initiator to revoke a lock, he needs to pay the revocation. On the other hand, the governor can change the revocation fee to any value which will give him the option to choose a huge revocation fee to prevent all the initiators from revoking their locks.

Remediation
Set a limitation for the revocation fee that can be set by the governor.

Acknowledged
The Locklet team has acknowledged the risk by stating that it's really hard for them to set a limit because the $LKT price can vary much from now.

Low level severity issues

4. Renounce Ownership

```
Line 15:
contract LockletTokenVault is AccessControl, Pausable {
```

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.

Acknowledged
The Locklet team has acknowledged the risk by stating that in the long term they will give up full ownership of the contract to another governance contract that will be granted a GOVERNOR role and this contract will be solely responsible for governance actions.

5. Floating Pragma

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.

Solved
The Locklet team has fixed the issue in version 2 by locking the pragma version to 0.8.3.

6. Missing Value Verification

```
Line 441:
function setCreationPercentFee(uint256 amount) external onlyGovernor {
        _creationPercentFee = amount;
    }
```

Description
Certain functions lack a safety check in the value, the amount argument should be lower than 100. Otherwise, the contract's logic might be harmed.

Remediation
Add a require or modifier in order to verify that the input value is lower than 10000.

Solved
The Locklet team has fixed the issue in version 2 by adding a require that verifies that the amount argument is lower than 10000.

7. Loop over a dynamic array

```
Line 165:
for (uint256 i = 0; i < recipientsData.length; i++) {
            RecipientCallData calldata recipientData = recipientsData[i];
            uint256 unlockedAmountPerDay = recipientData.amount.div(
                durationInDays);
            require(unlockedAmountPerDay > 0,
             "LockletTokenVault: The unlocked amount per day is equal to zero");
            totalAmountCheck = totalAmountCheck.add(recipientData.amount);
            Recipient memory recipient = Recipient({
                recipientAddress: recipientData.recipientAddress,
                amount: recipientData.amount,
                daysClaimed: 0,
                amountClaimed: 0,
                isActive: true
            });
            _recipientsLocksIndexes[recipientData.recipientAddress].push(
                lockIndex);
            _locksRecipients[lockIndex].push(recipient);
            }
           require(totalAmountCheck == totalAmount,
              "LockletTokenVault: The calculated total amount is not equal to the actual total
amount");
            emit LockAdded(lockIndex);
        }
```

```
Line 245:
for (uint256 i = 0; i < recipients.length; i++) {
            Recipient storage recipient = recipients[i];
            totalAmount = totalAmount.add(recipient.amount);
            uint16 daysVested;
            uint256 unlockedAmount;
            (daysVested, unlockedAmount) = calculateClaim(lock, recipient);
            if (unlockedAmount > 0) {
                address recipientAddr = recipient.recipientAddress;
                _refunds[recipientAddr][tokenAddr] = _refunds[recipientAddr][
                    tokenAddr
                ].add(unlockedAmount);
            }
            totalUnlockedAmount = totalUnlockedAmount.add(
                recipient.amountClaimed.add(unlockedAmount)
            );
        }
```

```
Line 310:
for (currentLockIndex; currentLockIndex >= queryEndLockIndex;
            currentLockIndex--) {
            uint256 currentLockIndexAsUnsigned = uint256(currentLockIndex);
            if (currentLockIndexAsUnsigned <= getLocksLength().sub(1)) {
                results[index] = getLock(currentLockIndexAsUnsigned);
            }
            index++;
        }
```

```
Line 328:
for (uint256 index = 0; index < initiatorLocksLength; index++) {
            uint256 lockIndex = _initiatorsLocksIndexes[initiatorAddress][
                index];
            results[index] = getLock(lockIndex);
        }
```

```
Line 342:
for (uint256 index = 0; index < recipientLocksLength; index++) {
            uint256 lockIndex = _recipientsLocksIndexes[recipientAddress][
                index];
            results[index] = getLock(lockIndex);
        }
```

```
Line 390:
for (uint256 i = 0; i < recipients.length; i++) {
            if (recipients[i].recipientAddress == recipientAddress) {
                recipientIndex = int256(i);
                break;
            }
        }
```

Description

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Remediation
Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Acknowledged
The Locklet team has acknowledged the risk knowing that the arrays won't exceed a specific amount so looping over these arrays won't consume much.

8. Usage of block.timestamp:

```
Line 402:
if (block.timestamp < lock.startTime) {
            return (0, 0);
        }
```

```
Line 421:
function blockTime() private view 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 Locklet team has acknowledged the risk knowing that a 900 seconds delay won't hurt the business logic.

9. Missing Address Verification

```
Line 86:
constructor(address lockletTokenAddr) {
        lockletTokenAddress = lockletTokenAddr;
        _nextLockIndex = 0;
        _stakersRedisAddress = address(0);
        _foundationRedisAddress= 0x25Bd291bE258E90e7A0648aC5c690555aA9e8930;
        _isDeprecated = false;
        _creationFlatFeeLktAmount = 0;
        _revocationFlatFeeLktAmount = 0;
        _creationPercentFee = 35;
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _setupRole(GOVERNOR_ROLE, msg.sender);
    }
```

```
Line 445:
function setStakersRedisAddress(address addr) external onlyGovernor {
        _stakersRedisAddress = addr;
    }
```

Description

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible or tokens may be burned in perpetuity.

Remediation

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Solved

The Locklet team has fixed the issue in version 2 by adding a require that verifies that the argument address is different than the zero-address.

10. Divide Before Multiply

```
Line 122:
uint256 burnAmount = _creationFlatFeeLktAmount.div(100).mul(45);
    uint256 stakersRedisAmount = _creationFlatFeeLktAmount.div(100).mul(45);
    uint256 foundationRedisAmount= _creationFlatFeeLktAmount.div(100).mul(10);
```

```
Line 133:
uint256 creationPercentFeeAmount = totalAmount.div(10000)
.mul(_creationPer centFee);
```

```
Line 139:
uint256 stakersRedisAmount = creationPercentFeeAmount.div(100).mul(90);
uint256 foundationRedisAmount = creationPercentFeeAmount.div(100).mul(10);
```

```
Line 229:
uint256 burnAmount = _creationFlatFeeLktAmount.div(100).mul(45);
    uint256 stakersRedisAmount = _creationFlatFeeLktAmount.div(100).mul(45);
    uint256 foundationRedisAmount= _creationFlatFeeLktAmount.div(100).mul(10);
```

Description

Integer division in solidity may truncate. As a result, dividing before multiplying may result in a loss of precision. Due to precision's sensitivity, this may result in certain abnormalities in the contract's logic

Remediation

Do the multiplication operations before the division operations.

Solved

The Locklet team has fixed in version 2 the issue by doing the multiplications before divisions.

Informational

11. Usage of SafeMath and SignedSafeMath in solidity ^0.8.0:

```
Line 16:
using SafeMath for uint256;
        using SafeMath for uint16;

        using SignedSafeMath for int256;
```

Description
SafeMath and SignedSafeMath are no longer needed starting with Solidity 0.8. The compiler now has built-in overflow checking.

Remediation
Use the mathematical operations directly since the risk of the overflow is already remediated in the latest versions of solidity.

Solved

Automated Tests

Slither:

```
INFO:Detectors:
LockletToken._totalSupply (LockletToken.sol#10) shadows:
        - ERC20._totalSupply (token/ERC20/ERC20.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
Context._msgData() (utils/Context.sol#20-22) 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 (LockletToken.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.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:
LockletToken.constructor() (LockletToken.sol#12-16) uses literals with too many digits:
        - _initialSupply = 150000000 * 10 ** 18 (LockletToken.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (token/ERC20/ERC20.sol#112-115)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (token/ERC20/ERC20.sol#120-122)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
```

```
        - ERC20.transferFrom(address,address,uint256) (token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (token/ERC20/ERC20.sol#196-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
ERROR:ContractSolcParsing:Missing function 'name'
INFO:Detectors:
LockletToken._totalSupply (LockletToken.sol#10) shadows:
        - ERC20._totalSupply (token/ERC20/ERC20.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188) performs a multiplication on the result of a division:
        -creationPercentFeeAmount = totalAmount.div(10000).mul(_creationPercentFee) (LockletTokenVault.sol#133)
LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188) performs a multiplication on the result of a division:
        -burnAmount = _creationFlatFeeLktAmount.div(100).mul(45) (LockletTokenVault.sol#122)
LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188) performs a multiplication on the result of a division:
        -stakersRedisAmount = _creationFlatFeeLktAmount.div(100).mul(45) (LockletTokenVault.sol#123)
LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188) performs a multiplication on the result of a division:
        -stakersRedisAmount_scope_0 = creationPercentFeeAmount.div(100).mul(90) (LockletTokenVault.sol#139)
LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188) performs a multiplication on the result of a division:
        -foundationRedisAmount = _creationFlatFeeLktAmount.div(100).mul(10) (LockletTokenVault.sol#124)
LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188) performs a multiplication on the result of a division:
        -foundationRedisAmount_scope_1 = creationPercentFeeAmount.div(100).mul(10) (LockletTokenVault.sol#140)
LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266) performs a multiplication on the result of a division:
        -burnAmount = _creationFlatFeeLktAmount.div(100).mul(45) (LockletTokenVault.sol#229)
LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266) performs a multiplication on the result of a division:
        -stakersRedisAmount = _creationFlatFeeLktAmount.div(100).mul(45) (LockletTokenVault.sol#230)
LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266) performs a multiplication on the result of a division:
        -foundationRedisAmount = _creationFlatFeeLktAmount.div(100).mul(10) (LockletTokenVault.sol#231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
LockletTokenVault.constructor(address).lockletTokenAddr (LockletTokenVault.sol#86) lacks a zero-check on :
                - lockletTokenAddress = lockletTokenAddr (LockletTokenVault.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188):
        External calls:
        - require(bool)(lktToken.transferFrom(msg.sender,address(this),_creationFlatFeeLktAmount)) (LockletTokenVault.sol#120)
        - require(bool)(lktToken.burn(burnAmount)) (LockletTokenVault.sol#126)
        - require(bool)(lktToken.transfer(_stakersRedisAddress,stakersRedisAmount)) (LockletTokenVault.sol#127)
        - require(bool)(lktToken.transfer(_foundationRedisAddress,foundationRedisAmount)) (LockletTokenVault.sol#128)
        - require(bool)(token.transferFrom(msg.sender,address(this),totalAmount)) (LockletTokenVault.sol#131)
        - require(bool)(token.transferFrom(msg.sender,address(this),totalAmountWithFees)) (LockletTokenVault.sol#137)
        - require(bool)(token.transfer(_stakersRedisAddress,stakersRedisAmount_scope_0)) (LockletTokenVault.sol#142)
        - require(bool)(token.transfer(_foundationRedisAddress,foundationRedisAmount_scope_1)) (LockletTokenVault.sol#143)
        State variables written after the call(s):
        - _initiatorsLocksIndexes[msg.sender].push(lockIndex) (LockletTokenVault.sol#161)
        - _locks.push(lock) (LockletTokenVault.sol#160)
        - _locksRecipients[lockIndex].push(recipient) (LockletTokenVault.sol#182)
        - _nextLockIndex = _nextLockIndex.add(1) (LockletTokenVault.sol#147)
        - _recipientsLocksIndexes[recipientData.recipientAddress].push(lockIndex) (LockletTokenVault.sol#181)
Reentrancy in LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266):
        External calls:
        - require(bool)(lktToken.transferFrom(msg.sender,address(this),_revocationFlatFeeLktAmount)) (LockletTokenVault.sol#227)
        - require(bool)(lktToken.burn(burnAmount)) (LockletTokenVault.sol#233)
        - require(bool)(lktToken.transfer(_stakersRedisAddress,stakersRedisAmount)) (LockletTokenVault.sol#234)
        - require(bool)(lktToken.transfer(_foundationRedisAddress,foundationRedisAmount)) (LockletTokenVault.sol#235)
        State variables written after the call(s):
        - _refunds[recipientAddr][tokenAddr] = _refunds[recipientAddr][tokenAddr].add(unlockedAmount) (LockletTokenVault.sol#256)
        - _refunds[initiatorAddr][tokenAddr] = _refunds[initiatorAddr][tokenAddr].add(totalLockedAmount) (LockletTokenVault.sol#263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
```

```
Reentrancy in LockletTokenVault.addLock(address,uint256,uint16,uint16,LockletTokenVault.RecipientCallData[],bool,bool) (LockletTokenVault.sol#103-188):
        External calls:
        - require(bool)(lktToken.transferFrom(msg.sender,address(this),_creationFlatFeeLktAmount)) (LockletTokenVault.sol#120)
        - require(bool)(lktToken.burn(burnAmount)) (LockletTokenVault.sol#126)
        - require(bool)(lktToken.transfer(_stakersRedisAddress,stakersRedisAmount)) (LockletTokenVault.sol#127)
        - require(bool)(lktToken.transfer(_foundationRedisAddress,foundationRedisAmount)) (LockletTokenVault.sol#128)
        - require(bool)(token.transferFrom(msg.sender,address(this),totalAmount)) (LockletTokenVault.sol#131)
        - require(bool)(token.transferFrom(msg.sender,address(this),totalAmountWithFees)) (LockletTokenVault.sol#137)
        - require(bool)(token.transfer(_stakersRedisAddress,stakersRedisAmount_scope_0)) (LockletTokenVault.sol#142)
        - require(bool)(token.transfer(_foundationRedisAddress,foundationRedisAmount_scope_1)) (LockletTokenVault.sol#143)
        Event emitted after the call(s):
        - LockAdded(lockIndex) (LockletTokenVault.sol#187)
Reentrancy in LockletTokenVault.claimLockedTokens(uint256) (LockletTokenVault.sol#190-214):
        External calls:
        - require(bool,string)(token.transfer(recipient.recipientAddress,unlockedAmount),LockletTokenVault: Unlocked tokens transfer failed) (LockletTokenVault.sol#212)
        Event emitted after the call(s):
        - LockedTokensClaimed(lockIndex,recipient.recipientAddress,unlockedAmount) (LockletTokenVault.sol#213)
Reentrancy in LockletTokenVault.pullRefund(address) (LockletTokenVault.sol#268-278):
        External calls:
        - require(bool,string)(token.transfer(msg.sender,refundAmount),LockletTokenVault: Refund tokens transfer failed) (LockletTokenVault.sol#275)
        Event emitted after the call(s):
        - LockRefundPulled(msg.sender,tokenAddress,refundAmount) (LockletTokenVault.sol#277)
Reentrancy in LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266):
        External calls:
        - require(bool)(lktToken.transferFrom(msg.sender,address(this),_revocationFlatFeeLktAmount)) (LockletTokenVault.sol#227)
        - require(bool)(lktToken.burn(burnAmount)) (LockletTokenVault.sol#233)
        - require(bool)(lktToken.transfer(_stakersRedisAddress,stakersRedisAmount)) (LockletTokenVault.sol#234)
        - require(bool)(lktToken.transfer(_foundationRedisAddress,foundationRedisAmount)) (LockletTokenVault.sol#235)
        Event emitted after the call(s):
        - LockRevoked(lockIndex,totalUnlockedAmount,totalLockedAmount) (LockletTokenVault.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (utils/Address.sol#26-36) uses assembly
        - INLINE ASM (utils/Address.sol#32-34)
Address.verifyCallResult(bool,bytes,string) (utils/Address.sol#195-215) uses assembly
        - INLINE ASM (utils/Address.sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LockletTokenVault.claimLockedTokens(uint256) (LockletTokenVault.sol#190-214) compares to a boolean constant:
        -require(bool,string)(lock.isRevoked == false,LockletTokenVault: This lock has been revoked) (LockletTokenVault.sol#193)
LockletTokenVault.claimLockedTokens(uint256) (LockletTokenVault.sol#190-214) compares to a boolean constant:
        -require(bool,string)(lock.isActive == true,LockletTokenVault: Lock not existing) (LockletTokenVault.sol#192)
LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266) compares to a boolean constant:
        -require(bool,string)(lock.isActive == true,LockletTokenVault: Lock not existing) (LockletTokenVault.sol#218)
LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266) compares to a boolean constant:
        -require(bool,string)(lock.isRevoked == false,LockletTokenVault: This lock has already been revoked) (LockletTokenVault.sol#221)
LockletTokenVault.revokeLock(uint256) (LockletTokenVault.sol#216-266) compares to a boolean constant:
        -require(bool,string)(lock.isRevocable == true,LockletTokenVault: Lock not revocable) (LockletTokenVault.sol#220)
LockletTokenVault.getLock(uint256) (LockletTokenVault.sol#282-287) compares to a boolean constant:
        -require(bool,string)(lock.isActive == true,LockletTokenVault: Lock not existing) (LockletTokenVault.sol#284)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
AccessControl._checkRole(bytes32,address) (access/AccessControl.sol#94-107) is never used and should be removed
AccessControl._grantRole(bytes32,address) (access/AccessControl.sol#204-209) is never used and should be removed
AccessControl._revokeRole(bytes32,address) (access/AccessControl.sol#216-221) is never used and should be removed
AccessControl._setRoleAdmin(bytes32,bytes32) (access/AccessControl.sol#193-197) is never used and should be removed
Address.functionCall(address,bytes) (utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (utils/Address.sol#108-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (utils/Address.sol#141-143) is never used and should be removed
```

```
Address.functionCall(address,bytes) (utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (utils/Address.sol#108-114) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (utils/Address.sol#122-133) is never used and should be removed
Address.functionDelegateCall(address,bytes) (utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (utils/Address.sol#178-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (utils/Address.sol#151-160) is never used and should be removed
Address.sendValue(address,uint256) (utils/Address.sol#54-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (utils/Address.sol#195-215) is never used and should be removed
Context._msgData() (utils/Context.sol#20-22) is never used and should be removed
Pausable._pause() (security/Pausable.sol#74-77) is never used and should be removed
Pausable._unpause() (security/Pausable.sol#86-89) is never used and should be removed
SafeMath.div(uint256,uint256,string) (utils/math/SafeMath.sol#190-199) is never used and should be removed
SafeMath.mod(uint256,uint256) (utils/math/SafeMath.sol#150-152) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (utils/math/SafeMath.sol#216-225) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (utils/math/SafeMath.sol#167-176) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (utils/math/SafeMath.sol#21-27) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (utils/math/SafeMath.sol#63-68) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (utils/math/SafeMath.sol#75-80) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (utils/math/SafeMath.sol#46-56) is never used and should be removed
SafeMath.trySub(uint256,uint256) (utils/math/SafeMath.sol#34-39) is never used and should be removed
SignedSafeMath.add(int256,int256) (utils/math/SignedSafeMath.sol#64-66) is never used and should be removed
SignedSafeMath.div(int256,int256) (utils/math/SignedSafeMath.sol#36-38) is never used and should be removed
SignedSafeMath.mul(int256,int256) (utils/math/SignedSafeMath.sol#22-24) is never used and should be removed
SignedSafeMath.sub(int256,int256) (utils/math/SignedSafeMath.sol#50-52) is never used and should be removed
Strings.toHexString(uint256) (utils/Strings.sol#39-50) is never used and should be removed
Strings.toHexString(uint256,uint256) (utils/Strings.sol#55-65) is never used and should be removed
Strings.toString(uint256) (utils/Strings.sol#14-34) 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 (LockletToken.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (LockletTokenVault.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (access/AccessControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (access/IAccessControl.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (security/Pausable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Strings.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/introspection/ERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/introspection/IERC165.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/math/SafeMath.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/math/SignedSafeMath.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:
Low level call in Address.sendValue(address,uint256) (utils/Address.sol#54-59):
        - (success) = recipient.call{value: amount}() (utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (utils/Address.sol#122-133):
        - (success,returndata) = target.call{value: value}(data) (utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (utils/Address.sol#151-160):
        - (success,returndata) = target.staticcall(data) (utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (utils/Address.sol#178-187):
        - (success,returndata) = target.delegatecall(data) (utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
LockletToken.constructor() (LockletToken.sol#12-16) uses literals with too many digits:
        - _initialSupply = 150000000 * 10 ** 18 (LockletToken.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
```

```
INFO:Detectors:
LockletTokenVault (LockletTokenVault.sol#15-476) does not implement functions:
        - AccessControl._checkRole(bytes32,address) (access/AccessControl.sol#94-107)
        - AccessControl._grantRole(bytes32,address) (access/AccessControl.sol#204-209)
        - Context._msgData() (utils/Context.sol#20-22)
        - Context._msgSender() (utils/Context.sol#16-18)
        - Pausable._pause() (security/Pausable.sol#74-77)
        - AccessControl._revokeRole(bytes32,address) (access/AccessControl.sol#216-221)
        - AccessControl._setRoleAdmin(bytes32,bytes32) (access/AccessControl.sol#193-197)
        - AccessControl._setupRole(bytes32,address) (access/AccessControl.sol#184-186)
        - Pausable._unpause() (security/Pausable.sol#86-89)
        - LockletTokenVault.blockTime() (LockletTokenVault.sol#421-423)
        - LockletTokenVault.calculateClaim(LockletTokenVault.Lock,LockletTokenVault.Recipient) (LockletTokenVault.sol#399-419)
        - LockletTokenVault.getClaimByLockAndRecipient(uint256,address) (LockletTokenVault.sol#354-370)
        - LockletTokenVault.getCreationFlatFeeLktAmount() (LockletTokenVault.sol#372-374)
        - LockletTokenVault.getCreationPercentFee() (LockletTokenVault.sol#380-382)
        - LockletTokenVault.getLocksByInitiator(address) (LockletTokenVault.sol#322-334)
        - LockletTokenVault.getLocksByRecipient(address) (LockletTokenVault.sol#336-348)
        - LockletTokenVault.getRecipientIndexByAddress(LockletTokenVault.Recipient[],address) (LockletTokenVault.sol#388-397)
        - LockletTokenVault.getRefundAmount(address) (LockletTokenVault.sol#350-352)
        - LockletTokenVault.getRevocationFlatFeeLktAmount() (LockletTokenVault.sol#376-378)
        - AccessControl.getRoleAdmin(bytes32) (access/AccessControl.sol#115-117)
        - AccessControl.grantRole(bytes32,address) (access/AccessControl.sol#129-131)
        - AccessControl.hasRole(bytes32,address) (access/AccessControl.sol#83-85)
        - LockletTokenVault.isDeprecated() (LockletTokenVault.sol#384-386)
        - LockletTokenVault.lockletToken() (LockletTokenVault.sol#425-427)
        - LockletTokenVault.pause() (LockletTokenVault.sol#449-451)
        - Pausable.paused() (security/Pausable.sol#39-41)
        - AccessControl.renounceRole(bytes32,address) (access/AccessControl.sol#160-164)
        - AccessControl.revokeRole(bytes32,address) (access/AccessControl.sol#142-144)
        - LockletTokenVault.setCreationFlatFeeLktAmount(uint256) (LockletTokenVault.sol#433-435)
        - LockletTokenVault.setCreationPercentFee(uint256) (LockletTokenVault.sol#441-443)
        - LockletTokenVault.setDeprecated(bool) (LockletTokenVault.sol#457-459)
        - LockletTokenVault.setRevocationFlatFeeLktAmount(uint256) (LockletTokenVault.sol#437-439)
        - LockletTokenVault.setStakersRedisAddress(address) (LockletTokenVault.sol#445-447)
        - AccessControl.supportsInterface(bytes4) (access/AccessControl.sol#76-78)
        - LockletTokenVault.unpause() (LockletTokenVault.sol#453-455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
AccessControl._roles (access/AccessControl.sol#54) is never used in LockletTokenVault (LockletTokenVault.sol#15-476)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
getLock(uint256) should be declared external:
        - LockletTokenVault.getLock(uint256) (LockletTokenVault.sol#282-287)
getLocksLength() should be declared external:
        - LockletTokenVault.getLocksLength() (LockletTokenVault.sol#289-291)
getLocks(int256,int256) should be declared external:
        - LockletTokenVault.getLocks(int256,int256) (LockletTokenVault.sol#293-320)
getLocksByInitiator(address) should be declared external:
        - LockletTokenVault.getLocksByInitiator(address) (LockletTokenVault.sol#322-334)
getLocksByRecipient(address) should be declared external:
        - LockletTokenVault.getLocksByRecipient(address) (LockletTokenVault.sol#336-348)
getClaimByLockAndRecipient(uint256,address) should be declared external:
        - LockletTokenVault.getClaimByLockAndRecipient(uint256,address) (LockletTokenVault.sol#354-370)
getCreationFlatFeeLktAmount() should be declared external:
        - LockletTokenVault.getCreationFlatFeeLktAmount() (LockletTokenVault.sol#372-374)
getRevocationFlatFeeLktAmount() should be declared external:
        - LockletTokenVault.getRevocationFlatFeeLktAmount() (LockletTokenVault.sol#376-378)
getCreationPercentFee() should be declared external:
        - LockletTokenVault.getCreationPercentFee() (LockletTokenVault.sol#380-382)
isDeprecated() should be declared external:
        - LockletTokenVault.isDeprecated() (LockletTokenVault.sol#384-386)
grantRole(bytes32,address) should be declared external:
```

```
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (access/AccessControl.sol#129-131)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (access/AccessControl.sol#142-144)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (access/AccessControl.sol#160-164)
name() should be declared external:
        - ERC20.name() (token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (token/ERC20/ERC20.sol#112-115)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (token/ERC20/ERC20.sol#120-122)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (token/ERC20/ERC20.sol#196-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
LockletPrivateSale.withdrawEth() (LockletPrivateSale.sol#101-103) sends eth to arbitrary user
        Dangerous calls:
        - address(msg.sender).transfer(address(this).balance) (LockletPrivateSale.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
LockletPrivateSale.claim() (LockletPrivateSale.sol#50-60) ignores return value by _lktToken.transfer(msg.sender,lktAmount) (LockletPrivateSale.sol#59)
LockletPrivateSale.withdrawLkt() (LockletPrivateSale.sol#105-107) ignores return value by _lktToken.transfer(msg.sender,_lktToken.balanceOf(address(this))) (LockletPrivateSale.sol#106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LockletPrivateSale.claim() (LockletPrivateSale.sol#50-60) compares to a boolean constant:
        -require(bool,string)(_claimable == true,LockletPrivateSale: Claim is not activated) (LockletPrivateSale.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._msgData() (utils/Context.sol#20-22) is never used and should be removed
ERC20._burn(address,uint256) (token/ERC20/ERC20.sol#274-289) is never used and should be removed
ERC20._mint(address,uint256) (token/ERC20/ERC20.sol#251-261) is never used and should be removed
SafeMath.div(uint256,uint256,string) (utils/math/SafeMath.sol#190-199) is never used and should be removed
SafeMath.mod(uint256,uint256) (utils/math/SafeMath.sol#150-152) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (utils/math/SafeMath.sol#216-225) is never used and should be removed
SafeMath.sub(uint256,uint256) (utils/math/SafeMath.sol#106-108) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (utils/math/SafeMath.sol#167-176) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (utils/math/SafeMath.sol#21-27) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (utils/math/SafeMath.sol#63-68) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (utils/math/SafeMath.sol#75-80) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (utils/math/SafeMath.sol#46-56) is never used and should be removed
SafeMath.trySub(uint256,uint256) (utils/math/SafeMath.sol#34-39) 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 (LockletPrivateSale.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (access/Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (security/Pausable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (token/ERC20/extensions/IERC20Metadata.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (utils/math/SafeMath.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:
Variable LockletPrivateSale._raisedEth (LockletPrivateSale.sol#17) is not in mixedCase
Variable LockletPrivateSale._soldedLkt (LockletPrivateSale.sol#18) is not in mixedCase
Variable LockletPrivateSale._lktPerEth (LockletPrivateSale.sol#20) is not in mixedCase
Variable LockletPrivateSale._maxEthPerAddr (LockletPrivateSale.sol#21) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
pause() should be declared external:
        - LockletPrivateSale.pause() (LockletPrivateSale.sol#81-83)
unpause() should be declared external:
        - LockletPrivateSale.unpause() (LockletPrivateSale.sol#85-87)
setLktPerEth(uint256) should be declared external:
        - LockletPrivateSale.setLktPerEth(uint256) (LockletPrivateSale.sol#89-91)
setMaxEthPerAddr(uint256) should be declared external:
        - LockletPrivateSale.setMaxEthPerAddr(uint256) (LockletPrivateSale.sol#93-95)
setClaimable(bool) should be declared external:
        - LockletPrivateSale.setClaimable(bool) (LockletPrivateSale.sol#97-99)
withdrawEth() should be declared external:
        - LockletPrivateSale.withdrawEth() (LockletPrivateSale.sol#101-103)
withdrawLkt() should be declared external:
        - LockletPrivateSale.withdrawLkt() (LockletPrivateSale.sol#105-107)
getRaisedEth() should be declared external:
        - LockletPrivateSale.getRaisedEth() (LockletPrivateSale.sol#113-115)
getSoldedLkt() should be declared external:
        - LockletPrivateSale.getSoldedLkt() (LockletPrivateSale.sol#117-119)
getLktPerEth() should be declared external:
        - LockletPrivateSale.getLktPerEth() (LockletPrivateSale.sol#121-123)
getMaxEthPerAddr() should be declared external:
        - LockletPrivateSale.getMaxEthPerAddr() (LockletPrivateSale.sol#125-127)
getClaimable() should be declared external:
        - LockletPrivateSale.getClaimable() (LockletPrivateSale.sol#135-137)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (access/Ownable.sol#61-64)
name() should be declared external:
        - ERC20.name() (token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
        - ERC20.symbol() (token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
        - ERC20.decimals() (token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
        - ERC20.totalSupply() (token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (token/ERC20/ERC20.sol#112-115)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (token/ERC20/ERC20.sol#120-122)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
```

```
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (token/ERC20/ERC20.sol#131-134)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (token/ERC20/ERC20.sol#149-163)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (token/ERC20/ERC20.sol#177-180)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (token/ERC20/ERC20.sol#196-204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:. analyzed (28 contracts with 75 detectors), 180 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

MythX

Report for LockletTokenVault.sol
https://dashboard.mythx.io/#/console/analyses/1c38a091-e4cd-4671-bece-e1c2e29276f5

Line	SWC Title	Severity	Short Description
4	(SWC-103) Floating Pragma	Low	A floating pragma is set.

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

Function Tests

Function Names	Testing results	Logic results
addLock	PASSED	PASSED
grantRole	PASSED	PASSED
pause	PASSED	PASSED
pullRefund	PASSED	PASSED
renounceRole	PASSED	PASSED
claimLockedTokens	PASSED	PASSED
revokeLock	PASSED	PASSED
revokeRole	PASSED	PASSED
setCreationFlatFeeLktAmount	PASSED	PASSED
setCreationPercentFee	PASSED	PASSED
setDeprecated	PASSED	PASSED
setRevocationFlatFeeLktAmount	PASSED	PASSED
setStakersRedisAddress	PASSED	PASSED
unpause	PASSED	PASSED

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.
No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract.
Many issues were discovered during the initial audit; all these vulnerabilities are fixed by the Locklet Team.

Disclaimer

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

Audit Report September, 2021

For

locklet

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com